

Tal Hassner · Ce Liu *Editors*

Dense Image Correspondences for Computer Vision

 Springer

Dense Image Correspondences for Computer Vision

Tal Hassner • Ce Liu
Editors

Dense Image Correspondences for Computer Vision



Springer

Editors

Tal Hassner
Department of Mathematics
and Computer Science
The Open University of Israel
Raanana, Israel

Ce Liu
Google Research
Cambridge, MA, USA

ISBN 978-3-319-23047-4

DOI 10.1007/978-3-319-23048-1

ISBN 978-3-319-23048-1 (eBook)

Library of Congress Control Number: 2015953102

Springer Cham Heidelberg New York Dordrecht London
© Springer International Publishing Switzerland 2016

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

*To my wife, Osnat, and my children, Ben,
Ella, Daniel, and May*

With love beyond words
– T.H.

To my wife, Irene, and my daughter, Mary
– C.L.

Preface

Correspondence estimation is a task of matching pixels of one image with those of another. When referring to *dense* correspondence estimation, the emphasis is on finding suitable matches (*correspondences*) for every one of those pixels.

Throughout much of the history of computer vision as a field of research, work on dense correspondence estimation has mostly been motivated by two specific problems: stereo vision, in which the pixels in one view of a 3D scene are matched with pixels in another view of the same scene to determine displacements and reason about 3D structure; and optical flow, in which the two images are taken from the same camera, but at different points in time. Many solutions have been offered for both these tasks, varying in their algorithmic design or the assumptions they make on the nature of the scene and imaging conditions. Implicit in most, however, is the *same scene* assumption: that is, that the two images involved capture the same physical scene, with possible differences due to independent scene motion.

This assumption plays an important role in defining the criteria for how pixels should be matched. If the same scene is visible in both images, then any physical scene point is expected to appear the same in both views. This similar appearance can therefore be used to match the pixels which capture its appearance in both images. Classical optical flow methods often make this concrete by using the *brightness constancy* assumption which interprets similar appearance as similar local patterns of pixel intensities.

In recent years, however, there has been a growing interest in breaking away from this same scene assumption and designing methods for correspondence estimation even in cases where the two images capture entirely different scenes. The rational for doing so is not entirely obvious and requires some explanation. For one thing, matching pixels between images of different scenes implies a harder problem: If the two images present different scenes, a physical point appearing in one image will obviously not appear in the other image and in particular cannot be expected to appear the same in both images. Therefore, the brightness constancy assumption cannot be applied in such cases and new criteria must be established in order to determine when two pixels actually match.

But even if correspondences could reliably be established even under these challenging circumstances, there is still the question of why this would even make sense. It is not immediately obvious what disparities can say about scene structure when two scenes are involved. Similarly, both scene and camera motion are not necessarily meaningful in such cases.

This book is intended to present the problem, solutions and applications of dense correspondence estimation with a particular emphasis on cross scene correspondences. Its chapters tackle the two issues raised above: first, by describing techniques designed to enable correspondence estimation under increasingly challenging conditions and second, by showing what can be done with these correspondences. The book is accordingly divided into the following two parts.

In Part I, we focus on *how* dense correspondences may be estimated. Chapter “Introduction to Dense Optical Flow” provides a survey of classical optical flow methods with an emphasis on the pioneering work of Horn/Schunck. Chapter “SIFT Flow: Dense Correspondence Across Scenes and Its Applications” takes dense correspondences beyond the same scene settings, by introducing the *SIFT flow* method. In chapter “Dense, Scale-Less Descriptors”, our focus shifts from the correspondence estimation method to the per pixel representation. It presents the *Scale Less SIFT* descriptor which widens the application of SIFT flow to images presenting information at different scales. A different approach to scale invariance in cross scene correspondence estimation, the *Scale-Space SIFT flow*, is described in chapter “Scale-Space SIFT Flow”. A descriptor design approach intended to improve the discriminative quality of the per pixel representation is provided in chapter “Dense Segmentation-Aware Descriptors”, which describes *segmentation-aware* descriptors. Chapter “SIFTpack: A Compact Representation for Efficient SIFT Matching” takes on a different concern: the storage and computational requirements often involved in dense correspondence estimation. The *SIFTpack* representation offers a compact and more efficient alternative to the Dense SIFT representation used by SIFT flow and related methods. Finally, whereas methods such as SIFT flow use a graph-based search for alignment, chapter “In Defense of Gradient-Based Alignment on Densely Sampled Sparse Features” explores an alternative approach of *gradient-based alignment* by continuous optimization.

In Part II, we focus on *why* dense correspondences are useful, even when they are computed between images of different scenes, by showing how they may be used to solve a wide range of computer vision problems. Specifically, chapter “From Images to Depths and Back” looks back to one of the early uses of cross scene dense correspondence estimation for estimating scene depth from a single view. The more recent *Depth Transfer* method for single view depth estimation by dense correspondence estimation is presented in chapter “DepthTransfer: Depth Extraction from Video Using Non-parametric Sampling”. Single image scene parsing by the *Label Transfer* method is described in chapter “Nonparametric Scene Parsing via Label Transfer”. The Label Transfer approach assumes many reference images with matching label information, which is transferred through dense correspondences to novel query images. The *Joint Inference* approach described in chapter “Joint Inference in Image Datasets via Dense Correspondence” shows how

this approach can be applied even when labels are available for only a few reference images. Finally, chapter “Dense Correspondences and Ancient Texts” takes dense correspondence estimation to an entirely different imaging domain and shows how dense correspondences may be used to process challenging ancient, handwritten texts.

Taken as a whole, this book shows that accurate dense correspondence estimation is possible, even under challenging settings, and can be key to solving image understanding problems in problem domains far beyond stereo and optical flow. We hope that this book will make the methods and applications of dense correspondence estimation accessible. Going beyond existing work, we would like to see this book motivate the development of new, more accurate, more robust and more efficient dense correspondence estimation techniques.

The editors are most grateful to all the friends and colleagues who have supported this book by contributing their work for its chapters: Xiang Bai, Ronen Basri, Hilton Bristow, Nachum Dershowitz, Alexandra Gilinsky, Sing Bing Kang, Kevin Karsch, Iasonas Kokkinos, Simon Lucey, Viki Mayzels, Francesc Moreno-Noguer, Weichao Qiu, Miki Rubinstein, Gil Sadeh, Alberto Sanfeliu, Daniel Stökl Ben-Ezra, Antonio Torralba, Eduard Trulls, Zhiwen Tu, Xinggang Wang, Lior Wolf, Jenny Yuen, Alan Yuille and Lihi Zelnik-Manor.

Tal Hassner thanks the Open University of Israel (OUI) and, in particular, the chief of its research authority, Daphna Idelson, for their generous support for this book. He also gratefully acknowledges the longtime guidance, support and friendship of Ronen Basri, Michal Irani, Lior Wolf and Lihi Zelnik-Manor.

Ce Liu thanks Harry Shum, Rick Szeliski, Bill Freeman, Ted Adelson, Antonio Torralba and Yair Weiss for their advice and support in his life. He is grateful to his collaborators, Antonio Torralba, Jenny Yuen, Miki Rubinstein, Marshall Tappen, Kevin Karsch, Jaechul Kim and Philip Isola, on the topic of dense correspondences.

Raanana, Israel
Cambridge, MA, USA

Tal Hassner
Ce Liu

Contents

Part I Establishing Dense Correspondences

Introduction to Dense Optical Flow	3
Ce Liu	
SIFT Flow: Dense Correspondence Across Scenes and Its Applications ..	15
Ce Liu, Jenny Yuen, and Antonio Torralba	
Dense, Scale-Less Descriptors	51
Tal Hassner, Viki Mayzels, and Lihi Zelnik-Manor	
Scale-Space SIFT Flow	71
Weichao Qiu, Xinggang Wang, Xiang Bai, Alan Yuille, and Zhuowen Tu	
Dense Segmentation-Aware Descriptors	83
Eduard Trulls, Iasonas Kokkinos, Alberto Sanfeliu, and Francesc Moreno-Noguer	
SIFTpack: A Compact Representation for Efficient SIFT Matching	109
Alexandra Gilinsky and Lihi Zelnik-Manor	
In Defense of Gradient-Based Alignment on Densely Sampled Sparse Features	135
Hilton Bristow and Simon Lucey	

Part II Dense Correspondences and Their Applications

From Images to Depths and Back	155
Tal Hassner and Ronen Basri	
Depth Transfer: Depth Extraction from Videos Using Nonparametric Sampling	173
Kevin Karsch, Ce Liu, and Sing Bing Kang	

Nonparametric Scene Parsing via Label Transfer.....	207
Ce Liu, Jenny Yuen, and Antonio Torralba	
Joint Inference in Weakly-Annotated Image Datasets via Dense Correspondence	237
Michael Rubinstein, Ce Liu, and William T. Freeman	
Dense Correspondences and Ancient Texts	279
Tal Hassner, Lior Wolf, Nachum Dershowitz, Gil Sadeh, and Daniel Stökl Ben-Ezra	

Part I

Establishing Dense Correspondences

Introduction to Dense Optical Flow

Ce Liu

Abstract Before the notion of motion is generalized to arbitrary images, we first give a brief introduction to motion analysis for videos. We will review how motion is estimated when the underlying motion is *slow* and *smooth*, especially the Horn–Schunck (*Artif Intell* 17:185–203, 1981) formulation with robust functions. We show step-by-step how to optimize the optical flow objective function using iteratively reweighted least squares (IRLS), which is equivalent to conventional Euler–Lagrange variational approach but more succinct to derive. Then we will briefly discuss how motion is estimated when the slow and smooth assumption becomes invalid, especially how large displacement motion is estimated.

1 Introduction

Motion estimation is one of the corner stones of computer vision. It is widely used in video processing to compress videos and to enhance video qualities, and also used in 3D reconstruction, object/event tracking, segmentation, and recognition.

Although video cameras are able to record pixels of the moving objects, motion is unfortunately not recorded directly. Although the amount of motion can be physically measured at very high accuracy in a lab setup, motion remains as a percept instead of direct measurement for general videos. The challenge of motion estimation is therefore to obtain motion that is consistent with human perception.

Although multiple representations of motion have been invented, the most popular ones are *parametric motion* such as affine and homography (projective) where the displacement of pixels undergoes certain parametric forms, or *optical flow fields*, where every pixel has its own displacement vector. These two representations mainly differ in how the motion fields are regularized across the image lattice, while the optimization and initialization strategies are almost the same. Therefore, in this chapter we focus on optical flow estimation. For parametric motion estimation, please refer to [1].

C. Liu (✉)
Google Research, Cambridge, MA, USA
e-mail: celiu@google.com

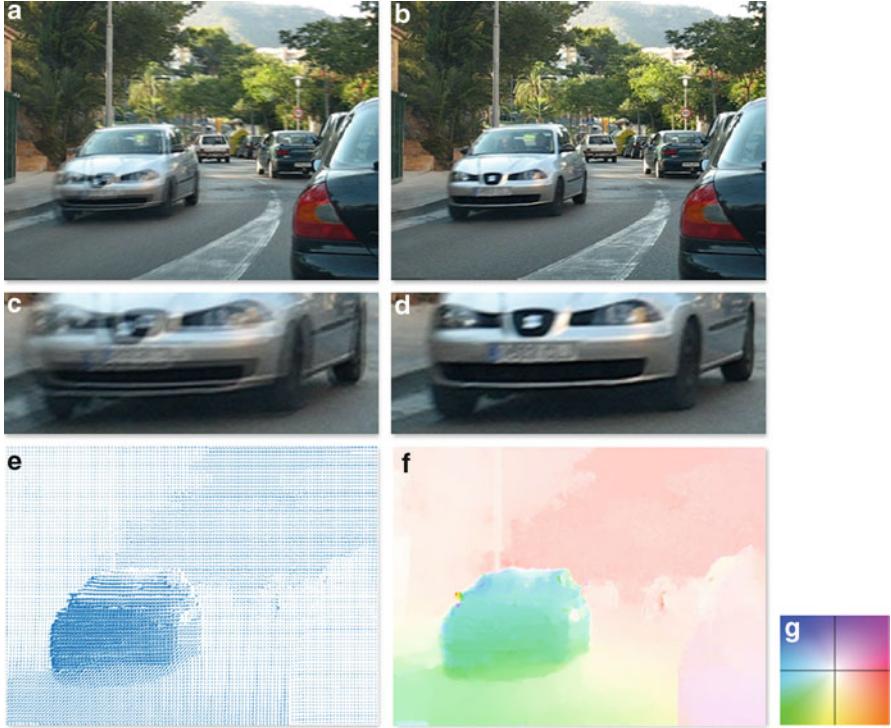


Fig. 1 Illustration of optical flow fields. (a) Superimposition of two input frames. (b) Superimposition of frame 1 and warped frame 2 according to the estimated flow field. (c) and (d) are the zoomed-in versions of (a) and (b), respectively. Notice the double imaging in (a) and (c) due to the motion between the two frames and the sharp boundaries in (b) and (d) as the motion is canceled via warping. This is a common way to inspect motion on printed paper, while flipping back and forth two frames is a better way to inspect motion on a digital display. (e) and (f) are two different ways to visualize the flow fields. In (e) flow fields are plotted as vectors on sparse grid. Consequently, it is challenging to visualize a dense flow field with large magnitude. In (f) flow fields are visualized via a color-coding scheme in (g) [2], where *hue* indicates orientation and *saturation* indicates magnitude. It has become the standard of optical flow visualization since it is possible to see the motion of every pixel. The two frames are from the MIT ground-truth motion database [15]

The effect of motion is illustrated in Fig. 1. For two adjacent frames in a video sequence, the correct flow field should be able to cancel the underlying motion such that the second frame should be identical to the first frame after being warped according to the flow field. In addition, the discontinuity of the flow field should reflect the boundary of the objects in the scene. These are general guidelines to inspect the correctness of the estimated flow field when the ground-truth flow field is absent.

In this chapter we will introduce the optical flow formulation from the basic brightness constancy and smoothness assumption and derive the optical flow estimation algorithm via *incrementalization* and *linearization* (Taylor expansion) of

the objective function. The optimization is derived using iteratively reweighted least squares (IRLS), which is equivalent to the conventional Euler–Lagrange variational approach but is more succinct to derive. Initialization is a key component of optical flow estimation that has been often overlooked. We will discuss about the conventional coarse-to-fine scheme and recent advances using feature matching for initialization to account for fast moving scenes.

For the readers who are interested in knowing more about the literature of optical flow estimation, there are a number of references. An early survey and evaluation of optical flow algorithms can be found in [4]. Various forms of motion estimation are discussed in Rick Szeliski’s book [20], while details of optical flow optimization strategies are discussed in Ce Liu’s PhD Thesis [14].

2 Slow and Smooth Optical Flow

The motion of the moving scenes both in the nature and in the man-made world takes drastically different forms. Most of the things in our surroundings do not move typically, such as the land, buildings, walls, and desks. The motion of vehicles can be regarded as moving planes from side view except for the wheels. Animals and humans’ motion is more complicated with degrees of articulation. The motion of water, fire, and explosion can be so complicated that it is beyond human comprehension. To make the motion estimation problem tractable, a common assumption is that motion is *slow* and *smooth*, namely the pixels tend to stay still or move at low speed, and neighbor pixels tend to move together.

2.1 Basic Formulation

Let image lattice be $p = (x, y) \in \Lambda$, and the two images to match be $I_1(p)$ and $I_2(p)$. Denote $w_p = (u_p, v_p)$ the flow vector at pixel p , where u_p and v_p are the horizontal and vertical components of the flow fields, respectively.

As motion before, the optical flow field should be able to align the two images along with the flow field. The objective function of optical flow is [12]

$$E(w) = \sum_p \psi(|I_1(p) - I_2(p + w_p)|^2) + \lambda \sum_p \phi(|\nabla u_p|^2 + |\nabla v_p|^2), \quad (1)$$

where the first term is often called the *data term* and the second term is called the *smoothness term*. λ is a coefficient that balances the two terms. In this equation, $\psi(\cdot)$ and $\phi(\cdot)$ are both robust functions [6], which can take following forms:

- L2 norm: $\psi(z^2) = z^2$
- L1 norm: $\psi(z^2) = \sqrt{z^2 + \varepsilon^2}$
- Lorentzian: $\psi(z^2) = \log(1 + \gamma z^2)$

The same functions can be chosen for the smoothness robust function ϕ as well. Both L1 and Lorentzian forms make the function robust, namely to be able to account for matching outliers in the data term and to encourage piecewise smooth discontinuities. This is often called L1 total variation optical flow [9].

It is worth noting that the two images I_1 and I_2 are not limited to grayscale images. If they can be multiple-channel images (such as RGB, HSV, and YUV), an extra summation over the channels is needed in the data term. The images may also contain some image features such as gradients and other features from linear or nonlinear filtering. When gradients are used, then gradient constancy is implied, which can make the flow more stable at the presence of global illumination change.

This objective function in Eq. (1) is very difficult to minimize because of the warping function $I_2(p + w_p)$. To deal with warping, we follow the typical incrementation and linearization strategy. First, the objective function can be rewritten to optimize over incremental $dw = (du, dv)$ of the flow field:

$$\begin{aligned} E(w, dw) = & \sum_p \psi(|I_1(p) - I_2(p + w_p + dw_p)|^2) \\ & + \lambda \sum_p \phi(|\nabla(u_p + du_p)|^2 + |\nabla(v_p + dv_p)|^2). \end{aligned} \quad (2)$$

Second, we can linearize the warping using Taylor expansion

$$I_2(p + w_p + dw_p) - I_1(p) \approx I_t(p) + I_x(p)du_p + I_y(p)dv_p, \quad (3)$$

where

$$I_t(p) = I_2(p + w_p) - I_1(p), \quad (4)$$

$$I_x(p) = \frac{\partial}{\partial x} I_2(p + w_p), \quad (5)$$

$$I_y(p) = \frac{\partial}{\partial y} I_2(p + w_p). \quad (6)$$

Now the objective function becomes

$$\begin{aligned} E(w, du, dv) = & \sum_p \psi(|I_t(p) + I_x(p)du_p + I_y(p)dv_p|^2) \\ & + \lambda \sum_p \phi(|\nabla(u + du)_p|^2 + |\nabla(v + dv)_p|^2). \end{aligned} \quad (7)$$

Our goal is to rewrite Eq. (7) in vector and matrix forms. We define the robust function Ψ and Φ as applying the robust function to each element of a vector:

$$\Psi(X) = [\psi(X_1), \psi(X_2), \dots, \psi(X_n)]^T, \quad (8)$$

$$\Phi(X) = [\phi(X_1), \phi(X_2), \dots, \phi(X_n)]^T. \quad (9)$$

To be succinct, we denote XY and X^2 as element-wise multiplication and element-wise square for vectors

$$XY = [X_1 Y_1, X_2 Y_2, \dots, X_n Y_n]^T, \quad (10)$$

$$X^2 = [X_1^2, X_2^2, \dots, X_n^2]^T. \quad (11)$$

In this way Eq. (7) can be rewritten in a vector form

$$\begin{aligned} E(w, du, dv) &= \mathbf{1}^T \Psi \left((I_t + I_x du + I_y dv)^2 \right) \\ &\quad + \lambda \mathbf{1}^T \Phi \left((\mathbf{D}_x(u + du))^2 + (\mathbf{D}_y(u + du))^2 \right. \\ &\quad \left. + (\mathbf{D}_x(v + dv))^2 + (\mathbf{D}_y(v + dv))^2 \right), \end{aligned} \quad (12)$$

where \mathbf{D}_x and \mathbf{D}_y are matrices corresponding to x - and y -derivative filters, such as $[-1, 1]$ filters.

2.2 Optimization via Iteratively Reweighted Least Squares

The typical approach that can be found in the optical flow literature is to use Euler–Lagrange to find fixed point of the partial differential equations (PDEs). We are going to show how to derive using Euler–Lagrange in the next subsection. Here, we are going to use IRLS to directly optimize the objective function in Eq. (7).

To simplify the notations, denote

$$\begin{aligned} \Psi' &= \Psi' \left((I_t + I_x du + I_y dv)^2 \right), \\ \Psi'_{xx} &= \text{diag}(\Psi' I_x I_x), \quad \Psi'_{xy} = \text{diag}(\Psi' I_x I_y), \quad \Psi'_{yy} = \text{diag}(\Psi' I_y I_y), \\ \Psi'_{xt} &= \text{diag}(\Psi' I_x I_t), \quad \Psi'_{yt} = \text{diag}(\Psi' I_y I_t), \\ \Phi' &= \text{diag} \left(\Phi \left((\mathbf{D}_x(u + du))^2 + (\mathbf{D}_y(u + du))^2 \right. \right. \\ &\quad \left. \left. + (\mathbf{D}_x(v + dv))^2 + (\mathbf{D}_y(v + dv))^2 \right) \right), \\ \mathbf{L} &= \mathbf{D}_x^T \Phi' \mathbf{D}_x + \mathbf{D}_y^T \Phi' \mathbf{D}_y. \end{aligned} \quad (13)$$

Taking the derivative of the objective function w.r.t. du and setting it to be zero, we obtain

$$\begin{aligned}
\frac{\partial}{\partial du} E(w, du, dv) &= 2\Psi'(I_t + I_x du + I_y dv)I_x \\
&\quad + 2\lambda(\mathbf{D}_x^T \mathbf{D}_x du + \mathbf{D}_x^T \mathbf{D}_x u + \mathbf{D}_y^T \mathbf{D}_y du + \mathbf{D}_y^T \mathbf{D}_y u) \\
&= 0.
\end{aligned} \tag{14}$$

We can get similar result for partial derivative w.r.t. dv . Using the aforementioned shortcuts in Eq. (13) we obtain the following equation:

$$\begin{bmatrix} \Psi'_{xx} + \lambda \mathbf{L} & \Psi'_{xy} \\ \Psi'_{xy} & \Psi'_{yy} + \lambda \mathbf{L} \end{bmatrix} \begin{bmatrix} du \\ dv \end{bmatrix} = - \begin{bmatrix} \Psi'_{xt} + \lambda \mathbf{L} u \\ \Psi'_{yt} + \lambda \mathbf{L} v \end{bmatrix}. \tag{15}$$

To solve this linear equation we need to know Ψ' and Φ' , which depend on the current estimate of du and dv . Naturally, we can have an iterative algorithm that alternates between estimating the weighting matrices in Eq. (13) and solving the linear system in Eq. (15). This method is called IRLS that is widely used in nonlinear image reconstruction such as Levin and Weiss [13].

2.3 Variational Approach via Euler–Lagrange

Notice that a more popular way to derive optical flow optimization is Euler–Lagrange [9, 12], which has continuous formulation instead of discrete (again $w = (u, v)$, $dw = (du, dv)$)

$$\int_p \psi(|I_1(p) - I_2(p + w_p + dw_p)|^2) + \lambda \phi(|\nabla(u_p + du_p)|^2 + |\nabla(v_p + dv_p)|^2) dp. \tag{16}$$

The 2D Euler–Lagrange theorem states that the functional

$$S = \int \int_{\Omega} L(x, y, f, f_x, f_y) dx dy \tag{17}$$

is minimized only if f satisfies the PDE

$$\frac{\partial L}{\partial f} - \frac{\partial}{\partial x} \frac{\partial L}{\partial f_x} - \frac{\partial}{\partial y} \frac{\partial L}{\partial f_y} = 0. \tag{18}$$

Consider the simple case where $\psi(z^2) = z^2$ and $\phi(z^2) = z^2$, which implies no robust function is imposed. This is exactly the same formulation as Horn and Schunck [12]. In this formulation, we have the following observations (omitting subscript p to be succinct):

$$\begin{aligned}
L(du, dv, du_x, du_y, dv_x, dv_y) &= (I_x du + I_y dv + I_t)^2 + \lambda((u_x + du_x)^2 \\
&\quad + (u_y + du_y)^2 + (v_x + dv_x)^2 + (v_y + dv_y)^2), \\
\frac{\partial L}{\partial du} &= 2(I_x du + I_y dv + I_t)I_x, \\
\frac{\partial L}{\partial du_x} &= 2\lambda(u_x + du_x), \frac{\partial}{\partial x} \frac{\partial L}{\partial du_x} = 2\lambda(u_{xx} + du_{xx}), \\
\frac{\partial L}{\partial du_y} &= 2\lambda(u_y + du_y), \frac{\partial}{\partial y} \frac{\partial L}{\partial du_y} = 2\lambda(u_{yy} + du_{yy}).
\end{aligned}$$

Using the Euler–Lagrange in Eq. (18) we obtain

$$\begin{cases} (I_x du + I_y dv + I_t)I_x - \lambda(u_{xx} + du_{xx} + u_{yy} + du_{yy}) = 0, \\ (I_x du + I_y dv + I_t)I_y - \lambda(v_{xx} + dv_{xx} + v_{yy} + dv_{yy}) = 0. \end{cases} \quad (19)$$

Notice that $u_{xx} + u_{yy}$ can be obtained by a Laplacian operator:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}. \quad (20)$$

Denote by \mathbf{L} the filtering matrix corresponding to Laplacian operator in the above equation, which can be expressed as

$$\mathbf{L} = \mathbf{D}_x^T \mathbf{D}_x + \mathbf{D}_y^T \mathbf{D}_y, \quad (21)$$

where the derivative filter in \mathbf{D}_x and \mathbf{D}_y has to be $[-1, 1]$. After discretizing Eq. (19), we obtain

$$\begin{bmatrix} \mathbf{I}_x^2 + \lambda \mathbf{L} & \mathbf{I}_x \mathbf{I}_y \\ \mathbf{I}_x \mathbf{I}_y & \mathbf{I}_y^2 + \lambda \mathbf{L} \end{bmatrix} \begin{bmatrix} du \\ dv \end{bmatrix} = - \begin{bmatrix} \mathbf{I}_x I_t + \lambda \mathbf{L} u \\ \mathbf{I}_y I_t + \lambda \mathbf{L} v \end{bmatrix}, \quad (22)$$

where $\mathbf{I}_x = \text{diag}(I_x)$ and $\mathbf{I}_y = \text{diag}(I_y)$ are diagonal matrices where the diagonal are the x - and y -derivatives. If we drop all the robust functions, we can see that Eq. (15) is exactly the same as Eq. (22). The same conclusion holds with robust functions imposed as well [14]. However, with robust functions, fixed-point iterations have to be derived under Euler–Lagrange, which can be quite cumbersome. It is more succinct to derive optical flow equation using IRLS after all.

Many solvers can be used for the sparse linear systems in Eq. (15), such as conjugate gradient [18] and Gauss–Seidel [10]. Notice that this linear system is positive semi-definite by definition. The condition number of the matrix can be improved by adding a small constant to the diagonal, which is equivalent to explicitly penalizing the magnitude of the increments du and dv .

2.4 Coarse-to-Fine Initialization

The basic assumption for linearization, namely Taylor expansion in Eq. (3), is that the two images $I_1(p)$ and $I_2(p + w_p)$ are close enough, which implies that the underlying motion has to be small. But motion may not be small in real-life videos. One popular solution to handle larger motion is to estimate optical flow progressively in a coarse-to-fine scheme as illustrated in Fig. 2. Image pyramids are built for both images. At the top pyramid level, the flow is initialized as zero motion. If the top pyramid level is η times smaller than the bottom level, then the motion should also be η times smaller. For instance, for a Gaussian pyramid with 5 levels and the downsampling rate of 0.5, flow vectors at the top level are 1/16 of those at the bottom level. Therefore, this zero motion assumption reasonably holds for slow and smooth moving sequences. We will come back to this topic in the next section dealing with large-displacement optical flow.

After the flow estimation is performed at a coarser image pyramid level, the estimated flow field is upsampled and scaled to initialize the flow estimation in the finer pyramid level. In [9], the downsampling rate of the pyramid is chosen to be close to 1, for instance 0.9, which means almost 6.5 pyramid levels are in-between a regular pyramid with downsampling rate 0.5. This gives benefit of high accuracy flow at the cost of high computation.



Fig. 2 Illustration of the coarse-to-fine scheme of optical flow estimation. *Left:* the mean of input two frames (at various pyramid level); *right:* the estimated flow field at each pyramid level

2.5 Classical Optical Flow

So far we have introduced the basic formulations and techniques in optical flow. There are many variants to the classical optical flow formulation in Eq. (7), as well as alternative optimization approaches. For example, in addition to brightness constancy, gradient constancy can be explicitly formulated as well [9] to deal with lighting changes. Graduated non-convexity (GNC) [7] was introduced so that a quadratic version of the non-convex objective function is first optimized and then the objective function is gradually transitioned to the original non-convex function.

In [19], many such variants were thoroughly tested via extensive experiments. The authors discovered that the single most important trick to do in flow estimation is to perform median filtering in a 5×5 window to the estimated flow field. To be succinct, we will just show the original objective function before incrementation from this point on. As a result, all the dw , du , and dv terms will be dropped in the equations. In addition, we will just use the L1 norm as the robust function for both data and smoothness terms. The optical flow estimation that corresponds to median filtering becomes

$$\begin{aligned} E_A(w, \hat{w}) = & \sum_p |I_1(p) - I_2(p + w_p)| + \lambda |\nabla u_p| + \lambda |\nabla v_p| \\ & \sum_p \lambda_2 |w_p - \hat{w}_p| + \sum_p \sum_{(p,q) \in N_p} \lambda_3 |\hat{w}_p - \hat{w}_q|, \end{aligned} \quad (23)$$

where N_p is a (5×5) neighborhood at pixel p . Optical flow is estimated by alternating optimizing w and \hat{w} . The last term in this objective function on \hat{w} is called a nonlocal term, which imposes a particular smoothness assumption within a specified region of the auxiliary flow field $\hat{w} = (\hat{u}, \hat{v})$.

Furthermore, we can assume that there is some sort of correlation between optical flow boundaries and image boundaries—although image boundaries do not necessarily indicate motion boundaries, motion boundaries often lie on image boundaries. Using this cue, an improved model is

$$\begin{aligned} E_A(w, \hat{w}) = & \sum_p |I_1(p) - I_2(p + w_p)| + \lambda |\nabla u_p| + \lambda |\nabla v_p| \\ & \sum_p \lambda_2 |w_p - \hat{w}_p| + \sum_p \sum_{(p,q) \in N_p} \eta_{pq} |\hat{w}_p - w_q|, \end{aligned} \quad (24)$$

where η_{pq} is a bilateral weight,

$$\eta_{pq} \propto \exp \left\{ -\frac{|p - q|^2}{2\sigma_s^2} - \frac{|I(p) - I(q)|^2}{2\sigma_I^2} \right\} \frac{o(q)}{o(p)}, \quad (25)$$

and the occlusion variable $o(p)$ and $o(q)$ are calculated in [17]. The algorithm based on Eq. (24), called *Classic-NL*, performs very well on a number of optical flow benchmark data set such as Middlebury optical flow data set [2].

3 Large-Displacement Optical Flow

While the coarse-to-fine optical flow algorithm is able to handle large motion to some degree, it comes with the cost that small objects and/or thin structures that move differently from larger regions in the video tend to overwhelmed during the pyramid construction process so that they become invisible at top levels. As a result, the flow estimation at top levels of the pyramid tends to only capture the motion of larger regions while ignoring the motion of small objects. Certainly these scenarios violate the basic assumption of *slow* and *smooth* in the existing optical flow estimation framework, but such scenarios appear quite often not only in some benchmark databases such as MPI-Sintel but also in daily lives such as fast moving balls in sports. Fast motion may also come from dramatic camera view change, fast motion of objects, and under temporal sampling. These are all challenging scenarios to existing flow algorithms designed based on the *slow* and *smooth* assumption.

On the other hand, sparse features and descriptors such as SIFT [16], HOG [11], and SURF [5] are very liable for building correspondences between two images undergoing drastic view changes or fast motion. However, the representation is sparse and does not cover the entire image lattice.

In [8], the authors proposed to integrate feature point matching into optical flow estimation. With an auxiliary flow field \hat{w} , the objective function of optical flow is modified to

$$\begin{aligned}
 E_A(w, \hat{w}) = & \sum_p \|I_1(p) - I_2(p + w_p)\| \\
 & + \gamma \sum_p \|\nabla I_1(p) - \nabla I_2(p + w_p)\| \\
 & + \alpha \sum_p \|\nabla u_p\| + \|\nabla v_p\| \\
 & + \beta \sum_p \delta(p) \rho(p) \Psi(|w_p - \hat{w}_p|^2) \\
 & + \sum_p \delta(p) \|f_1(p) - f_2(p + \hat{w}_p)\|^2. \tag{26}
 \end{aligned}$$

In this formulation, the first and second terms are brightness and gradient constancy. The third term is (piece-wise) smoothness of the flow field. The fourth and fifth terms are new here. The fifth term is descriptor matching on a sparse grid $\delta(p)$

where only a few pixels are 1s. The fourth term is the correlation between the flow field w and the sparse matching \hat{w} on the sparse grid $\delta(p)$ modulated by weight $\rho(p)$, which reflects the matching quality. This formulation is called large displacement optical flow (LDOF) [8], where the sparse matching is implemented by uniformly sampling the image grid and extracting HOG features.

Although the flow field w is still estimated in a coarse-to-fine scheme in Eq. (26), the sparse matching \hat{w} is estimated in advance so the initialization at the top pyramid also reflects sparse matching. LDOF is able to achieve much better results than conventional optical flow on videos containing fast moving small objects.

Motion details preserving optical flow (MDPOF) [21] attempted to extend the LDOF framework. The HOG features were uniformly sampled on the image lattice in LDOF, but advanced features such as SIFT [16], fast dense matching algorithms such as PatchMatch [3], and the conventional classical optical flow can generate reliable flow proposals as well. Optical flow becomes a discrete labeling problem for every pixel to choose the best flow vector from a set of flow fields. As more flow proposals were considered, MDPOF outperforms LDOF on the standard benchmark data sets.

4 Conclusion

The fundamental assumption of the optical flow estimation is *slow* and *smooth* motion. As a result, Taylor expansion is valid to linearize the nonlinear brightness and/or gradient constancy to allow for an iterative algorithm to find the flow field so that the second frame can be successfully warped to the first frame. The initialization of the flow estimation can be a zero flow field at the top level of a Gaussian pyramid and the estimation is progressively performed from top to the bottom level of the pyramid. When there are small objects that move differently from the larger regions, LDOF is designed to incorporate feature matching to better initialize/constrain flow estimation.

However, the condition for the Taylor expansion to hold may not exist in scenarios when the dramatic appearance changes take place between two frames to be registered/aligned. As the brightness constancy is no longer valid, we need to seek for other sort of constancy between two images to establish correspondences.

References

1. Baker, S., Matthews, I.: Lucas-kanade 20 years on: a unifying framework. *Int. J. Comput. Vis.* **56**(3), 221–255 (2004)
2. Baker, S., Scharstein, D., Lewis, J.P., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. *Int. J. Comput. Vis.* **92**, 1–31 (2010)
3. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM Trans. Graph. (Proc. SIGGRAPH)* **28**(3) (2009)

4. Barron, J.L., Fleet, D.J., Beauchemin, S.S.: Performance of optical flow techniques. *Int. J. Comput. Vis.* **12**, 43–77 (1994)
5. Bay, H., Tuytelaars, T., Gool, L.V.: SURF: speeded up robust features. In: European Conference on Computer Vision (ECCV) (2006)
6. Black, M.J., Anandan, P.: The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Comput. Vis. Image Underst.* **63**(1), 75–104 (1996)
7. Blake, A., Zisserman, A.: Visual Reconstruction. MIT Press, Cambridge (1987)
8. Brox, T., Malik, J.: Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(3), 500–513 (2011)
9. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: European Conference on Computer Vision (ECCV), pp. 25–36 (2004)
10. Bruhn, A., Weickert, J., Schnörr, C.: Lucas/Kanade meets Horn/Schunk: combining local and global opti flow methods. *Int. J. Comput. Vis.* **61**(3), 211–231 (2005)
11. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005)
12. Horn, B.K.P., Schunck, B.G.: Determining optical flow. *Artif. Intell.* **17**, 185–203 (1981)
13. Levin, A., Weiss, Y.: User assisted separation of reflections from a single image using a sparsity prior. *IEEE Trans. Pattern Anal. Mach. Intell.* **1**(9), 72–91 (2007)
14. Liu, C.: Beyond pixels: exploring new representations and applications for motion analysis. Ph.D. thesis, Massachusetts Institute of Technology (2009)
15. Liu, C., Freeman, W.T., Adelson, E.H., Weiss, Y.: Human-assisted motion annotation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2008)
16. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **60**(2), 91–110 (2004)
17. Sand, P., Teller, S.: Particle video: long-range motion estimation using point trajectories. *Int. J. Comput. Vis.* **1**(80), 72–91 (2008)
18. Strang, G.: Introduction to Applied Mathematics. Wellesley-Cambridge Press, Wellesley (1986)
19. Sun, D., Roth, S., Black, M.J.: Secrets of optical flow estimation and their principles. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010)
20. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer, Berlin (2010)
21. Xu, L., Jia, J., Matsushita, Y.: Motion detail preserving optical flow estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* **34**(9), 1744–1757 (2012)

SIFT Flow: Dense Correspondence Across Scenes and Its Applications

Ce Liu, Jenny Yuen, and Antonio Torralba

Abstract While image alignment has been studied in different areas of computer vision for decades, aligning images depicting different scenes remains a challenging problem. Analogous to optical flow where an image is aligned to its temporally adjacent frame, we propose scale-invariant feature transform (*SIFT*) *flow*, a method to align an image to its nearest neighbors in a large image corpus containing a variety of scenes. The SIFT flow algorithm consists of matching densely sampled, pixel-wise SIFT features between two images while preserving spatial discontinuities. The SIFT features allow robust matching across different scene/object appearances, whereas the discontinuity-preserving spatial model allows matching of objects located at different parts of the scene. Experiments show that the proposed approach robustly aligns complex scene pairs containing significant spatial differences. Based on SIFT flow, we propose an alignment-based large database framework for image analysis and synthesis, where image information is transferred from the nearest neighbors to a query image according to the dense scene correspondence. This framework is demonstrated through concrete applications, such as motion field prediction from a single image, motion synthesis via object transfer, satellite image registration, and face recognition.

1 Introduction

Image alignment, registration, and correspondence are central topics in computer vision. There are several levels of scenarios in which image alignment dwells. The simplest level, aligning different views of the same scene, has been studied for the

C. Liu (✉)
Google Research, Cambridge, MA, USA
e-mail: celiu@google.com

J. Yuen
Facebook, Menlo Park, CA, USA
e-mail: jenny@csail.mit.edu

A. Torralba
Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: torralba@csail.mit.edu

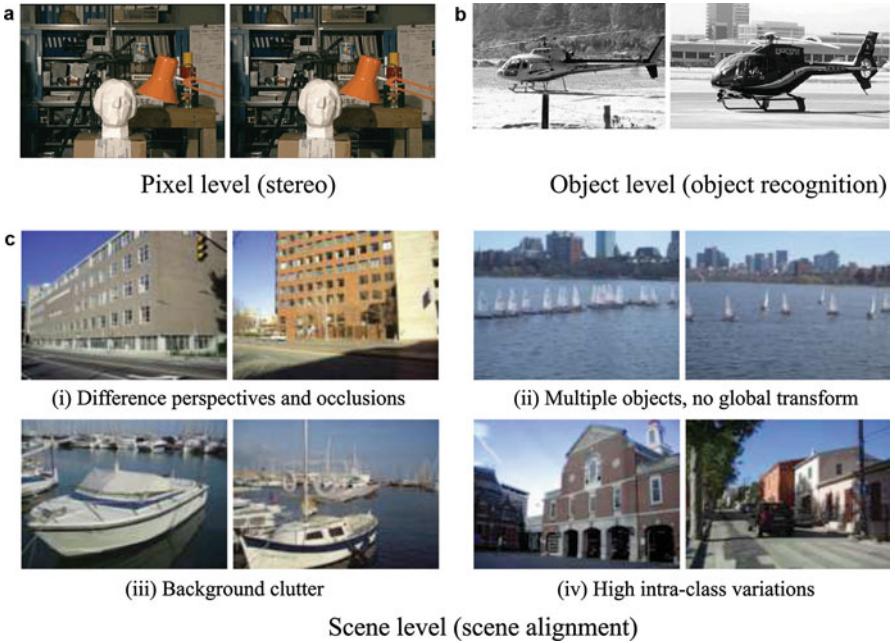


Fig. 1 Image alignment resides at different levels. Researchers used to study image alignment problems at the pixel level, where two images are captured from the same scene with slightly different time or at different perspectives [45] (a). Recently, correspondence has been extended to the object level (b) for object recognition [6]. We are interested in image alignment at the scene level, where two images come from different 3D scenes, but share similar scene characteristics (c). SIFT flow is proposed to align the examples in (c) for scene alignment

purpose of image stitching [51] and stereo matching [45], e.g., in Fig. 1a. The considered transformations are relatively simple (e.g., parametric motion for image stitching and 1D disparity for stereo), and images to register are typically assumed to have the same pixel value after applying the geometric transformation.

The image alignment problem becomes more complicated for dynamic scenes in video sequences, e.g., optical flow estimation [12, 29, 38]. The correspondence between two adjacent frames in a video is often formulated as an estimation of a 2D flow field. The extra degree of freedom transitioning from 1D in stereo to 2D in optical flow introduces an additional level of complexity. Typical assumptions in optical flow algorithms include brightness constancy and piecewise smoothness of the pixel displacement field [3, 8].

Image alignment becomes even more difficult in the object recognition scenario, where the goal is to align different instances of the same object category, as illustrated in Fig. 1b. Sophisticated object representations [4, 6, 19, 57] have been developed to cope with the variations of object shapes and appearances. However, these methods still typically require objects to be salient, similar, with limited background clutter.

In this work, we are interested in a new, higher level of image alignment: aligning two images from different 3D scenes but sharing similar scene characteristics. Image alignment at the scene level is thus called *scene alignment*. As illustrated in Fig. 1c, the two images to match may contain object instances captured from different viewpoints, placed at different spatial locations, or imaged at different scales. The two images may also contain different quantities of objects of the same category, and some objects present in one image might be missing in the other. Due to these issues the scene alignment problem is extremely challenging.

Ideally, in scene alignment we want to build correspondence at the semantic level, i.e., matching at the object class level, such as buildings, windows, and sky. However, current object detection and recognition techniques are not robust enough to detect and recognize all objects in images. Therefore, we take a different approach for scene alignment by matching local, salient, and transform-invariant image structures. We hope that semantically meaningful correspondences can be established through matching these image structures. Moreover, we want to have a simple, effective, object-free model to align image pairs such as the ones in Fig. 1c.

Inspired by optical flow methods, which are able to produce dense, pixel-to-pixel correspondences between two images, we propose scale-invariant feature transform (*SIFT*) flow, adopting the computational framework of optical flow, but by matching SIFT descriptors instead of raw pixels. In SIFT flow, a SIFT descriptor [37] is extracted at each pixel to characterize local image structures and encode contextual information. A discrete, discontinuity-preserving, flow estimation algorithm is used to match the SIFT descriptors between two images. The use of SIFT features allows robust matching across different scene/object appearances and the discontinuity-preserving spatial model allows matching of objects located at different parts of the scene. Moreover, a coarse-to-fine matching scheme is designed to significantly accelerate the flow estimation process.

Optical flow is only applied between two adjacent frames in a video sequence in order to obtain meaningful correspondences; likewise, we need to define the *neighborhood* for SIFT flow. Motivated by the recent progress in large image database methods [28, 43], we define the neighbors of SIFT flow as the top matches retrieved from a large database. The chance that some of the nearest neighbors share the same scene characteristics with a query image increases as the database grows, and the correspondence obtained by SIFT flow can be semantically meaningful.

Using SIFT flow, we propose an alignment-based large database framework for image analysis and synthesis. The information to infer for a query image is transferred from the nearest neighbors in a large database to this query image according to the dense scene correspondence estimated by SIFT flow. Under this framework, we apply SIFT flow to two novel applications: *motion prediction from a single static image*, where a motion field is hallucinated from a large database of videos, and *motion transfer*, where a still image is animated using object motions transferred from a similar moving scene. We also apply SIFT flow back to the regime of traditional image alignment, such as satellite image registration and face recognition. Through these examples we demonstrate the potential of SIFT flow for broad applications in computer vision and computer graphics.

The rest of the chapter is organized as follows: after reviewing the related work in Sect. 2, we introduce the concept of SIFT flow and the inference algorithm in Sect. 3. In Sect. 4, we show scene alignment examples using SIFT flow with evaluations. In Sect. 5, we show how to infer the motion field from a single image and how to animate a still image, both with the support of a large video database and scene alignment. We further apply SIFT flow to satellite image registration and face recognition in Sect. 6. After briefly discussing how SIFT flow fits in the literature of image alignment in Sect. 7, we conclude the chapter in Sect. 8.

2 Related Work

Image alignment, a.k.a. image registration or correspondence, is a broad topic in computer vision, computer graphics, and medical imaging, covering stereo, motion analysis, video compression, shape registration, and object recognition. It is beyond the scope of this chapter to give a thorough review on image alignment. Please refer to [51] for a comprehensive review on this topic. In this section, we will review the image alignment literature focusing on

- (a) *What* to align, or the features that are consistent across images, e.g., pixels, edges, descriptors;
- (b) *Which* way to align, or the representation of the alignment, e.g., sparse vs. dense, parametric vs. nonparametric;
- (c) *How* to align, or the computational methods to obtain alignment parameters.

In addition, correspondence can be established between two images or between an image and image models such as in [15]. We will focus on the correspondence between two images.

In image alignment we must first define the features based on which image correspondence will be established: an image measurement that does not change from one image to another. In stereo [26] and optical flow [29, 38], the brightness constancy assumption was often made for building the correspondence between two images. But soon researchers came to realize that pixel values are not reliable for image matching due to changes of lighting, perspective, and noise [25]. Features such as phase [21], filter banks [30], mutual information [54], and gradient [10] are used to match images since they are more reliable than pixel values across frames, but they still fail to deal with drastic changes. Middle level representations such as SIFT [37], shape context [5, 6], and histogram of oriented gradients (HOG) [17] have been introduced to account for stronger appearance changes and are proven to be effective in a variety of applications such as visual tracking [1], optical flow estimation [11], and object recognition [37]. Nevertheless, little has been investigated for exploring features to establish correspondences at the scene level.

The representation of the correspondence is another important aspect of image alignment. One can utilize the information of every pixel to obtain a dense

correspondence, or merely use sparse feature points. The form of the correspondence can be pixel-wise displacement such as a 1D disparity map (stereo) and a 2D flow field (optical flow), or parametric models such as affine and homography. Although a parametric model can be estimated from matching every pixel [7] and a dense correspondence can be interpolated from sparse matching [56], typically, pixel-wise displacement is obtained through pixel-wise correspondence, and parametric motion is estimated from sparse, interest point detection, and matching [46]. In between the sparse and dense representation is correspondence on contours [33, 55], which has been used in tracking objects and analyzing motion for textureless objects. The fact that the underlying correspondence between scenes is complicated and unclear, and detecting contours from scenes can be unreliable, leads us to seek for dense, pixel-wise correspondence for scene alignment.

Estimating dense correspondence between two images is a nontrivial problem with spatial regularity, i.e., the displacements (flow vectors) of neighboring pixels tend to be similar. When the feature values of the two images are close and temporally smooth, this displacement can be formulated as a continuous variable and the estimation problem is often reduced to solving PDEs using Euler–Lagrange [10, 29]. When the feature values are different or other information such as occlusion needs to be taken into account, one can use belief propagation [22, 49] and graph cuts [9, 31] to optimize objective functions formulated on Markov random fields (MRFs). The recent studies show that optimization tools such as belief propagation, tree-reweighted belief propagation, and graph cuts can achieve very good local optimum for these optimization problems [52]. In [47], a dual layer formulation is proposed to apply tree-reweighted BP to estimate optical flow fields. These advances in inference on MRFs allow us to solve dense scene matching problems effectively.

Scene retrieval, parsing, and recognition have become an important research direction to understand images at the scene level [39, 53]. Image representations, such as color histograms [50], texture models [23], segmented regions [14], GIST descriptors [39], bag-of-words [18] and spatial pyramids [32], have been proposed to find similar images at a global level. Common to all these representations is the lack of meaningful correspondences across different image regions, and therefore, spatial structural information of images tends to be ignored. Our interest is to establish dense correspondences between images across scenes, an alignment problem that can be more challenging than aligning images from the same scene and aligning images of the same object category since we wish all the elements that compose the scene to be aligned. Our work relates to the task of co-segmentation [41] that tried to simultaneously segment the common parts of an image pair, and to the problem of shape matching [5] that was used in the context of object recognition.

Inspired by the recent advances in image alignment and scene parsing, we propose SIFT flow to establish the correspondence between images across scenes. An early version of our work was presented in [34]. In this chapter, we will explore the SIFT flow algorithm in more depth and will demonstrate a wide array of applications for SIFT flow.

3 The SIFT Flow Algorithm

3.1 Dense SIFT Descriptors and Visualization

SIFT is a local descriptor to characterize local gradient information [37]. In [37], SIFT descriptor is a sparse feature representation that consists of both feature extraction and detection. In this chapter, however, we only use the feature extraction component. For every pixel in an image, we divide its neighborhood (e.g. 16×16) into a 4×4 cell array, quantize the orientation into eight bins in each cell, and obtain a $4 \times 4 \times 8 = 128$ -dimensional vector as the SIFT representation for a pixel. We call this per pixel SIFT descriptor *SIFT image*.

To visualize SIFT images, we compute the top three principal components of SIFT descriptors from a set of images and then map these principal components to the principal components of the RGB space, as shown in Fig. 2. Via projection of a 128D SIFT descriptor to a 3D subspace, we are able to compute the SIFT image from an RGB image in Fig. 2c and visualize it in (d). In this visualization, the pixels that have similar color may imply that they share similar local image structures. Note that this projection is only for visualization; in SIFT flow, the entire 128 dimensions are used for matching.

Notice that even though this SIFT visualization may look blurry as shown in Fig. 2d, SIFT images indeed have high spatial resolution as suggested by Fig. 3. We designed an image with a horizontal step edge (Fig. 3a), and show the first component of the SIFT image of (a) in (c). Because every row is the same in (a) and (c), we plot the middle row of (a) and (c) in (b) and (d), respectively. Clearly, the SIFT image contains a sharp edge with respect to the sharp edge in the original image.

Now that we have per pixel SIFT descriptors for two images, our next task is to build dense correspondence to match these descriptors.

3.2 Matching Objective

We designed an objective function similar to that of optical flow to estimate SIFT flow from two SIFT images. Similar to optical flow [10, 12], we want SIFT descriptors to be matched along the flow vectors, and the flow field to be smooth, with discontinuities agreeing with object boundaries. Based on these two criteria, the objective function of SIFT flow is formulated as follows. Let $\mathbf{p} = (x, y)$ be the grid coordinate of images and $\mathbf{w}(\mathbf{p}) = (u(\mathbf{p}), v(\mathbf{p}))$ be the flow vector at \mathbf{p} . We only allow $u(\mathbf{p})$ and $v(\mathbf{p})$ to be integers and we assume that there are L possible states for $u(\mathbf{p})$ and $v(\mathbf{p})$, respectively. Let s_1 and s_2 be two SIFT images that we want to match. Set ε contains all the spatial neighborhoods (a four-neighbor system is used). The energy function for SIFT flow is defined as

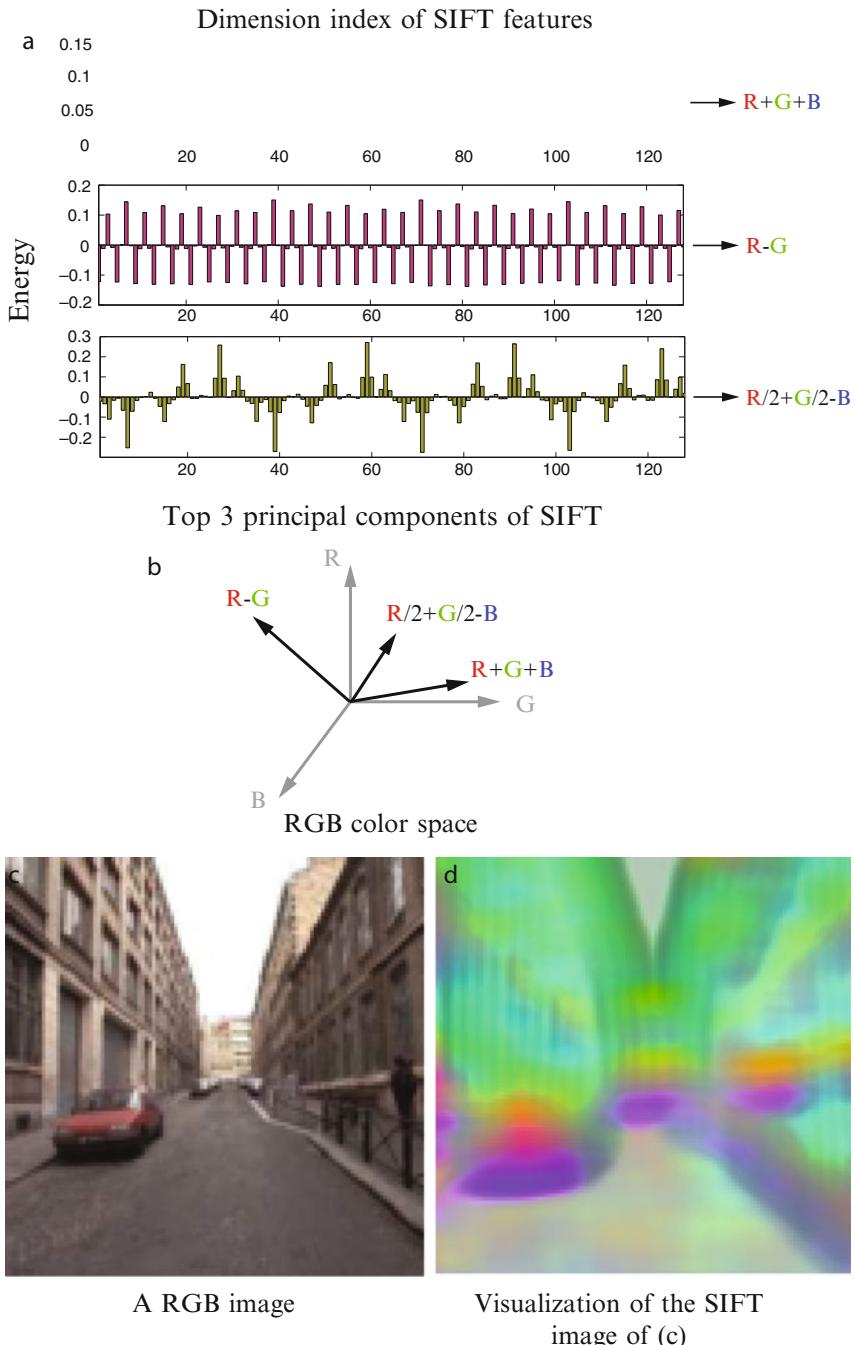


Fig. 2 Visualization of SIFT images. To visualize SIFT images, we compute the top three principal components of SIFT descriptors from a set of images (a) and then map these principal components to the principal components of the RGB space (b). For an image in (c), we compute the 128D SIFT feature for every pixel, project the SIFT feature to 3D color space, and visualize the SIFT image as shown in (d). Intuitively, pixels with similar *colors* share similar structures

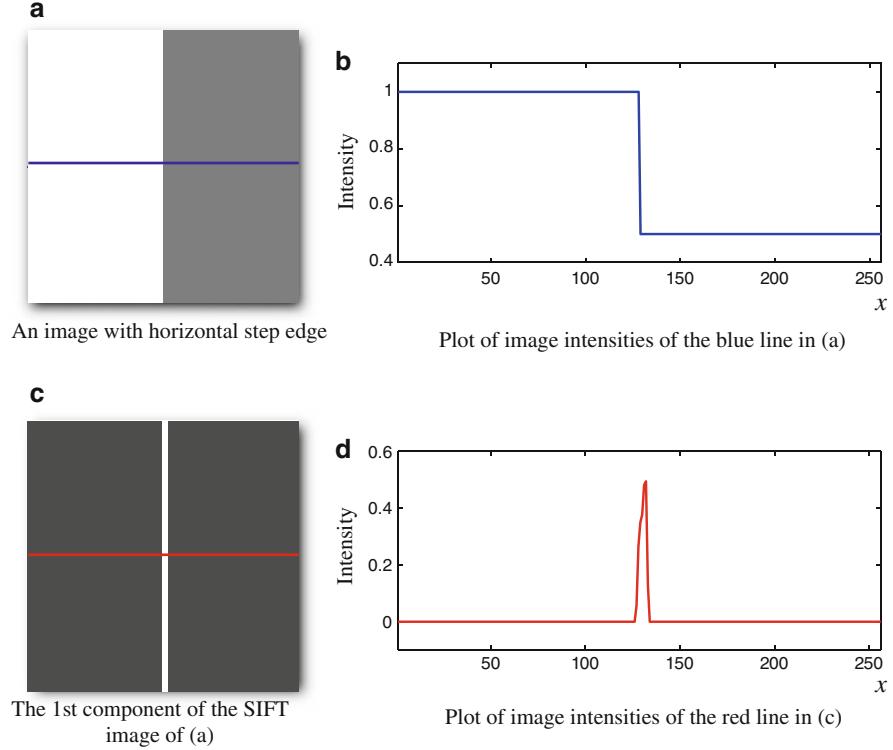


Fig. 3 The resolution of SIFT images. Although histograms are used to represent SIFT features, SIFT images are able to capture image details. For a toy image with a horizontal step edge in (a), we show the first component of the SIFT image in (c). We plot the slice of a *horizontal line* in (a) (blue) and (c) (red) in (b) and (d), respectively. The sharp boundary in (d) suggests that SIFT images have high resolutions

$$E(\mathbf{w}) = \sum_{\mathbf{p}} \min \left(\|s_1(\mathbf{p}) - s_2(\mathbf{p} + \mathbf{w}(\mathbf{p}))\|_1, t \right) \quad (1)$$

$$+ \sum_{\mathbf{p}} \eta(|u(\mathbf{p})| + |v(\mathbf{p})|) \quad (2)$$

$$+ \sum_{(\mathbf{p}, \mathbf{q}) \in \varepsilon} \min \left(\alpha |u(\mathbf{p}) - u(\mathbf{q})|, d \right) \\ + \min \left(\alpha |v(\mathbf{p}) - v(\mathbf{q})|, d \right), \quad (3)$$

which contains a *data term*, *small displacement term*, and *smoothness term* (a.k.a. spatial regularization). The *data term* in Eq. (1) constrains the SIFT descriptors to be matched along with the flow vector $\mathbf{w}(\mathbf{p})$. The *small displacement term* in Eq. (2)

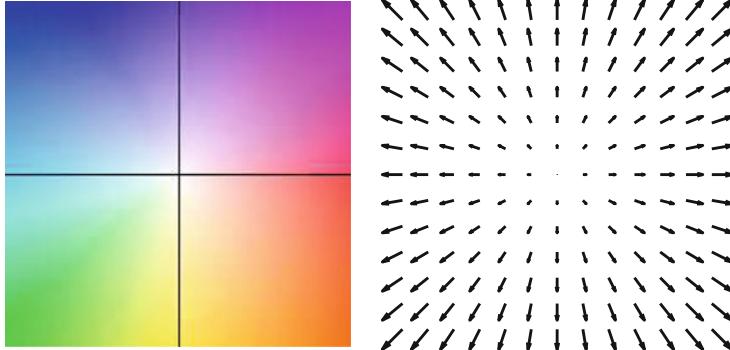


Fig. 4 The visualization of flow fields. We follow the code in [2] to visualize a flow field: each pixel denotes a flow vector where the orientation and magnitude are represented by the huge and saturation of the pixel, respectively

constrains the flow vectors to be as small as possible when no other information is available. The *smoothness term* in Eq. (3) constrains the flow vectors of adjacent pixels to be similar. In this objective function, truncated L1 norms are used in both the data term and the smoothness term to account for matching outliers and flow discontinuities, with t and d as the threshold, respectively.

We use a dual-layer loopy belief propagation as the base algorithm to optimize the objective function. Different from the usual formulation of optical flow [10, 12], the smoothness term in Eq. (3) is decoupled, which allows us to separate the horizontal flow $u(\mathbf{p})$ from the vertical flow $v(\mathbf{p})$ in message passing, as suggested by Shekhovtsov et al. [47]. As a result, the complexity of the algorithm is reduced from $O(L^4)$ to $O(L^2)$ at one iteration of message passing. The factor graph of our model is shown in Fig. 5. We set up a horizontal layer u and a vertical layer v with exactly the same grid, with the data term connecting pixels at the same location. In message passing, we first update *intra*-layer messages in u and v separately and then update *inter*-layer messages between u and v . Because the functional form of the objective function has truncated L1 norms, we use distance transform function [20] to further reduce the complexity, and sequential belief propagation (BP-S) [52] for better convergence.

3.3 Coarse-to-Fine Matching Scheme

Despite the speed up, directly optimizing Eq. (3) using this dual-layer belief propagation scales poorly with respect to image dimension. In SIFT flow, a pixel in one image can literally match to any pixels in the other image. Suppose the image has h^2 pixels, then $L \approx h$ and the time and space complexity of this dual-layer BP

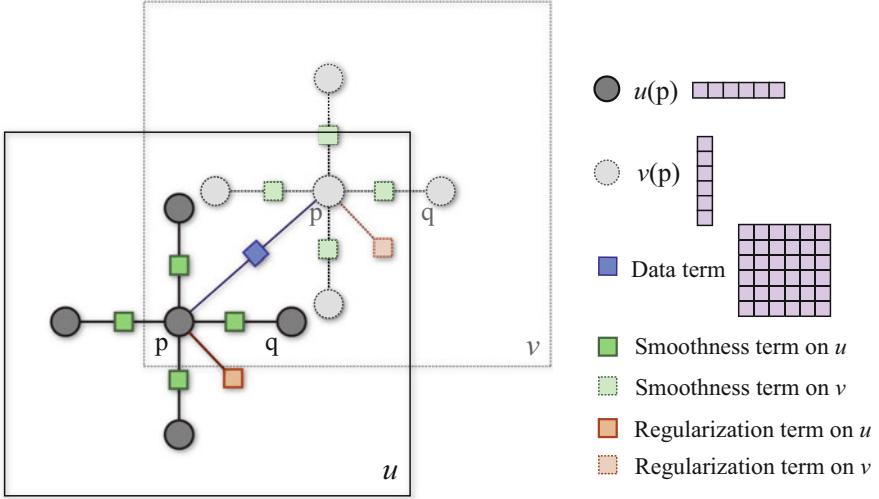


Fig. 5 Dual-layer belief propagation. We designed the objective function of SIFT flow to be decoupled for horizontal (u) and vertical (v) components

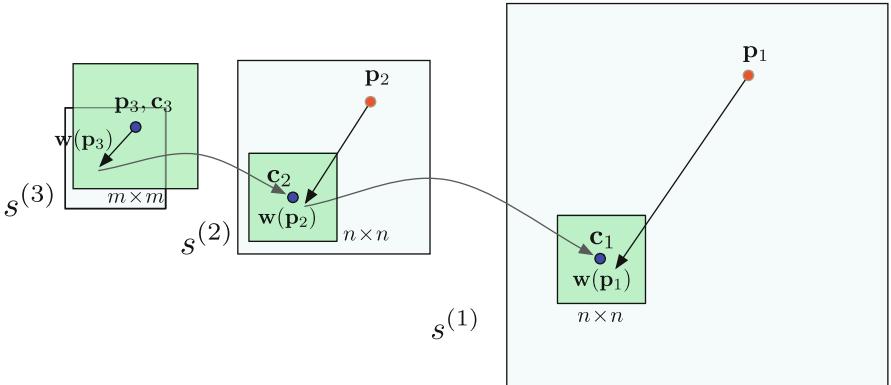


Fig. 6 An illustration of coarse-to-fine SIFT flow matching on pyramid. The *green square* is the searching window for \mathbf{p}_k at each pyramid level k . For simplicity only one image is shown here, where \mathbf{p}_k is on image s_1 , and \mathbf{c}_k and $w(\mathbf{p}_k)$ are on image s_2 . See text for details

is $O(h^4)$. For example, the computation time for 145×105 images with an 80×80 search window is 50 s. It would require more than 2 h to process a pair of 256×256 images with a memory usage of 16 GB to store the data term.

To address the performance drawback, we designed a coarse-to-fine SIFT flow matching scheme that significantly improves the performance. The basic idea is to roughly estimate the flow at a coarse level of image grid, then gradually propagate and refine the flow from coarse to fine. The procedure is illustrated in Fig. 6. For simplicity, we use s to represent both s_1 and s_2 . A SIFT pyramid $\{s^{(k)}\}$ is established,

where $s^{(1)} = s$ and $s^{(k+1)}$ is smoothed and downsampled from $s^{(k)}$. At each pyramid level k , let \mathbf{p}_k be the coordinate of the pixel to match, \mathbf{c}_k be the offset or centroid of the searching window, and $\mathbf{w}(\mathbf{p}_k)$ be the best match from BP. At the top pyramid level $s^{(3)}$, the searching window is centered at \mathbf{p}_3 ($\mathbf{c}_3 = \mathbf{p}_3$) with size $m \times m$, where m is the width (height) of $s^{(3)}$. The complexity of BP at this level is $O(m^4)$. After BP converges, the system propagates the optimized flow vector $\mathbf{w}(\mathbf{p}_3)$ to the next (finer) level to be \mathbf{c}_2 where the searching window of \mathbf{p}_2 is centered. The size of this searching window is fixed to be $n \times n$ with $n = 11$. This procedure iterates from $s^{(3)}$ to $s^{(1)}$ until the flow vector $\mathbf{w}(\mathbf{p}_1)$ is estimated. The complexity of this coarse-to-fine algorithm is $O(h^2 \log h)$, a significant speed up compared to $O(h^4)$. Moreover, we double η and retain α and d as the algorithm moves to a higher level of pyramid in the energy minimization.

When the matching is propagated from a coarser level to a finer level, the searching windows for two neighboring pixels may have different offsets (centroids). We modify the distance transform function developed for truncated L1 norm [20] to cope with this situation, with the idea illustrated in Fig. 7. To compute the message passing from pixel \mathbf{p} to its neighbor \mathbf{q} , we first gather all other messages and data term, and apply the routine in [20] to compute the message from \mathbf{p} to \mathbf{q} assuming that \mathbf{q} and \mathbf{p} have the same offset and range. The function is then extended to be outside the range by increasing α per step, as shown in Fig. 7a. We take the function in the range that \mathbf{q} is relative to \mathbf{p} as the message. For example, If the offset

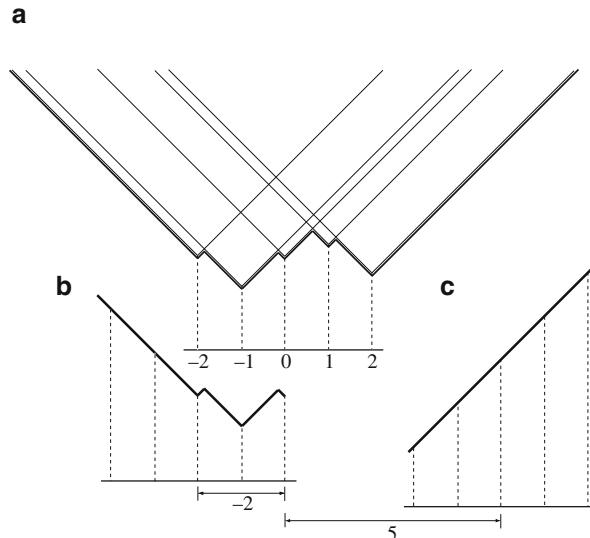


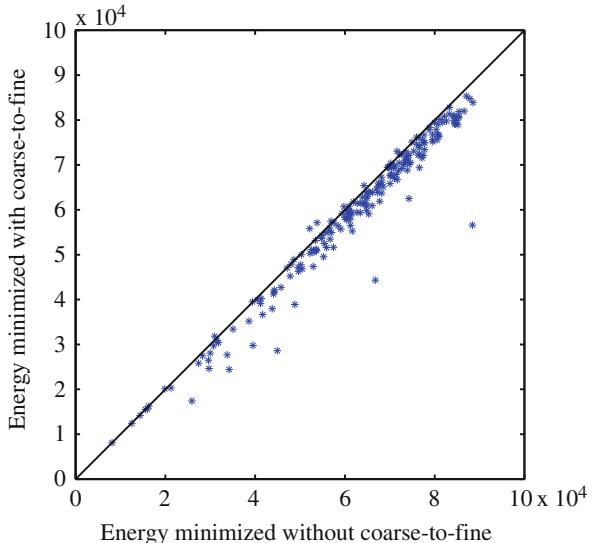
Fig. 7 We generalized the distance transform function for truncated L1 norm [20] to pass messages between neighboring nodes that have different offsets (centroids) of the searching window. See text for more details

of the searching window for \mathbf{p} is 0 and the offset for \mathbf{q} is 5, then the message from \mathbf{p} to \mathbf{q} is plotted in Fig. 7c. If the offset of the searching window for \mathbf{q} is -2 otherwise, the message is shown in Fig. 7b.

Using the proposed coarse-to-fine matching scheme and modified distance transform function, the matching between two 256×256 images takes 31 s on a workstation with two quad-core 2.67 GHz Intel Xeon CPUs and 32 GB memory, in a C++ implementation. Further speedup (up to $50\times$) can be achieved through GPU implementation [16] of the BP-S algorithm since this algorithm can be parallelized. We leave this as future work.

A natural question is whether the coarse-to-fine matching scheme can achieve the same minimum energy as the ordinary matching scheme (using only one level). We randomly selected 200 pairs of images to estimate SIFT flow and check the minimum energy obtained using coarse-to-fine scheme and ordinary scheme (not coarse-to-fine), respectively. For these 256×256 images, the average running time of coarse-to-fine SIFT flow is 31 s, compared to 127 min in average for the ordinary matching. The coarse-to-fine scheme not only runs significantly faster but also achieves lower energies most of the time compared to the ordinary matching algorithm as shown in Fig. 8. This is consistent with what has been discovered in the optical flow community: coarse-to-fine search not only speeds up computation but also leads to better solutions.

Fig. 8 Coarse-to-fine SIFT flow not only runs significantly faster, but also achieves lower energies most of the time. Here we compare the energy minimized using the coarse-to-fine algorithm (y-axis) and using the single level version (x-axis) by running them on 200 pairs of examples. The coarse-to fine matching achieves lower energies compared to the ordinary matching algorithm most of the time



3.4 Neighborhood of SIFT Flow

In theory, we can apply optical flow to two arbitrary images to estimate a correspondence, but we may not get a meaningful correspondence if the two images are from different scene categories. In fact, even when we apply optical flow to two adjacent frames in a video sequence, we assume dense sampling in time so that there is significant overlap between two neighboring frames. Similarly, in SIFT flow, we define the neighborhood of an image as the nearest neighbors when we query a large database with the input. Ideally, if the database is large and dense enough to contain almost every possible image in the world, the nearest neighbors will be close to the query image, sharing similar local structures. This motivates the following analogy with optical flow:

Dense sampling in time	:	optical flow ::
Dense sampling in the space of all images	:	SIFT flow

As dense sampling of the time domain is assumed to enable tracking, dense sampling in (some portion of) the space of world images is assumed to enable scene alignment. In order to make this analogy possible, we collect a large database consisting of 102,206 frames from 731 videos, mostly from street scenes. Analogous to the time domain, we define the “adjacent frames” to a query image as its N nearest neighbors in this database. SIFT flow is then established between the query image and its N nearest neighbors.

For a query image, we use a fast indexing technique to retrieve its nearest neighbors that will be further aligned using SIFT flow. As a fast search we use spatial histogram matching of quantized SIFT features [32]. First, we build a dictionary of 500 visual words [48] by running K -means on 5000 SIFT descriptors randomly selected out of all the video frames in our data set. Then, histograms of the visual words are obtained on a two-level spatial pyramid [24, 32], and histogram intersection is used to measure the similarity between two images.

Other scene metrics such as GIST [39] can also be used for retrieving nearest neighbors [36]. It has been reported that various nearest matching algorithms do not result in significant difference in obtaining nearest neighbors for matching [42] (Figs. 9 and 10).

4 Experiments on Video Retrieval

4.1 Results of Video Retrieval

We conducted several experiments to test the SIFT flow algorithm on our video database. One frame from each of the 731 videos was selected as the query image and histogram intersection matching was used to find its 20 nearest neighbors, excluding all other frames from the query video. The SIFT flow algorithm was

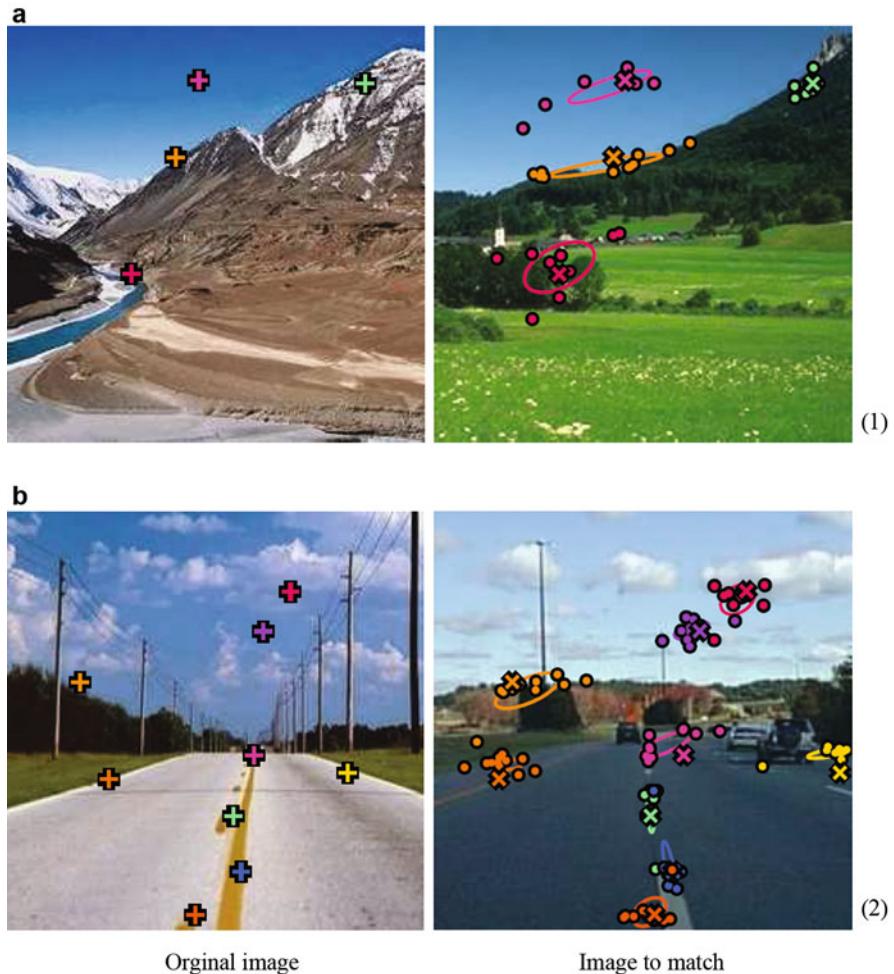


Fig. 9 For an image pair such as row (1) and row (2), a user defines several sparse points in (a) as “+”. The human annotated matchings are marked as *dot* in (b), from which a Gaussian distribution is estimated and displayed as an *ellipse*. The correspondence estimated from SIFT flow is marked as “ \times ” in (b)

then used to estimate the dense correspondence (represented as a pixel displacement field) between the query image and each of its neighbors. The best matches are the ones with the minimum energy defined by (3). Alignment examples are shown in Figs. 11, 12, and 13. The original query image and its extracted SIFT descriptors are shown in columns (a) and (b). The minimum energy match (out of the 20 nearest neighbors) and its extracted SIFT descriptors are shown in columns (c) and (d). To investigate the quality of the pixel displacement field, we use the computed displacements to warp the best match onto the query image. The warped image

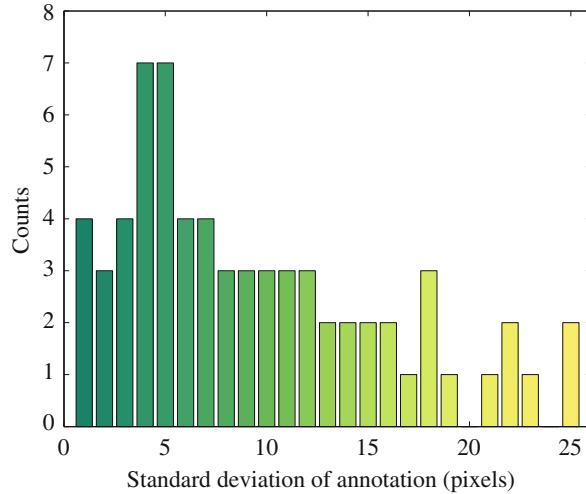


Fig. 10 The evaluation of SIFT flow using human annotation. *Left:* the probability of one human-annotated flow lies within r distance to the SIFT flow as a function of r (red curve). For comparison, we plot the same probability for direct minimum L1-norm matching (blue curve). Clearly, SIFT flow matches human perception better. *Right:* the histogram of the standard deviation of human annotation. Human perception of scene correspondence varies from subject to subject

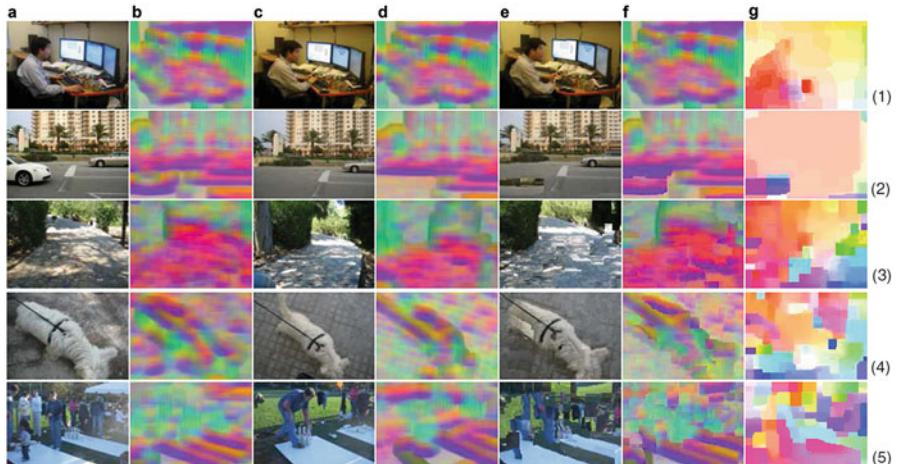


Fig. 11 SIFT flow for image pairs depicting the same scene/object. (a) Shows the query image and (b) its densely extracted SIFT descriptors. (c) and (d) show the best (lowest energy) match from the database and its SIFT descriptors, respectively. (e) shows (c) warped onto (a). (f) shows the warped SIFT image (d). (g) shows the estimated displacement field with the minimum alignment energy shown to the right



Fig. 12 SIFT flow computed for image pairs depicting the same scene/object category where the visual correspondence is obvious

and warped SIFT descriptor image are shown in columns (e) and (f). The visual similarity between (a) and (e) and (b) and (f) demonstrates the quality of the matching. Finally, the displacement field, visualized using the color-coding in Fig. 4 [2], is shown in column (g).

Figure 11 shows examples of matches between frames coming from exactly the same (3D) scene, but different video sequences. The reasonable matching in (1) and (2) demonstrates that SIFT flow reduces to classical optical flow when the two images are temporally adjacent frames in a video sequence. In (3)–(5), the query and the best match are more distant within the video sequence, but the alignment algorithm can still match them reasonably well.

Figure 12 shows more challenging examples, where the two frames come from different videos while containing the same type of objects. SIFT flow attempts to match the query image by reshuffling the pixels in the candidate image. Notice significant changes of objects between the query and the match in examples (8), (9), (11), (13), (14), and (16). The large amount of discontinuities in the flow field is due to: (i) the coefficient on spatial regularization α is small and (ii) the contents of

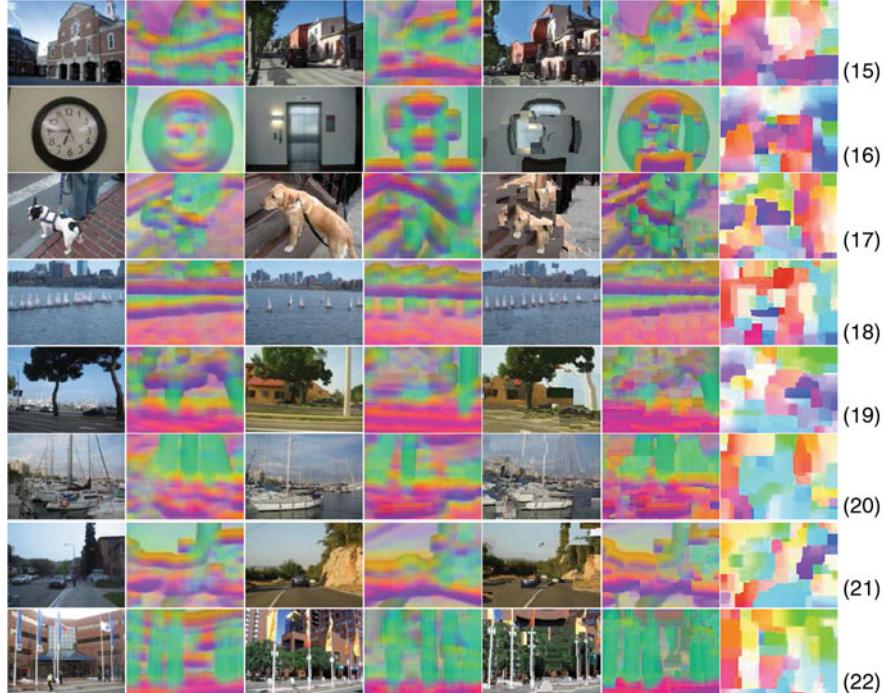


Fig. 13 SIFT flow for challenging examples where the correspondence is not obvious

the two images are too different to produce smooth matches (compared to examples (1) and (2) in Fig. 11). The square-shaped discontinuities are a consequence of the decoupled regularizer on the horizontal and vertical components of the pixel displacement vector.

Figure 13 shows alignment results for examples with no obvious visual correspondences. Despite the lack of direct visual correspondences, SIFT flow attempts to rebuild the house (15), change the shape of the door into a circle (16), or reshuffle boats (18).

Some failure cases are shown in Fig. 14, where the correspondences are not semantically meaningful. Typically, the failures are caused by the lack of visually similar images in the video database for the query image. It shows that our database is not dense enough in the space of images.

We find that SIFT flow improves the ranking of the K -nearest neighbors retrieved by histogram intersection, as illustrated in Fig. 15. This improvement demonstrates that image similarities can be better measured by taking into account displacement, an idea that will be used for later applications of SIFT flow.



Fig. 14 Some failure examples with semantically incorrect correspondences. Although a SIFT flow field is obtained through minimizing the objective function, the query images are rare in the database and the best SIFT flow matches do not belong to the same scene category as the queries. However, these failures can be overcome through increasing the size of the database

4.2 Evaluation of the Dense Scene Alignment

After showing some examples of scene alignment, we want to evaluate how well SIFT flow performs compared to human perception of scene alignment. Traditional optical flow is such a well-defined problem that it is straightforward for humans to annotate motion for evaluation [35]. In the case of SIFT flow, however, there may not be obvious or unique pixel-to-pixel matching as the two images may contain different objects, or the same object categories with very different instances.

To evaluate the matching obtained by SIFT flow, we performed a user study where we showed 11 users image pairs with ten preselected sparse points in the first image and asked the users to select the corresponding points in the second image. This process is explained in Fig. 9. The corresponding points selected by different users can vary, as shown on the right of Fig. 9. Therefore, we use the following metric to evaluate SIFT flow: for a pixel \mathbf{p} , we have several human annotations \mathbf{z}_i as its flow vector and $\mathbf{w}(\mathbf{p})$ as the estimated SIFT flow vector. We compute $\Pr(\exists \mathbf{z}_i, \|\mathbf{z}_i - \mathbf{w}(\mathbf{p})\| \leq r | r)$, namely the probability of one human-annotated flow is within distance r to SIFT flow $\mathbf{w}(\mathbf{p})$. This function of r is plotted on the left of Fig. 9 (red curve). For comparison, we plot the same probability function (blue curve) for minimum L1-norm SIFT matching, i.e., SIFT flow matching without spatial terms. Clearly, SIFT flow matches better to human annotation than minimum L1-norm SIFT matching.

5 Dense Scene Alignment Applications

As illustrated in the previous section, we are able to find dense scene correspondences between two images using SIFT flow, and the correspondence can be semantically meaningful if the two images contain similar objects. In this section,

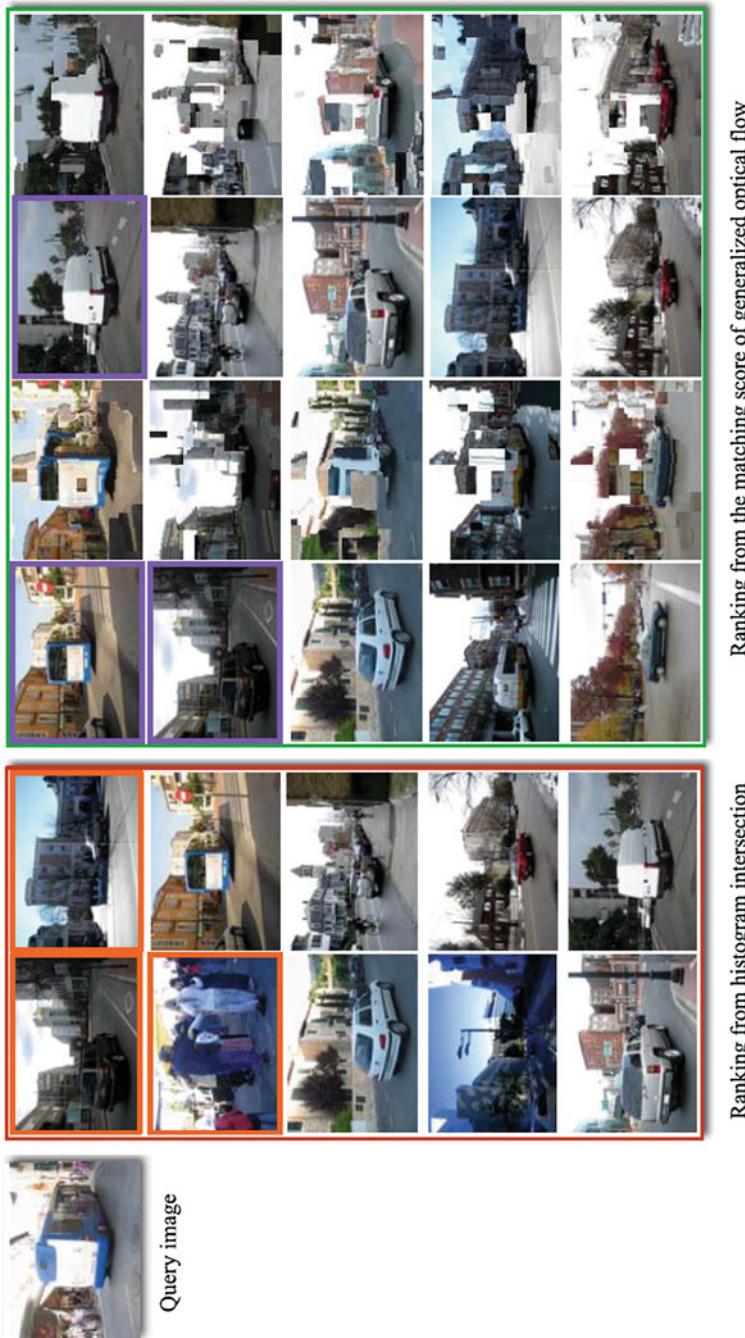


Fig. 15 SIFT flow typically improves ranking of the nearest neighbors. Images enclosed by the *red rectangle* (*middle*) are the top ten nearest neighbors found by histogram intersection, displayed in scan line order (*left to right, top to bottom*). The top three nearest neighbors are enclosed by *orange rectangles*. Images enclosed by the *green rectangle* are the top ten nearest neighbors ranked by the minimum energy obtained by SIFT flow, and the top three nearest neighbors are enclosed by *purple*. The warped nearest neighbor image is displayed to the right of the original image. Note how the retrieved images are re-ranked according to the size of the depicted vehicle by matching the size of the bus in the query

we will introduce two novel applications for dense scene alignment: motion field prediction from a single image and motion synthesis via transfer of moving objects common in similar scenes.

5.1 Predicting Motion Fields from a Single Image

We are interested in predicting motion fields from a single image, namely to know which pixels could move and how they move. This adds potential temporal motion information onto a single image for further applications, such as animating a still image and event analysis.

A scene retrieval infrastructure is established to query still images over a database of videos containing common moving objects. The database consists of sequences depicting common events, such as cars driving through a street and kids playing in a park. Each individual frame was stored as a vector of word-quantized SIFT features, as described in Sect. 3.4. In addition, we store the temporal motion field estimated using [12] between every two consecutive frames of each video.

We compare two approaches for predicting the motion field for a query still image. In the first approach, using the SIFT-based histogram matching in Sect. 3.4, we retrieve nearest neighbors (similar video frames) that are roughly spatially aligned with the query and directly transfer the motion from the nearest neighbors to the query. In the second approach, dense correspondences are estimated between the query and nearest neighbors using SIFT flow, and the temporally estimated motion of the nearest neighbors are warped to the query according to the estimated SIFT flow fields. Figure 16 shows examples of predicted motion fields directly transferred from the top five database matches and the warped motion fields (Fig. 17).

A still image may have multiple plausible motions: a car can move forward, back up, turn, or remain static. This is handled by retrieving multiple nearest neighbors from the video database. Figure 18 shows an example of five motion fields predicted using our video database. All the motion fields are different, but plausible.

5.2 Quantitative Evaluation of Motion Prediction

Due to the inherent ambiguity of multiple plausible motions for a still image, we design the following procedure for quantitative evaluation. For each test video, we randomly select a test frame and obtain a *result* set of top n inferred motion fields using our motion prediction algorithm. Separately, we collect an *evaluation* set containing the temporally estimated motion for the test frame (the closest to a ground truth we have) and 11 random motion fields taken from other scenes in our database, acting as distractors. We take each of the n inferred motion fields from the result set and compute their similarity (defined below) to evaluation set. The rank

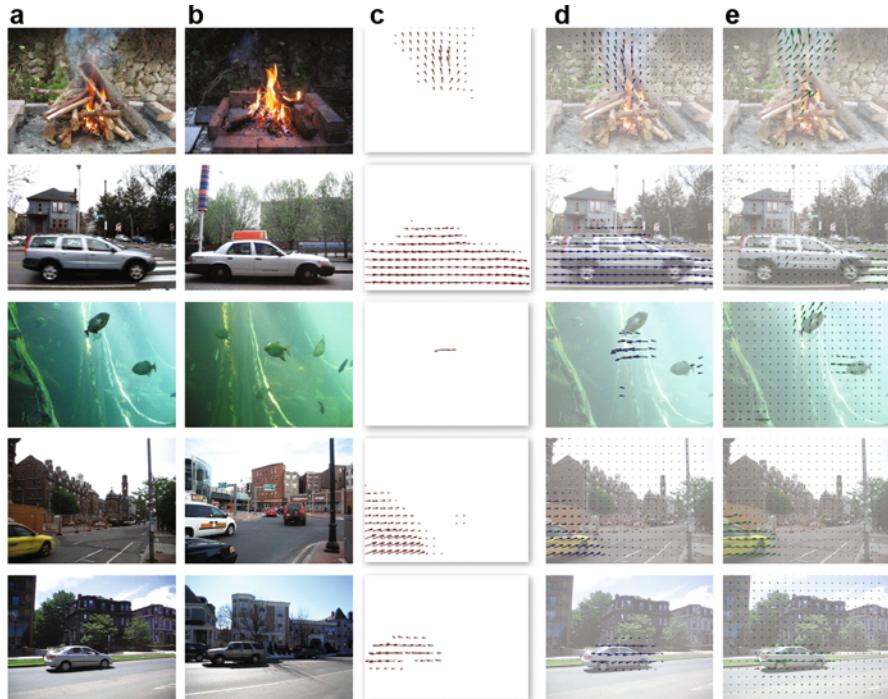


Fig. 16 Motion from a single image. (a) Original image; (b) best match in the video database; (c) temporal motion field of (b); (d) warped motion of (c) and superimposed on (a), according to the estimated SIFT flow; (e) the “ground truth” temporal motion of (a) (estimated from the video containing (a)). The predicted motion is based on the motion present in other videos with image content similar to the query image

of the ground truth motion with respect to the motion of random distractors is an indicator of how close the predicted motion is to the true motion estimated from the video sequence. Because there are many possible motions that are still realistic, we perform this comparison with each of the top n motion fields within the result set and keep the highest ranking achieved. This evaluation is repeated ten times with a different randomly selected test frame for each test video, and the median of the rank score across the different trials is reported.

There are a number of ways of comparing temporal motion (optical flow) fields [2], such as average angular error (AAE). For our experiment, we want to compare two motion fields at a coarse level because careful comparisons such as AAE would not be meaningful. In fact, we care more for *which* pixels move than *how* they move. For this evaluation, therefore, we represent each motion field as a regular two dimensional motion grid filled with 1s where there is motion and 0 otherwise. The similarity between two motion fields is defined as



Fig. 17 Motion synthesis via object transfer. Query images (a), the top video match (b), and representative frames from the synthesized sequence (c) obtained by transferring the moving objects from the video to the still query image



Fig. 18 Multiple motion field candidates. A still query image with its temporally estimated motion field (in the green frame) and multiple motion fields predicted by motion transfer from a large video database

$$S(\mathbf{M}, \mathbf{N}) \stackrel{\text{def}}{=} \sum_{(x,y) \in G} (\mathbf{M}(x,y) = \mathbf{N}(x,y)), \quad (4)$$

where \mathbf{M} and \mathbf{N} are binary motion fields $\mathbf{M}, \mathbf{N} \in \{0, 1\}$. Notice that this formula indeed compares the segmentation of motion fields.

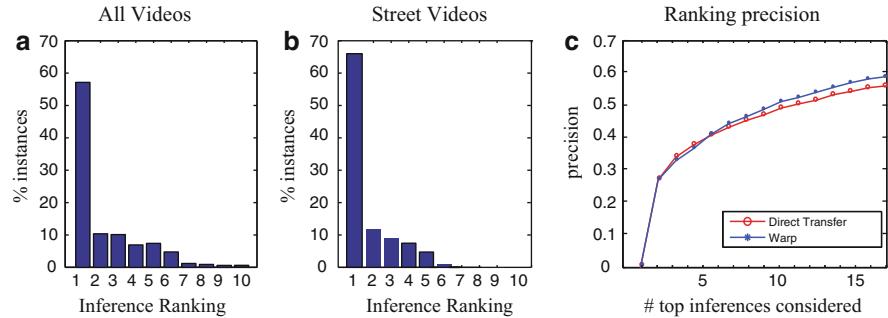


Fig. 19 Evaluation of motion prediction. (a) and (b) Show normalized histograms of prediction rankings (result set size of 15). (c) Shows the ranking precision as a function of the result set size



Fig. 20 Motion instances where the predicted motion was not ranked closest to the ground truth. A set of random motion fields (blue) together with the predicted motion field (green, ranked third). The number above each image represents the fraction of the pixels that were correctly matched by comparing the motion against the ground truth. In this case, some random motion fields appear closer to the ground truth than our prediction (green). However, our prediction also represents a plausible motion for this scene

Figure 19a shows the normalized histogram of these rankings across 720 predicted motion fields from our video data set. Figure 19b shows the same evaluation on a subset of the data that includes 400 videos with mostly streets and cars. Notice how, for more than half of the scenes, the inferred motion field is ranked the first suggesting a close match to the temporally estimated ground truth. Most other test examples are ranked within the top 5. Focusing on roads and cars gives even better results with 66 % of test trials ranked first and even more test examples ranked within the top five. Figure 19c shows the precision of the inferred motion (the percentage of test examples with rank 1) as a function of the size of the result set, comparing (1) direct motion field transfer (red circles) and (2) warped motion field transfer using SIFT flow (blue stars).

While histograms of ranks show that the majority of the inferred motions are ranked first, there are still a number of instances with lower rank. Figure 20 shows

an example where the inferred motion field is not ranked top despite the reasonable output. Notice that the top-ranked distracter fields are indeed quite similar to our prediction, indicating that predicted motion can still be realistic.

5.3 Motion Synthesis via Object Transfer

Besides predicting possible temporal motions for a still image, we want to infer moving objects that can appear in a single image and make a plausible video sequence. For example, a car moving forward can appear in a street scene with an empty road, and a fish may swim in a fish tank scene.

The goal of motion synthesis is to transfer moving objects from similar video scenes to a static image. Given a still image q that is not part of any videos in our database D , we identify and transfer moving objects from videos in D into q as follows:

1. Query D using the SIFT-based scene matching algorithm to retrieve the set of closest video frame matches $F = \{f_i | f_i \text{ is the } i\text{th frame from a video in } D\}$ for the query image q .
2. For each frame $f_i \in F$, synthesize a video sequence G in which the i th frame g_i is generated as follows:
 - (a) Estimate temporal motion field m_i from frame f_i and f_{i+1} ;
 - (b) Perform motion segmentation and obtain the mask of moving pixels: $z_i = |m_i| > T$, where T is a threshold;
 - (c) Treat q as background, f_i as foreground, z_i the mask of foreground, and apply Poisson editing [40] to obtain g_i : $g_i = \text{PoissonBlend}(q, f_i, z_i)$.

Examples of motion synthesis via object transfer are shown in Fig. 17. Given a still query image (a), we use histogram intersection and SIFT flow to find its retrieved video sequences (b) from our database and synthesize a new video sequence (some representative frames are shown in (c)) by transferring the moving objects from (b) into the still image (a). Notice the variety of region sizes transferred and the seamless integration of objects into the new scenes. Although it is challenging to estimate the correct size and orientation of the objects introduced to the still image, our framework inherently takes care of these constraints by retrieving sequences that are visually similar to the query image.

6 Experiments on Image Alignment and Face Recognition

We have demonstrated that SIFT flow can be effectively used for retrieval and synthesis purposes. In this section we show that SIFT flow can be applied to traditional image alignment scenarios to handle challenging registration problems.

6.1 Same Scene Image Registration

Image registration of the same 3D scene can be challenging when there is little overlap between two images or drastic appearance changes due to phenomena such as changes of seasons and variations of imaging conditions (angle, lighting, sensor), geographical deformations, and human activities. Although sparse feature detection and matching has been a standard approach to image registration of the same scene [51, 58], we are curious how SIFT flow would perform for this problem.

Take a look at two satellite images¹ of the same location in Mars, as shown in Fig. 21a, b. Because they were taken 4 years apart, image intensities vary drastically between the two images. For our experiment, we first use sparse SIFT feature detection [37] to detect SIFT feature points on both images ((c) and (d)), and a sparse correspondence is established through minimum SSD matching on SIFT features (e). This sparse correspondence is further interpolated to form a dense flow field as shown in (f). To investigate the quality of this dense correspondence, we warp image (b) to image (a) according to the dense flow field and display the pixel-wise matching error in (g). The mean absolute error of this correspondence is 0.030 (the pixel value is between 0 and 1). Clearly, the underlying correspondence between these two Mars images is not captured by this sparse correspondence approach.

We now apply SIFT flow to align these two images. The SIFT flow field is displayed in (j), and the pixel-wise matching error of the SIFT flow field is displayed in (k). The mean absolute error decreases to 0.021 for SIFT flow, and visually we can see that misalignment has been significantly reduced. To our surprise, there is a fracture in the estimated SIFT flow field in (j), which could be caused by some stitching artifact in the satellite images. This is automatically discovered by SIFT flow.

We further apply SIFT flow to align some challenging examples in [58] (the algorithm proposed in [58] is able to handle all these examples well) and the results are displayed in Fig. 22, where columns (a) and (b) are pairs of images to align. The correspondences between some pairs, e.g., rows (1), (3), and (4) are not obvious to human visual systems. The dense correspondences estimated from SIFT flow are displayed in column (c). For visualization purposes, we warp image 2 to image 1 according to the flow field and display the warped image 2 in column (d). To inspect the quality of the flow, we superimpose warped image 2 to image 1 on a checkerboard, as shown in column (e). From these results, the reader can see that SIFT flow is able to handle challenging image registration problems despite drastically different image appearances and large displacement.

¹Image source: http://www.msss.com/mars_images/moc/2006/12/06/gullies/sirenum_crater/index.html.

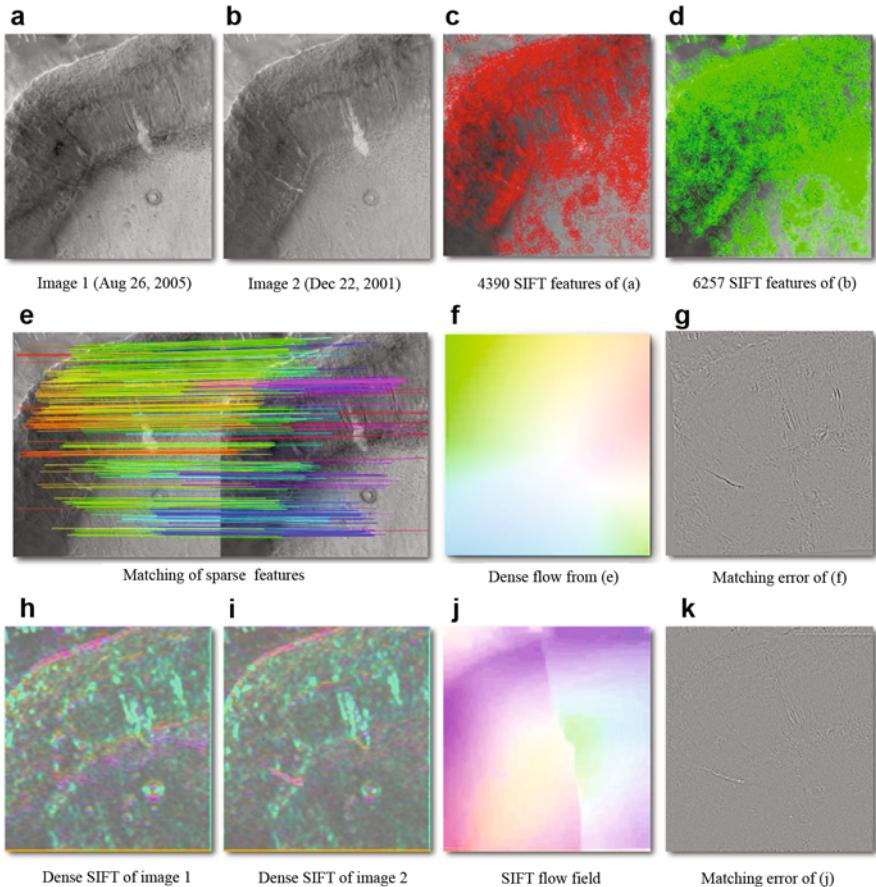


Fig. 21 SIFT flow can be applied to aligning satellite images. The two Mars satellite images (**a**) and (**b**) taken 4 years apart show different local appearances. The results of sparse feature detection and matching are shown in (**c**) to (**g**), whereas the results of SIFT flow are displayed in (**h**) to (**k**). The mean absolute error of the sparse feature approach is 0.030, while the mean absolute error of SIFT flow is 0.021, significantly lower. Please refer to <http://people.csail.mit.edu/celiu/SIFTflow/NGA/> for the animations showing the warping

6.2 Face Recognition

Aligning images with respect to structural image information contributes to building robust visual recognition systems. We design a generic image recognition framework based on SIFT flow and apply it to face recognition, since face recognition can be a challenging problem when there are large pose and lighting variations in large corpora of subjects.

We use the ORL database [44] for this experiment. This database contains a total of 40 subjects and ten images with some pose and expression variation per subject.

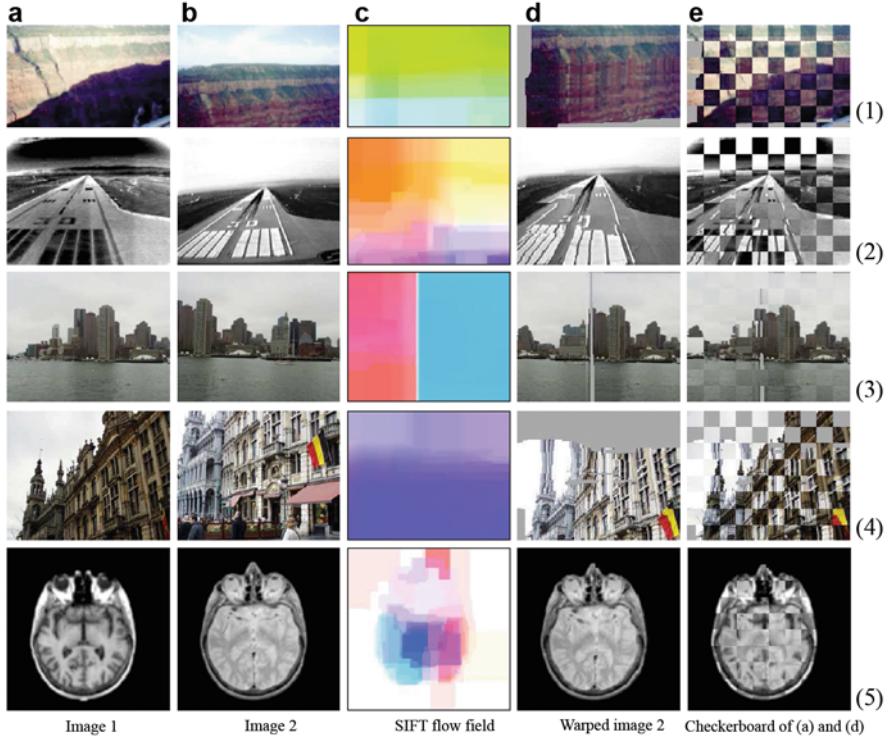


Fig. 22 SIFT flow can be applied to same scene image registration but under different lighting and imaging conditions. Columns (a) and (b) are some examples from [58]. Column (c) is the estimated SIFT flow field, (d) is the warped image 2. In (e), we follow [58] to display (a) and (d) in a checkerboard pattern. Even though originally designed for scene alignment, SIFT flow is also able to align these challenging pairs. Please refer to <http://people.csail.mit.edu/celiu/SIFTflow/NGA/> for the animations of the warping

In Fig. 23, a female sample is selected as an example to demonstrate how dense registration can deal with pose and expression variations. We first select the first image as the query, apply SIFT flow to align the rest of the images to the query, and display the warped images with respect to the SIFT flow field in Fig. 23b. Notice how the poses and expressions of other images are rectified to that of the query. We can also choose a different sample as query and align the rest of the images to this query, as demonstrated in (c). Distances established on images after the alignment will be more meaningful than the distances directly on the original images.

In order to compare with the state of the art, we conducted experiments for both original size (92×112) and downsampled (32×32) images. We randomly split a γ ($\gamma \in (0, 1)$) portion of the samples for each subject for training and use the rest $1 - \gamma$ portion for testing. For each test image, we first retrieve the top nearest neighbors (maximum 20) from the training database using GIST matching [39] and then apply SIFT flow to find the dense correspondence from the test to each of its



Fig. 23 SIFT flow can account for pose, expression, and lighting changes for face recognition. **(a)** Ten samples of one subject in ORL database [44]. Notice pose and expression variations of these samples. **(b)** We select the first image as the query, apply SIFT flow to align the rest of the images to the query, and display the warped images with respect to the dense correspondence. The poses and expressions are rectified to that of the query after the warping. **(c)** The same as **(b)** except for choosing the fifth sample as the query

nearest neighbors by optimizing the objective function in Eq.(3). We assign the subject identity associated with the best match, i.e., the match with the minimum matching objective, to the test image. In other words, this is a nearest-neighbor approach where the SIFT flow score is as the distance metric for object recognition.

The experimental results are shown in Fig. 24. We use the nearest-neighbor classifier based on pixel-level Euclidean distance (Pixels + NN + L2) and nearest-neighbor classifier using the L1-norm distance on GIST [39] features (GIST + NN + L1) as the benchmark. Clearly, GIST features outperform raw pixel values since GIST feature is invariant to lighting changes. SIFT flow further improves the performance as SIFT flow is able to align images across different poses. We observe that SIFT flow boosts the classification rate significantly especially when there are not enough samples (small γ). We compare the performance of SIFT flow with the state of the art [13], where facial components are explicitly detected and aligned. The results of few training samples are listed in Table 1. Our recognition system based on SIFT flow is comparable with the state of the art when there are very few samples for training.

7 Discussions

7.1 Scene Alignment

We introduced a new concept of image alignment, *scene alignment*, to establish dense correspondences between images across scenes, as illustrated by the examples in Fig. 1c. The concept of scene alignment advances image alignment from pixel and object levels to a new, scene level. Although seemingly impossible at a first glance, we showed in this chapter that dense scene alignment can be obtained by matching in large database using SIFT flow. We also demonstrated through many examples that scene alignment can be a very useful tool to many computer vision problems.

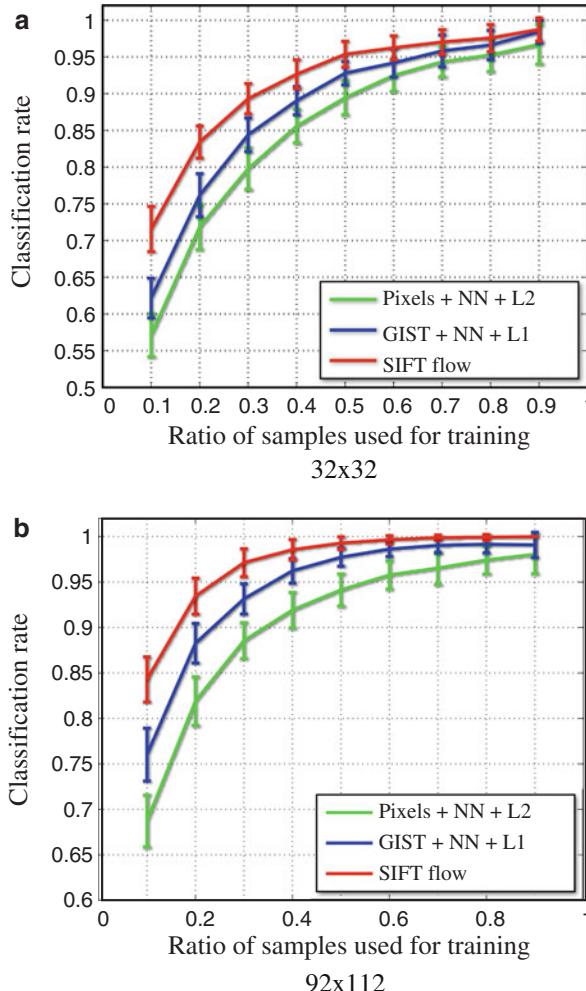


Fig. 24 SIFT flow is applied for face recognition. The curves in (a) and (b) are the performance plots for low-res and high-res images in the ORL face database, respectively. SIFT flow significantly boosted the recognition rate especially when there are not enough training samples

Table 1 Our face recognition system using SIFT flow is comparable with the state of the art [13] when there are only few (one or two) training samples

Test errors	1 Train	2 Train	3 Train
S-LDA [13]	N/A	17.1 ± 2.7	8.1 ± 1.8
SIFT flow	28.4 ± 3.0	16.6 ± 2.2	8.9 ± 2.1

7.2 Sparse vs. Dense Correspondence

There have been two schools of thought for image alignment: dense and sparse correspondence. In the sparse representation, images are summarized as feature points such as Harris corners [27], SIFT [37], and many others [46]. Correspondence is then established by matching these feature points. The algorithms based on the sparse representations are normally efficient and are able to handle lighting changes and large displacements. In the dense representation, however, correspondence is established at the pixel level in the two images, e.g., optical flow field for motion analysis and disparity field for stereo. Because of spatial regularities (i.e., the flow vectors of neighboring pixels are similar), estimating flow fields is reduced to optimization in MRFs. Despite the challenges in optimizing MRFs, via dense correspondence, we can easily warp one image to the other, and this warping can be very useful in many applications.

SIFT flow inherits the merits of both the dense representation by obtaining pixel-to-pixel correspondences and the sparse representation by matching transform-invariant feature of SIFT. In Sect. 4, we demonstrated that SIFT flow is able to align images across scenes, a task that cannot be achieved by traditional optical flow. In Sect. 6.1, we showed that SIFT flow outperforms traditional sparse feature matching in aligning satellite images of the same scene but different appearances. Therefore, SIFT flow becomes a suitable tool for scene alignment.

An important direction for improving SIFT flow is speed. The current system cannot be used for real-time image or video retrieval and matching. One potential approach is the GPU implementation [16] of the BP-S algorithm, which can get up to 50 \times speedup. However, we feel that there could be essential speedup from the sparse matching. The bottleneck of SIFT flow is the large search window size as the locations of objects may change drastically from one image to the other. The sparse, independent matching provides good, approximate matching for sparse points, and this correspondence can be propagated by abiding by the spatial regularities.

7.3 SIFT Flow vs. Optical Flow

Although derived from optical flow, SIFT flow is drastically different from optical flow. In SIFT flow, correspondence is built upon pixel-wise SIFT features instead of RGB or gradient that was used in optical flow [10]. We formulated a discrete optimization framework for SIFT flow, whereas often a continuous optimization is incurred for optical flow as sub-pixel precision is required. Even if optical flow is formulated in a discrete manner, the search window size in SIFT flow is much larger than that in optical flow as we want to handle large location changes of objects in SIFT flow.

However, the similarity between SIFT flow and optical flow can be helpful. Inspired by the coarse-to-fine scheme in optical flow, we also designed a coarse-to-fine matching scheme for SIFT flow that improves both speed and quality. Similar

to the dense temporal sampling as the foundation for obtaining meaningful optical flow fields, we proposed dense sampling in the space of world images for obtaining potentially semantically meaningful SIFT flow fields, namely correspondences are established between objects of the same categories.

Can we apply SIFT flow to analyze temporal motion? On the one hand, SIFT flow can be complementary to optical flow. Examples (1) and (2) in Fig. 11 and the results on satellite image alignment in Sect. 6.1 suggest the possibility. Recently, matching image features such as SIFT has been integrated into the traditional optical flow estimation framework to improve temporal motion estimation [11]. On the other hand, the continuous optical flow model can achieve sub-pixel accuracy, but discrete matching-based SIFT flow can only achieve pixel-level accuracy. Therefore, our intuition is that optical flow cannot be replaced by SIFT flow.

7.4 An Alignment-Based Large Database Framework for Image Analysis and Synthesis

Using SIFT flow as a tool for scene alignment, we designed an alignment-based large database framework for image analysis and synthesis, as illustrated in Fig. 25. For a query image, we retrieve a set of nearest neighbors in the database and transfer information such as motion, geometry, and labels from the nearest neighbors to the query. This framework is concretely implemented in motion prediction from a single image (Sect. 5.1), motion synthesis via object transfer (Sect. 5.3) and face recognition (Sect. 6.2). In [36], the same framework was applied for object recognition and scene parsing. Although large-database frameworks have been used before in visual recognition [42] and image editing [28], the dense scene alignment component of our framework allows greater flexibility for information transfer in limited data scenarios.

8 Conclusion

We introduced the concept of dense scene alignment: to estimate the dense correspondence between images across scenes. We proposed SIFT flow to match salient local image structures with spatial regularities and conjectured that matching in a large database using SIFT flow leads to semantically meaningful correspondences for scene alignment. Extensive experiments verified our theory, showing that SIFT flow is capable of establishing dense scene correspondence despite significant differences in appearances and spatial layouts of matched images. We further proposed an alignment-based large database framework for image analysis and synthesis where image information is transferred from the nearest neighbors in a large database to a query image according to the dense scene correspondence estimated by SIFT flow. This framework is concretely realized in motion prediction

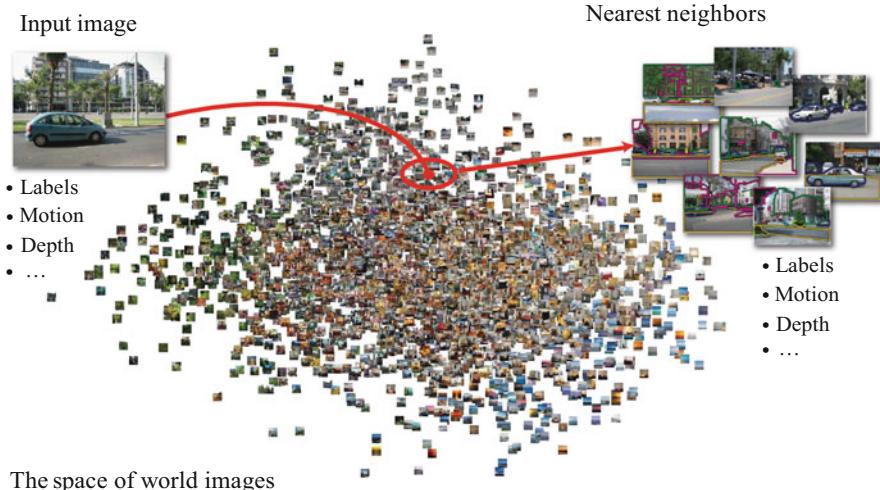


Fig. 25 An alignment-based large database framework for image analysis and synthesis. Under this framework, an input image is processed by retrieving its nearest neighbors and transferring their information according to some dense scene correspondence

from a single image, motion synthesis via object transfer and face recognition. We also applied SIFT flow to traditional image alignment problems. The preliminary success of these experiments suggests that scene alignment using SIFT flow can be a useful tool for various applications in computer vision and computer graphics.

The SIFT flow code package can be downloaded at <http://people.csail.mit.edu/celiu/SIFTflow/>.

References

1. Avidan, S.: Ensemble tracking. *IEEE Trans. Pattern Anal. Mach. Intell.* **29**(2), 261–271 (2007)
2. Baker, S., Scharstein, D., Lewis, J.P., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. In: Proceeding of ICCV (2007)
3. Barron, J.L., Fleet, D.J., Beauchemin, S.S.: Systems and experiment performance of optical flow techniques. *Int. J. Comput. Vis.* **12**(1), 43–77 (1994)
4. Belongie, S., Malik, J., Puzicha, J.: Shape context: a new descriptor for shape matching and object recognition. In: Advances in Neural Information Processing Systems (NIPS) (2000)
5. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.* **24**(4), 509–522 (2002)
6. Berg, A., Berg, T., Malik, J.: Shape matching and object recognition using low distortion correspondence. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005)
7. Bergen, J.R., Anandan, P., Hanna, K.J., Hingorani, R.: Hierarchical model-based motion estimation. In: European Conference on Computer Vision (ECCV), pp. 237–252 (1992)
8. Black, M.J., Anandan, P.: The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Comput. Vis. Image Underst.* **63**(1), 75–104 (1996)

9. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.* **23**(11), 1222–1239 (2001)
10. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High accuracy optical flow estimation based on a theory for warping. In: European Conference on Computer Vision (ECCV), pp. 25–36 (2004)
11. Brox, T., Bregler, C., Malik, J.: Large displacement optical flow. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
12. Bruhn, A., Weickert, J., Schnörr, C.: Lucas/Kanade meets Horn/Schunk: combining local and global optical flow methods. *Int. J. Comput. Vis.* **61**(3), 211–231 (2005)
13. Cai, D., He, X., Hu, Y., Han, J., Huang, T.: Learning a spatially smooth subspace for face recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2007)
14. Carson, C., Belongie, S., Greenspan, H., Malik, J.: Blobworld: color- and texture-based image segmentation using EM and its application to image querying and classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **24**(8), 1026–1038 (2002)
15. Cootes, T.F., Edwards, G.J., Taylor, C.J.: Active appearance models. In: European Conference on Computer Vision (ECCV), vol. 2, pp. 484–498 (1998)
16. Cornelis, N., Gool, L.V.: Real-time connectivity constrained depth map computation using programmable graphics hardware. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1099–1104 (2005)
17. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005)
18. Fei-Fei, L., Perona, P.: A bayesian hierarchical model for learning natural scene categories. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 524–531 (2005)
19. Felzenszwalb, P., Huttenlocher, D.: Pictorial structures for object recognition. *Int. J. Comput. Vis.* **61**(1), 55–79 (2005)
20. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. *Int. J. Comput. Vis.* **70**(1), 41–54 (2006)
21. Fleet, D.J., Jepson, A.D., Jenkin, M.R.M.: Phase-based disparity measurement. *Comput. Vis. Graph. Image Process.* **53**(2), 198–210 (1991)
22. Freeman, W.T., Pasztor, E.C., Carmichael, O.T.: Learning low-level vision. *Int. J. Comput. Vis.* **40**(1), 25–47 (2000)
23. Gorkani, M.M., Picard, R.W.: Texture orientation for sorting photos at a glance. In: IEEE International Conference on Pattern Recognition (ICPR), vol. 1, pp. 459–464 (1994)
24. Grauman, K., Darrell, T.: Pyramid match kernels: discriminative classification with sets of image features. In: IEEE International Conference on Computer Vision (ICCV) (2005)
25. Grimson, W.E.L.: Computational experiments with a feature based stereo algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.* **7**(1), 17–34 (1985)
26. Hannah, M.J.: Computer matching of areas in stereo images. Ph.D. thesis, Stanford University (1974)
27. Harris, C., Stephens, M.: A combined corner and edge detector. In: Proceedings of the 4th Alvey Vision Conference, pp. 147–151 (1988)
28. Hays, J., Efros, A.A.: Scene completion using millions of photographs. *ACM SIGGRAPH* **26**(3) (2007)
29. Horn, B.K.P., Schunck, B.G.: Determining optical flow. *Artif. Intell.* **17**, 185–203 (1981)
30. Jones, D.G., Malik, J.: A computational framework for determining stereo correspondence from a set of linear spatial filters. In: European Conference on Computer Vision (ECCV), pp. 395–410 (1992)
31. Kolmogorov, V., Zabih, R.: Computing visual correspondence with occlusions using graph cuts. In: IEEE International Conference on Computer Vision (ICCV), pp. 508–515 (2001)
32. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. II, pp. 2169–2178 (2006)

33. Liu, C., Freeman, W.T., Adelson, E.H.: Analysis of contour motions. In: Advances in Neural Information Processing Systems (NIPS) (2006)
34. Liu, C., Yuen, J., Torralba, A., Sivic, J., Freeman, W.T.: SIFT flow: dense correspondence across different scenes. In: European Conference on Computer Vision (ECCV) (2008)
35. Liu, C., Freeman, W.T., Adelson, E.H., Weiss, Y.: Human-assisted motion annotation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2008)
36. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing: label transfer via dense scene alignment. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
37. Lowe, D.G.: Object recognition from local scale-invariant features. In: IEEE International Conference on Computer Vision (ICCV), Kerkyra, pp. 1150–1157 (1999)
38. Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 674–679 (1981)
39. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int. J. Comput. Vis.* **42**(3), 145–175 (2001)
40. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. *ACM SIGGRAPH* **22**(3), 313–318 (2003)
41. Rother, C., Minka, T., Blake, A., Kolmogorov, V.: Cosegmentation of image pairs by histogram matching – incorporating a global constraint into MRFs. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 993–1000 (2006)
42. Russell, B.C., Torralba, A., Liu, C., Fergus, R., Freeman, W.T.: Object recognition by scene alignment. In: Advances in Neural Information Processing Systems (NIPS) (2007)
43. Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T.: LabelMe: a database and web-based tool for image annotation. *Int. J. Comput. Vis.* **77**(1–3), 157–173 (2008)
44. Samaria, F., Harter, A.: Parameterization of a stochastic model for human face identification. In: IEEE Workshop on Applications of Computer Vision (1994)
45. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vis.* **47**(1), 7–42 (2002)
46. Schmid, C., Mohr, R., Bauckhage, C.: Evaluation of interest point detectors. *Int. J. Comput. Vis.* **37**(2), 151–172 (2000)
47. Shekhovtsov, A., Kovtun, I., Hlavac, V.: Efficient MRF deformation model for non-rigid image matching. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2007)
48. Sivic, J., Zisserman, A.: Video Google: a text retrieval approach to object matching in videos. In: IEEE International Conference on Computer Vision (ICCV) (2003)
49. Sun, J., Zheng, N., Shum, H.: Stereo matching using belief propagation. *IEEE Trans. Pattern Anal. Mach. Intell.* **25**(7), 787–800 (2003)
50. Swain, M.J., Ballard, D.H.: Color indexing. *Int. J. Comput. Vis.* **7**(1), 11–32 (1991)
51. Szeliski, R.: Image alignment and stitching: a tutorial. *Found. Trends Comput. Graph. Comput. Vis.* **2**(1), 1–104 (2006)
52. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for Markov random fields with smoothness-based priors. *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(6), 1068–1080 (2008)
53. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: a large dataset for non-parametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(11), 1958–1970 (2008)
54. Viola, P., Wells, W., III: Alignment by maximization of mutual information. In: IEEE International Conference on Computer Vision (ICCV), pp. 16–23 (1995)
55. Weiss, Y.: Interpreting images by propagating bayesian beliefs. In: Advances in Neural Information Processing Systems (NIPS), pp. 908–915 (1997)
56. Weiss, Y.: Smoothness in layers: motion segmentation using nonparametric mixture estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 520–527 (1997)

57. Winn, J., Jojic, N.: Locus: learning object classes with unsupervised segmentation. In: IEEE International Conference on Computer Vision (ICCV), pp. 756–763 (2005)
58. Yang, G., Stewart, C.V., Sofka, M., Tsai, C.L.: Registration of challenging image pairs: initialization, estimation, and decision. *IEEE Trans. Pattern Anal. Mach. Intell.* **29**(11), 1973–1989 (2007)

Dense, Scale-Less Descriptors

Tal Hassner, Viki Mayzels, and Lihi Zelnik-Manor

Abstract Establishing correspondences between two images requires matching similar image regions. To do this effectively, local representations must be designed to allow for meaningful comparisons. As we discuss in previous chapters, one such representation is the SIFT descriptor used by SIFT flow. The scale selection required to make SIFT scale invariant, however, is only known to be possible at sparse interest points, where local image information varies sufficiently. SIFT flow and similar methods consequently settle for descriptors extracted using manually determined scales, kept constant for all image pixels. In this chapter we discuss alternative representations designed to capture multiscale appearance, even in image regions where existing methods for scale selection are not effective. We show that (1) SIFTs extracted from different scales, even in low contrast areas, vary in their values and so single scale selection often results in poor matches when images show content at different scales. (2) We propose representing pixel appearances with sets of SIFTs extracted at multiple scales. Finally, (3) low-dimensional, linear subspaces are shown to accurately represent such SIFT sets. By mapping these subspaces to points we obtain a novel representation, the Scale-Less SIFT (SLS), which can be used in a dense manner, throughout the image, to represent multiscale image appearances. We demonstrate how the use of the SLS descriptor can act as an alternative to existing, single scale representations, allowing for accurate dense correspondences between images with scale-varying content.

1 Introduction

Scale-invariant feature detectors, such as the Harris–Laplace [20] and robust descriptors such as the SIFT [17] have played pivotal roles in the development of

T. Hassner (✉)

Department of Mathematics and Computer Science, The Open University of Israel,
Raanana, Israel

e-mail: hassner@openu.ac.il

V. Mayzels • L. Zelnik-Manor

Technion Israel Institute of Technology, Haifa, Israel
e-mail: viki.mayzels@gmail.com; lihi@ee.technion.ac.il

modern computer vision systems. Their underlying idea is that one (or several) scales are selected at various image locations using a scale covariant function [e.g., the Laplacian of Gaussians (LoG)]. Presumably, the local extrema of such functions occur at scales corresponding to the same visual information in different images. This allows features extracted at these scales to be matched between images, even if they present information at different scales [22]. Typically, however, most images yield very few pixels for which such scales may be determined.

The methods described in the previous chapters typically compute dense correspondences by extracting image representations at a single, constant scale; the same scale used for all the pixels in both images. These include the traditional stereo methods (which often restrict settings to few or no scale changes between images [7]) or more modern methods such as SIFT flow [15, 16], which uses the Dense-SIFT (DSIFT) descriptor [30]; essentially SIFT [17] extracted for all image pixels at a single, constant scale. These methods are therefore all unsuitable whenever image portions change their scale from one image to the other (e.g., Fig. 1).

We describe an alternative approach to pixel representation, designed to capture image information in multiple scales, without relying on scale selection. By removing the need to choose a single scale, this approach can be applied anywhere in the image, in particular in the vast majority of pixels, where effective scale selection methods have so far been unavailable. The chapter’s topics include the following:

1. We examine the behavior of robust descriptors, namely SIFT [17], at multiple scales. We show that even in low contrast regions, where scale selection is difficult, SIFT descriptors vary in their values and cannot accurately be matched if extracted at different scales. Arbitrary, constant scale selection used by SIFT flow with DSIFT therefore causes false matches when dense correspondences are computed between images in different scales.
2. Rather than extracting a single SIFT descriptor at a single scale, we propose representing pixels by sets of SIFT descriptors extracted at multiple scales.

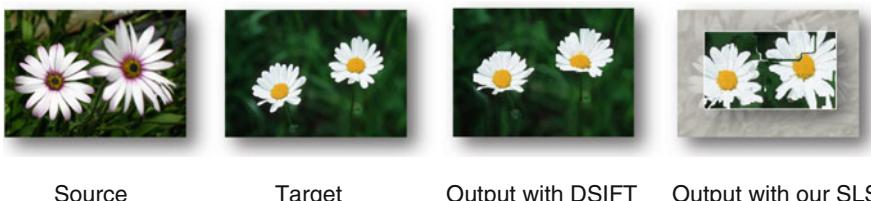


Fig. 1 Dense correspondences of different scenes in different scales. Correspondences computed from *Source* to *Target* images using SIFT flow. Results are shown for the original *DSIFT* descriptor and our multiscale *SLS* representation. Our result is shown overlaid on the *Source*, manually cropped, to demonstrate alignment accuracy. DSIFT cannot compare visual information from different scales. SIFT flow with DSIFT therefore returns a flow field optimized for small (near-zero) displacements, rather than appearance similarities. SLS captures scale changes at each pixel and so accurately maps *Target* appearances to their matching *Source* pixels

Matching these sets, from one image to another, can then be performed using set-to-set similarities. The computational cost of matching more descriptors is balanced by a substantial boost in accuracy.

3. We demonstrate that sets of SIFTs produced by varying only scale lie on low-dimensional subspaces. By using a subspace-to-point mapping [3, 4], each such subspace is converted to a point representation (a descriptor): the Scale-Less SIFT (SLS). SLS is shown to capture multiscale information and can be extracted anywhere in the image. It is therefore suitable as a robust representation for dense correspondence estimation, even when image content varies in scale.
4. Finally, the chapter details comprehensive qualitative and quantitative experiments testing the proposed SLS representation on a range of dense correspondence estimation applications.

This chapter is based on work which previously appeared in [10].

2 Previous Work

Images show information appearing in different scales. To describe and match visual appearances—either locally, per pixel, or globally for an entire image—such scale differences must be addressed. When global scale differences separate two images, these differences can typically be eliminated by global alignment techniques. Often, however, appearances vary in scale throughout the image. For example, a car moving up the street, closer to the camera, would increase in scale from one image to another. All the while, buildings in the street would remain at the same scale in all images.

In order to address these changing scales, early scale-space methods processed and compared images at multiple scales. Later methods, proposed since the early 1990s, instead sought stable, *characteristic* scales for image pixels. This had both computational advantages (less information was analyzed by higher level visual processes) and improved accuracy by focusing on more relevant information. For more on these early approaches, we refer to [13].

Selecting “interesting scales” for image features was originally proposed by Lindeberg in [14]. These scales presumably reflected a characteristic size of scene elements appearing in the image. In order to select such scales, he proposed seeking extrema in the LoG function computed over the image scales at each pixel. This function does not reach an extreme value at all pixels. Such pixels are therefore typically discarded. Scales in other pixels are also rejected whenever these extreme values do not cross a predefined threshold value, designed to remove pixels at low contrast image regions.

The LoG function used in [14] was replaced with the more efficiently computed Differences of Gaussian (DoG) filters function by Lowe in [17]. He proposed processing an image by considering three sets of sub-octave, DoG filters, and scanning them to locate local peaks in the resulting 3D structure (x , y , and *scale*). Here too, detections located in low contrast areas or near edges are rejected.

The Harris–Laplace detector [20], as well as others, localize scales along with spatial scale selection. It uses a scale-adapted Harris corner detector to localize points spatially and LoG filter extrema to localize points in scale, performing these two steps iteratively, seeking peaks in the two values. Similar to previous methods, it too rejects points with low responses, as being unreliable.

All these methods, as well as many others proposed over the years, produce very small sets of interest points, identified by their spatial position and selected scale. These points are typically located near corner structures in the image or in circular structures. In [19] it has been estimated that under a scale change factor of 4.4, pixels with scale detections produced by the DoG detector make up about 38 % of the image. Of these only 10.6 % were actually assigned with correct scales.

To produce dense correspondences, some have proposed using the few scale-invariant keypoints returned by existing detectors as seed for dense matches in wide baseline stereo systems [6, 26, 33]. None of these methods, however, are designed to provide dense correspondences across scale differences. The work of [25] uses few scale-invariant features to locate a (known) object in an image. It then returns dense matches from a reference image of the object to its appearance in the image, along with its segmentation. In order to do this, however, they rely on global alignment to overcome global scale changes. When such alignment is not available (e.g., several independent scene motions and deformable shapes), it is not clear how well this method would perform.

Related to the method described in this chapter are the Scale-Invariant Descriptors (SID) of [11]. SID, like our own SLS descriptors, are designed to capture information from multiple scales, without relying on a preliminary scale selection step. They are produced by transforming the intensities around a pixel to a log-polar coordinate system. This converts scale and rotation differences to translation differences. Translation invariance is then introduced by applying FFT which results in the final SID representation. Though scale and rotation invariant, even on a dense grid, SID descriptors are produced from intensities, directly, and so are less suited for matching images of different scenes [10]. Finally Varma and Garg [29] avoid scale selection by using multiscale fractal features, developed for texture classification tasks.

Dense Correspondences with Scale Differences In this chapter we rely and compare against what has recently become a standard approach to dense correspondences: forgoing scale estimation in favor of descriptors extracted on a regular grid using constant, arbitrarily selected scales. An example of such methods is the efficient DAISY representation of [28] used by the DAISY Filter Flow method of [32]. More related to us are the DSIFT descriptors [30] used by the SIFT flow [15, 16]. In object recognition tasks, regular sampling strategies such as these have been shown to outperform systems utilizing invariant features generated at stable coordinates and scales [23]. This may be due to the benefits of having descriptors for many pixels over accurate scales for just a few.

Dense correspondence estimation methods have largely ignored the issue of scale invariance. SIFT flow [15, 16] produces DSIFT descriptors at each pixel location.



Fig. 2 Dense correspondences with changing scales, comparing DSIFT vs. our own SLS descriptor. Target images warped onto Source using dense correspondences obtained by SIFT flow [15, 16] and its original DSIFT descriptor compared with the SLS descriptor described here. Results in the *bottom two rows* should appear similar to the *top left*, “Source” image. DSIFT descriptors provide some scale invariance despite the use of a single arbitrarily selected scale (*left column, middle row*). SLS provides scale invariance across far larger scale variations (*bottom*)

These features are then matched between images, taking advantage of the robustness of SIFT but making no attempt to provide additional scale invariance. Matching is performed using a modified optical flow formulation [7]. Although DSIFT provides some scale invariance, this quickly fails when scale differences increase (Fig. 2).

More recently, the Scale-Space SIFT flow of [24] proposes to augment SIFT flow by adding an optimization term reflecting scale differences between the two images. This method is described in more detail later in the book. Finally, a different approach altogether was recently proposed by Tau and Hassner [27]. Rather than attempting to estimate scale independently for each pixel, they propose scale propagation from the few image pixels where scale can be reliably established, to all other pixels in the image.

3 SIFTS in Scales

We begin by exploring the behavior of SIFT descriptors when extracted in the same image location but at multiple scales. The scale space $L(x, y, \sigma)$ of an image $I(x, y)$ is typically defined by the convolution of image $I(x, y)$ with the variable scale Gaussian $G(x, y, \sigma)$ [12], where

$$L(x, y, \sigma) = G(x, y, \sigma) \star I(x, y)$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

In many implementations (Sect. 2), a feature detector is first used to select coordinates in space x, y , and scale σ . A single SIFT descriptor $h_\sigma = h(x, y, \sigma)$ is then extracted at these space/scale coordinates [17]. Although in some cases more than one scale is selected for the same spatial location, such occurrences are typically treated independently of each other.

We consider all of the descriptors $h_{\sigma_i} = h(x, y, \sigma_i)$ extracted at the same spatial location. Here, σ_i is taken from a discrete set of scales $\{\sigma_1, \dots, \sigma_k\}$. Our key assumption is that corresponding pixels exhibit similar changes in appearances in different scales. That is, similar SIFT descriptors $h(x, y, \sigma_i)$ would be extracted from different scales of corresponding pixels. The challenge is then to effectively represent these similar SIFT descriptors across scales.

3.1 Multiscale SIFT Sets

Instead of making a single scale selection at each pixel, we extract descriptors from multiple scales. A pixel is then represented using a set of such SIFT descriptors. Formally, denote by p and p' a pair of corresponding pixels in images I and I' , respectively. Given a set of scales $\sigma_1, \dots, \sigma_k$, these pixels are represented by the sets $H = [h_{\sigma_1}, \dots, h_{\sigma_k}]$ and $H' = [h'_{\sigma_1}, \dots, h'_{\sigma_k}]$.

Determining the similarity of p and p' is performed using a set-to-set similarity. Several such measures of set similarity exist and some are surveyed in [31]. We later show that accurate matches are obtained when using the straightforward “min-dist” measure [31], defined by

$$\text{mindist}(p, p') = \min_{i,j} \text{dist}(h_{\sigma_i}, h'_{\sigma_j}) \quad (1)$$

Computationally, two pixels represented as n SIFT descriptors would require $O(128 \times n^2)$ operations to compare. This may be prohibitive if the sets are large. In practice, only few such scales are required to produce accurate representations (Sect. 5). To understand why, we make the following assumption:

Assumption 1 (Corresponding Points Have Similar SIFT Descriptors at Multiple Scales). We assume that there exists a set of scales $\sigma_1, \dots, \sigma_k$ for image I and a corresponding set of scales $\sigma'_1, \dots, \sigma'_k$ for image I' , such that the descriptors extracted at the two pixels in these corresponding scales are sufficiently similar: $h_{\sigma_i} = h'_{\sigma'_i}$. Let $H = [h_{\sigma_1}, \dots, h_{\sigma_k}]$ and $H' = [h'_{\sigma'_1}, \dots, h'_{\sigma'_k}]$, then we can write $H = H'$.

Realistically, this equality is expected to hold only when the scales $\sigma_1, \dots, \sigma_k$ and $\sigma'_1, \dots, \sigma'_k$ exactly correspond. In practice, these scale correspondences are unknown and the scales at each pixel are sampled at fixed intervals. The set of scales used to extract descriptors for a pixel in one image may be interleaved with those used for its corresponding pixel in the other image. SIFT values, however, change gradually with scale. Hence, only few scales need to be sampled in order to provide sufficiently

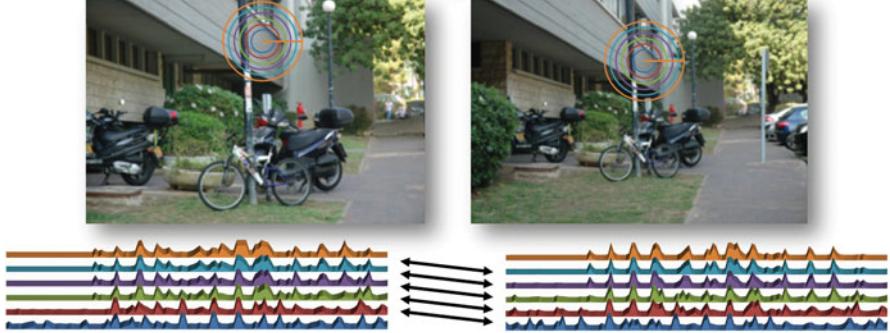


Fig. 3 SIFT values through scales. *Top:* two images at $\times 2$ scale factor difference. SIFTs were extracted at a low contrast region where interest points were *not* detected, at scales ranging from 10 to 35. *Bottom:* SIFT descriptor values visualized as histograms showing that (a) SIFTs from the *left image* match those at lower scales in the *right*, implying that setting the same scale to all pixels in both images may cause poor matches. (b) Even in the low contrast region studied here, SIFT values are far from uniform. (c) SIFT values appear to change gradually through scales

similar descriptor values, even in such cases. This is illustrated in Fig. 3. It shows SIFT values computed at multiple scales of two images separated by a $\times 2$ scale factor. SIFTs in the Target image clearly match the SIFTs in the Source image by a scale offset.

3.2 Multiscale SIFT Subspaces

An alternative, geometric representation for SIFT sets is obtained by considering the linear subspace spanned by these multiscale SIFTs. Subspaces are often used to represent varying information. Some recent examples from computer vision are listed in [3, 4]. Here, we examine the suitability of using subspaces to represent multiscale SIFTs extracted at a single image location. We begin with the following assumption.

Assumption 2 (SIFT Descriptors from Multiple Scales of the Same Image Point Span a Linear Subspace). SIFT descriptors are essentially gradient histograms. Often, local statistics of such gradients are equivalent at different scales. One example is homogeneous, low contrast regions or areas of stationary textures. In such cases the size of the local neighborhood does not significantly impact gradient distributions. We then get that $h_{\sigma_i} = h_{\sigma_j}$ for $\sigma_i \neq \sigma_j$.

In other cases, gradient statistics do change with the scale. However, if scales are sampled densely enough, these changes should be gradual and monotonic (Figs. 3 and 4). We get that for these pixels

$$h_{\sigma_i} = \sum_j w_{ij} h_{\sigma_j} \quad (2)$$

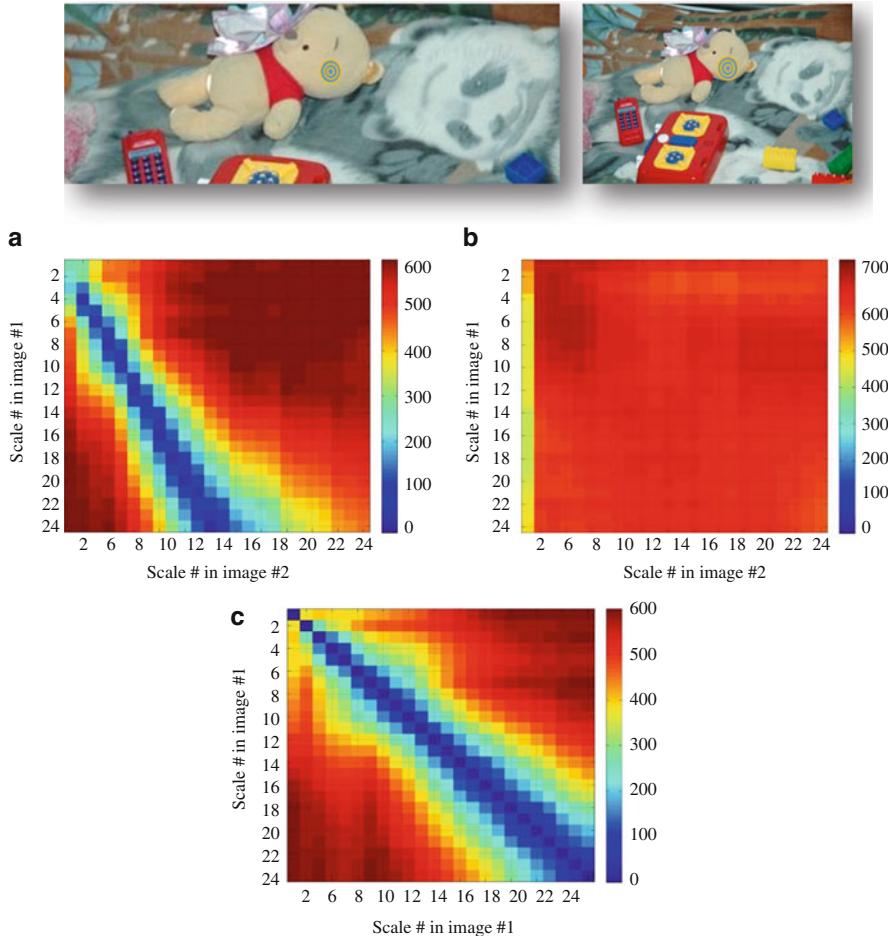


Fig. 4 Distances between SIFTs at different scales. *Top:* two images in different scales. SIFT descriptors are extracted from corresponding pixels in a low contrast region where interest points were *not* detected, using a range of 24 scales. *Bottom:* distances (*color coded*) between SIFTs in different scales from one image and SIFTs of different scales in the other image. In **(a)** distances are measured between ground-truth corresponding points. Clearly, SIFTs from the left image match descriptors on the right image, extracted at larger scales. Distances between SIFT descriptors extracted at a corresponding point but at mismatching scales result in very high distances, implying that the descriptors capture very different information. **(b)** Distances between non-corresponding points. Distances here are substantially greater than those for corresponding points at mismatched scales. **(c)** Distances between SIFTs extracted at different scales and other SIFTs extracted at the same point in the same image, but at different scales. Distances here grow gradually as the scale gap increases

where $w_{ij} = 0$ when h_{σ_i} does not depend on h_{σ_j} and $w_{ij} = \text{scalar}$ otherwise. That is, descriptors can be represented as linear combinations of other descriptors from the same pixel but at different scales. This happens when the regions surrounding the patch are piecewise stationary. Enlarging the window size by small steps maintains similar statistics within each window.

These two observations imply that a descriptor set $h_{\sigma_1}, \dots, h_{\sigma_k}$ approximately lies on a linear subspace:

$$H = [h_{\sigma_1}, \dots, h_{\sigma_k}] = [\hat{h}_1, \dots, \hat{h}_b] W = \hat{H}W \quad (3)$$

where $\hat{h}_1, \dots, \hat{h}_b$ are basis vectors spanning the space of descriptors from multiple scales and W is a matrix of coefficients.

Analysis: Mean Distances from SIFTs to Subspaces We evaluate our observations by considering distances from SIFT descriptors to these subspace representations. Effectively matching pixels requires that their representations accurately capture visual information and that the similarity between these representations can be estimated reliably. Here, this implies that a descriptor must be significantly closer to its own subspace than any other subspace. We illustrate this in Fig. 5. It shows the distances from descriptors extracted at each pixel, to the linear substance produced using SIFTs from multiple scales. We take the mean distance over the multiscale SIFTs extracted at a pixel, to the subspace produced for that pixel and color code these distances, from small (cool colors) to large (warm colors).

The distance of each descriptor to *other* subspaces (subspaces produced from SIFTs at other pixel locations; not shown in Fig. 5) can be used for reference: The maximum Euclidean distance between a descriptor and its own subspace is ≈ 40 greater than the mean distance of any descriptor extracted from the same pixel at any other scale. By comparison, its distance to other subspaces is on average ≈ 300 greater. Put differently, the distance between a descriptor and its correct subspace is substantially smaller than any wrong subspace.

Also evident from Fig. 5 is that large image regions are homogeneous and would likely yield few interest point detections. These vast image regions are where subspaces match descriptors best (cool colors).

Combining Our Two Assumptions According to Assumption 1, given two corresponding pixels, if we knew the set of corresponding scales, we would have $H = H'$. This implies that the two sets of descriptors share the same spanning basis, or that $\hat{H} = \hat{H}'$. Although we do not know the scales needed to construct H and H' , Assumption 2 assures us that this is not crucial: so long as scales are sampled densely enough, the bases \hat{H} and \hat{H}' can be computed and the two pixels matched correctly.

We can now define the distance between a pair of pixels, p and p' , represented by linear subspaces \mathcal{H}_p and $\mathcal{H}_{p'}$. Let \hat{H} and \hat{H}' be matrices with orthonormal columns

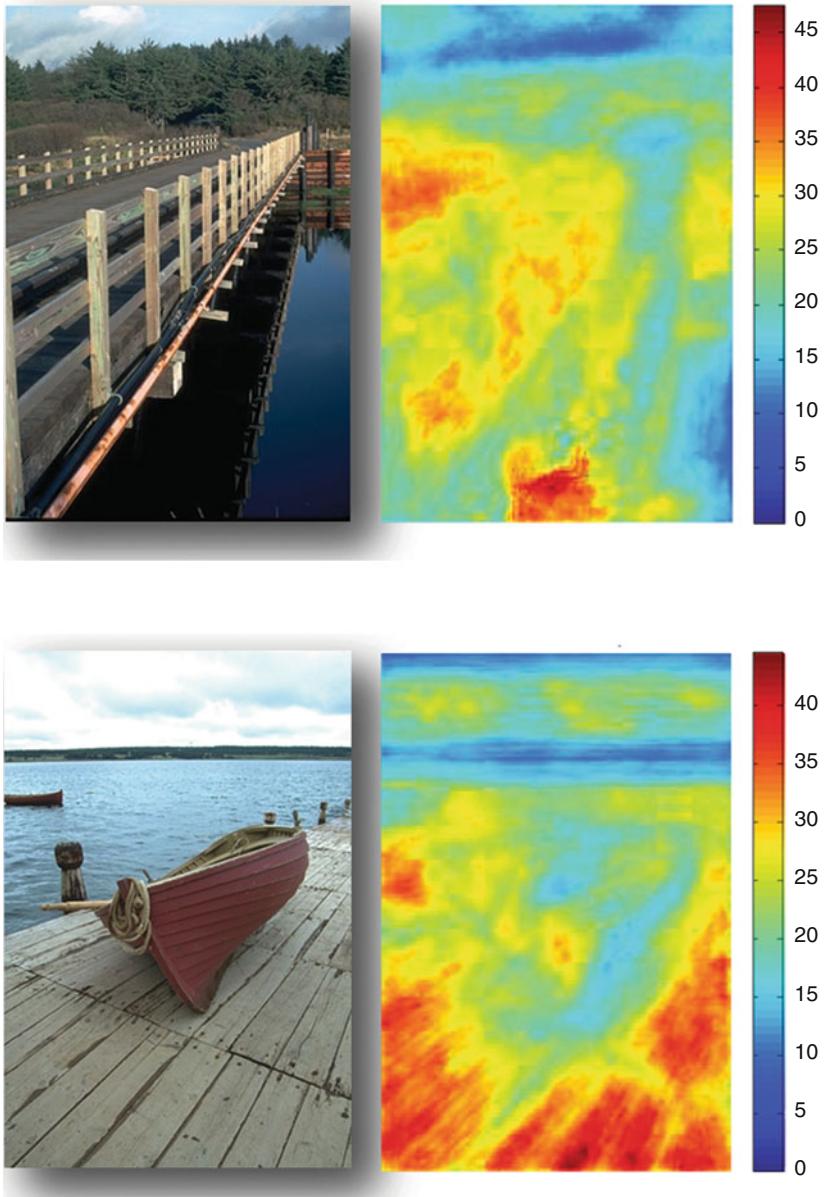


Fig. 5 Mean distances from SIFT descriptors to SIFT subspaces. Two examples showing on the left, a photo of a natural scene. Colors on the right visualize the mean distance between SIFT descriptors extracted at multiple scales of each pixel and the subspace they span. Low values (cool colors) imply that the subspace better fits its descriptors. Evident in both images is that subspaces fit best in homogenous regions. These regions make up most of the image, yet typically provide few reliable scale estimates by existing feature detection methods

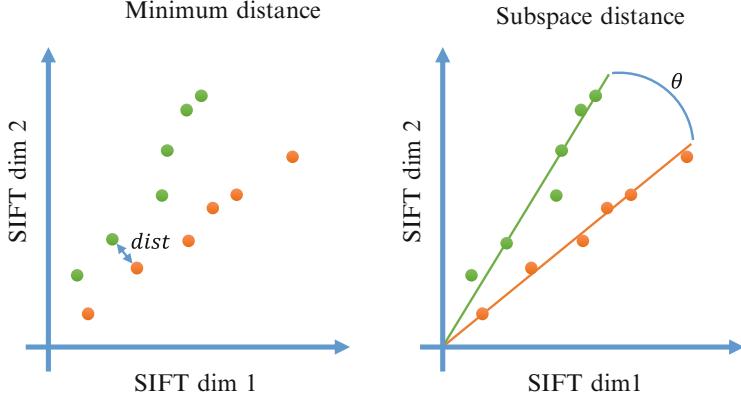


Fig. 6 Visualization in 2D: matching 2D SIFT descriptors from multiple scales. (*Left*) The min-dist of Eq.(1) computed as the minimal distance between any point in one set and a point in the other. (*Right*) Alternatively, we propose evaluating the distance between the subspaces spanned by SIFTS from multiple scales, using the Projection F-Norm [Eq.(4)]

representing the two subspaces. Various definitions exist for the distance between linear subspaces (for some examples, see [8]). We use the Projection Frobenius Norm (Projection F-Norm), defined by

$$\text{dist}^2(\mathcal{H}_p, \mathcal{H}_{p'}) = \|\sin \theta\|_2^2 \quad (4)$$

Here, $\sin \theta$ is a vector of the sines of the principal angles between subspaces \mathcal{H}_p and $\mathcal{H}_{p'}$. This value is computed from the cosines of the principal angles between the subspaces, obtained from $SVD(\hat{H}^T \hat{H}')$ in $O(128 \times d^2)$ operations, where d is the subspace dimension. This distance and its relation to the min-dist used for SIFT sets are visualized in Fig. 6.

3.3 A Scale-Less SIFT Representation

We have so far described a subspace representation for pixels. Many applications, however, are designed for *point* representations. These include systems that use efficient indexing techniques or, the subject of this book, dense correspondence estimation methods. We next show how the subspaces described in the previous section may be converted to points in a high dimensional space. Specifically, we use the subspace-to-point mapping proposed by Basri et al. [2–4]. We call the point representation obtained by mapping the multiscale subspace to a point, the SLS descriptor.

Consider the subspace \mathcal{H}_p computed for pixel p and represented as a $128 \times d$ matrix \hat{H} with orthonormal columns. Our SLS representation is produced by mapping this subspace to a point P by rearranging the elements of the projection matrix

$$A = \hat{H}\hat{H}^T \quad (5)$$

using the operator defined by

$$P \triangleq SLS(\hat{H}_p) = \left(\frac{a_{11}}{\sqrt{2}}, a_{12}, \dots, a_{1d}, \frac{a_{22}}{\sqrt{2}}, a_{23}, \dots, \frac{a_{dd}}{\sqrt{2}} \right)^T \quad (6)$$

where a_{ij} is the element (i, j) in matrix A . As shown by Basri et al. [3, 4], following this mapping, the distance between two mapped subspaces P and P' is monotonic with respect to the Projection F-Norm between the original subspaces \mathcal{H}_p and $\mathcal{H}_{p'}$, or formally

$$\|P - P'\|^2 = \mu \text{dist}^2(\mathcal{H}_p, \mathcal{H}_{p'}) \quad (7)$$

for a constant μ . The point P (our SLS descriptor) therefore captures the variability of SIFT descriptors extracted at a single image location but at multiple scales. This comes, however, at a quadratic cost in the dimension of the descriptors. The SLS descriptor, P , is used as an alternative representation for the subspace \mathcal{H}_p , making no other modifications to the underlying correspondence estimation method.

4 Region of Interest Segmentation

A particular aspect of dense correspondence estimation when images are at extremely different scales is that one image may present information which was not included in the field of view of the other. In such cases, when the Source image has the wider field of view, many of its pixels may either not have matches in the Target, or, depending on the correspondence estimation method used, may be matched to arbitrary, erroneous Target pixels. This introduces the problem of determining the region of interest (ROI) in the source image: the region of the image captured in the narrower field of view of the target image. In the past, this problem has often been avoided by assuming that the high-resolution image is neatly cropped, and we know exactly which parts of the scene appear in both images (see, e.g., [25]). Here, we automatically detect what parts are common to both views, using the estimated correspondences themselves.

Selecting the ROI is performed by seeking regions of high confidence correspondences; regions which appear to be present in both images. Given images I and I' , we compute the dense correspondences from Source I to Target I' and then,

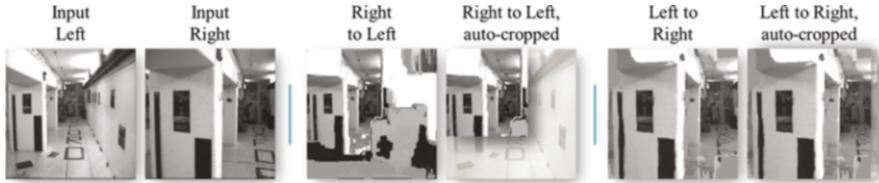


Fig. 7 Region of interest segmentation. Dense correspondences estimated without Epipolar Geometry, between the extreme two images in the Oxford Corridor sequence [9] (*left* and *right* input images). Results of warping the *right image* onto the *left* (*mid*) and the *left image* onto the *right* (*right*) are shown. Note the large regions where no information is available in the right input image to correspond with the *left image*, evident when warping the *right image* onto the *left*. These regions are automatically cropped, as described in Sect. 4)

separately, the correspondences from I' acting as the Source to I as the target. In both cases, we accumulate at each Target pixel the number of Source image pixels which were matched to it. A threshold is applied to these numbers, marking Target pixels with a high number of matching Source pixels. The resulting binary image is then processed with morphological operators to remove small pixel clusters. The ROI of image I is selected to be the bounding box of the remaining Target pixels, obtained by warping image I' . A similar process is performed by switching the roles of I and I' . Results of this approach are demonstrated in Fig. 7. We did not optimize this process and applied it without modification to our images.

5 Experiments

We implemented the methods described in this chapter in MATLAB, using the SIFT code of [30]. We compare with the SID implementation of [11]. Dense correspondences were estimated using the original SIFT flow code [15, 16], varying the per-pixel representation in order to compare the following: the original DSIFT, SID, and our own SLS descriptor. The SLS representation used in all our experiments, was computed using 8D, linear subspaces obtained by standard PCA from the scale-varying SIFT descriptors at each pixel. We used 20 scales for this purpose, linearly distributed in the range [0.5, 12]. Note that the dimension of our SLS descriptor and the matching time only depends on the dimension of the underlying representation, here 128D of the SIFT descriptors (Sect. 3.3). The code used in our experiments is publicly available [10].

Table 1 Quantitative results on rescaled Middlebury data [1].

Data	Angular error			Endpoint error		
	DISIFT	SID	SLS	DISIFT	SID	SLS
Dimetrodon	26.6±36.8	0.16±0.3	0.17±0.5	108.9±42.1	0.70±0.3	0.80±0.4
Grove2	7.06±5.7	0.66±4.4	0.15±0.3	59.07±40.9	1.50±5.0	0.77±0.4
Grove3	5.23±4.2	1.62±6.9	0.15±0.4	108.95±76.5	4.48±10.5	0.87±0.4
Hydrangea	4.24±4.5	0.32±0.6	0.22±0.8	33.80±32.2	1.59±2.8	0.91±1.1
Rubber Whale	24.63±26.9	0.16±0.3	0.15±0.3	116.83±57.7	0.73±1.1	0.80±0.4
Urban2	6.24±7.6	0.37±2.7	0.32±1.3	54.8±54.0	1.33±3.8	1.51±5.4
Urban3	10.26±14.0	0.27±0.6	0.35±0.9	91.81±66.1	1.55±3.7	9.41±24.6
Venus	4.30±4.9	0.24±0.6	0.23±0.5	31.52±34.0	1.16±3.8	0.74±0.3

Both angular and endpoint errors (\pm SD) show that multiple scales (SID [11] & SLS) are always advantageous over a single scale (DISIFT [30]) with SLS outperforming SID on most data sets. Bold values are the best results for each image pair

5.1 Quantitative Results on Middlebury Data

We compare our SLS with both SID and DSIFT, on the Middlebury optical flow set [1]. Since the pairs in this benchmark were produced from very close viewpoints, they exhibit little or no scale changes. We therefore modify the images, rescaling Source and Target images by factors of 0.7 and 0.2, respectively. We used standard means of quantifying the accuracy of the estimated flows, namely, the angular and endpoint errors (\pm SD) [1]. Results are reported in Table 1. Evidently, the multiscale approaches outperform single scale DSIFT by a substantial margin. SLS further improves over the multiscale SID representation of [11] in most cases.

5.2 Qualitative, Image Hallucination Results

We provide a visual comparison of the dense correspondences obtained using each of the following pixel representations: DSIFT, SID, and our SLS descriptor. Results are visualized by image hallucination: flow is computed from a Source to a Target image. Target image appearances (colors) are then warped back onto the source. Good results should present the Source image scene with Target image colors.

We select results demonstrating a wide variety of correspondence estimation tasks and challenges. Figure 8 provides results with pairs of images from the same scene, showing independent, deformable scene motion. Figures 9 and 1 include pairs of images taken from different scenes with similar content. These images include scale differences which are often quite extreme.

Figure 8 demonstrates that DSIFT typically manages to capture single scale information quite well, but cannot handle other scale changes in the scene. By comparison, the explicit representation of information from multiple scales allows

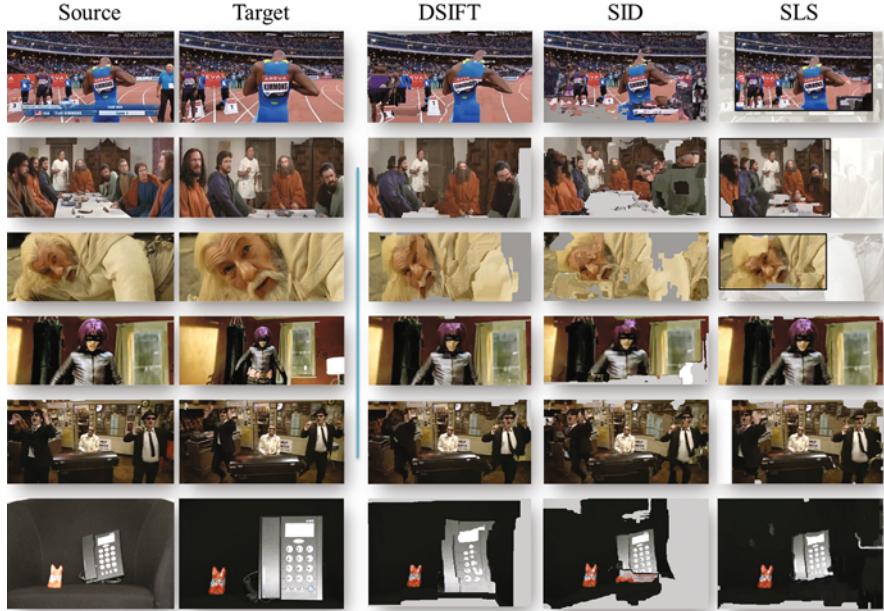


Fig. 8 Dense flow with scene motion. Image pairs presenting different scale changes in different parts of the scene, due to camera and scene motion. *From left to right*, the Source and Target input images, image hallucination results showing the Target image warped onto the Source using the estimated flow. Good results should present the Source image scene displayed in the Target image intensities. Correspondences estimated using [15], comparing the original, single scale DSIFT [30], SID [11] and our SLS, shown here cropped to regions corresponding to the scene visible in the Target image. See text for details

our SLS descriptor to better match more image regions. Moreover, the use of SIFT as the underlying representation provides a clear robustness to changing viewing conditions and scenes compared to the SID representation, which was originally designed for same scene, stereo applications (e.g., Fig. 9).

In Fig. 8 DSIFT typically manages to lock onto a single scale quite well, while missing other scale changes in the scene. The SLS descriptor better captures the scale-varying behavior at each pixel and so manages to better match pixels at different scales with only local misalignments.

5.3 Parameter Evaluation

We next evaluate how the various parameters used in defining the SLS descriptor influence its accuracy and matching time. To this end, we use images from the Berkeley collection [18]. Dense correspondences are computed between two copies

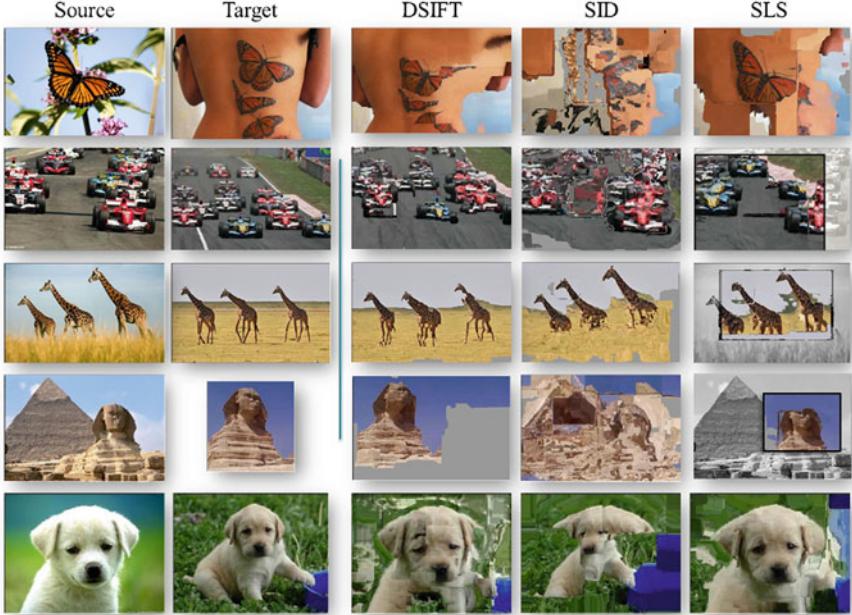


Fig. 9 Dense flow between different scenes. Image pairs presenting different scenes with similar content appearing in different scales. *From left to right*, the Source and Target input images, image hallucination results showing the Target image warped onto the Source using the estimated flow. Good results should present the Source image scene displayed in the Target image intensities. Correspondences estimated using [15], comparing the original, single scale DSIFT [30], SID [11], and our SLS, shown here cropped to regions corresponding to the scene visible in the Target image. See text for details

of each image: one copy in its original size and the other copy rescaled by a randomly determined factor uniformly distributed in the range $[1.5 \dots 4]$. We measure the mean \pm SD for both accuracy and runtime, when estimating dense correspondences between each image pair. Accuracy is measured as the ratio of times a pixel was matched correctly to the total number of pixels. Runtime measures the time required for matching.

Figure 10 presents the following results:

- 1. Point-to-point with scale selection:** A single scale is estimated using standard scale estimation techniques, applied here on a dense grid. Scale selection follows [17], by choosing the extremum value of the difference of Gaussians evaluated over multiple scales. In order to obtain scales for all pixels, we do not apply additional filtering.
- 2. Set-to-set, variable number of scales:** Using the min-dist similarity measure [Eq. (1)] to evaluate pixel similarities. We tested variable numbers of sampled scales: One to ten DSIFT descriptors were extracted from scales distributed linearly in the range of $[0.5, 12]$.

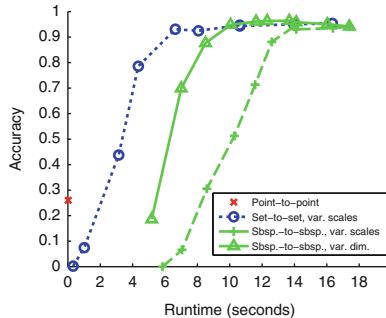


Fig. 10 Accuracy vs. runtime. Please see text for more details

3. **Subspace-to-subspace, variable number of scales:** Using the same sets as in (2) to fit linear subspaces at each pixel. Subspace dimensions equal the number of sampled scales. Distances between subspaces were computed using the Projection F-Norm of Eq. (4).
4. **Subspace-to-subspace, variable dimension:** Same as (3), but here ten DSIFT descriptors were used to fit subspaces, where the dimensions varied from 1D to 10D.

Evident in Fig. 10 is that when few scales are sampled, carefully selecting a single scale provides better performance than choosing an arbitrary scale. This advantage disappears at three scales, accuracy continuing to increase as more scales are sampled. By five scales, the quality of the matches reaches near perfect for all multiscale representations.

The accuracy of the subspace-to-subspace method testifies that linear subspaces accurately capture the variability of SIFT values through scale. In fact, this happens with as little as a 4D linear subspace. Notice that the use of a single scale is equivalent to comparing DSIFT descriptors using an arbitrary scale. The subspace-to-subspace distance further reduces to a sine similarity of two DSIFT descriptors. Both options are much worse than choosing the single scale at each pixel.

Runtimes for the set-based methods are naturally higher than those measured for comparing single points. No attempt was made to optimize the MATLAB code used in our experiments, and so these runtimes may conceivably be improved. The complexity of directly comparing two sets (Sect. 3.1) or two subspaces (Sect. 3.2), however, sets a limit on how much this performance can be improved. Our analysis here demonstrates the significance of the added computational burden compared to the gain in accuracy.

Figure 11 presents a few examples of the images used in these experiments, using pairs where one image is scaled to half the size of its counterpart. Flow vectors are visualized on the Target images. These demonstrate the accuracy for the following three representations: point-to-point correspondences of SIFT descriptors extracted on a dense grid using the standard DoG-based scale selection method of [17] [(1) in Fig. 10]; the subspace-to-subspace distance [Eq. (7)] between 4D subspaces

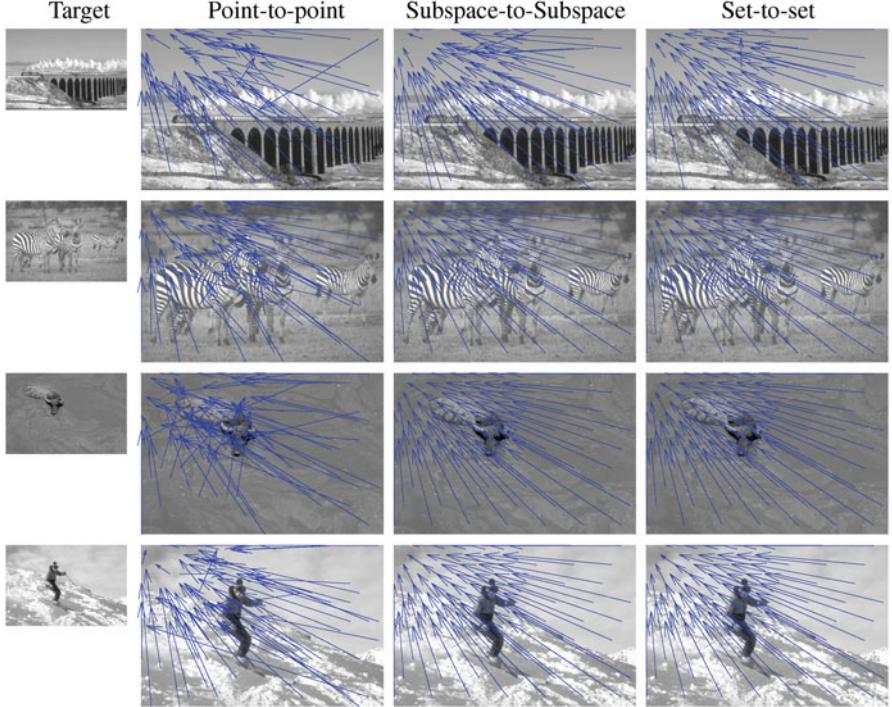


Fig. 11 Visualization of parameter evaluation tests. Image results from the Berkeley set [18] quantitative tests (Sect. 5.3). Target images are shown in the *left column*, scaled to half the size of the Source images on the *right*. Please see text for more details

produced by sampling ten scales linearly distributed in the range of $\sigma = [0.5, 12]$ [(3) in Fig. 10]; finally, min-dist [Eq. (4)] with set representations, using five scales samples linearly in the same range as the one used to produce the subspaces [(4) in Fig. 10]. Both multiscale representations clearly provide better correspondences than single descriptors. This is particularly true in low contrast areas where interest points are not typically detected.

6 Conclusion

This chapter aims to extend dense correspondence estimation methods beyond settings where both images contain the same scene, viewed at the same scale. We discuss how various per-pixel representation methods handle (or ignore) scale invariance. Specifically, early scale selection methods, going back to the early 1990s, were at least partially motivated by a need to reduce computational cost as well as by the assumption that information from only a few scales and a few image

locations can be matched reliably [13]. By contrast, we show that images contain valuable information in *multiple* scales. In selecting a single scale at sparse image locations, these earlier methods may be discarding information vital for establishing accurate dense correspondences.

We discuss the alternative approach of extracting SIFT descriptors from multiple scales. We show that this significantly improves correspondence accuracy but at a computational price. We examine several means of representing and comparing such SIFT sets and show that regardless of how they are used, they outperform the use of SIFTS extracted at single scales, with wide margins. Finally, we show that a subspace computed from SIFTS extracted at multiple scales at a single image location can be represented as a high-dimensional point. This, SLS representation is used as a stand-in for DSIFT in the SIFT flow method and shown to dramatically extend the range of settings in which dense correspondences may be estimated.

This chapter focuses on SIFT descriptors due to their popularity and their robustness to changing viewing conditions. A similar exploration may indeed result in similar Scale-Less representations, based on other popular local image descriptors. Scale-less DAISY [28], SURF [5], GLOH [21], or others would presumably be constructed using similar subspace representations and mapping. Another interesting direction left for future research is extending the work done here beyond scale, to other local affine transformations. The following chapters describe other approaches to scale invariance and applications of dense correspondence estimation which would benefit from such robust capabilities.

References

1. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., Szeliski, R.: A database and evaluation methodology for optical flow. *Int. J. Comput. Vis.* **92**(1), 1–31 (2001)
2. Basri, R., Hassner, T., Zelnik-Manor, L.: Approximate nearest subspace search with applications to pattern recognition. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (2007)
3. Basri, R., Hassner, T., Zelnik-Manor, L.: A general framework for approximate nearest subspace search. In: Proceedings of the International Conference on Computer Vision Workshop, pp. 109–116. IEEE, New York (2009)
4. Basri, R., Hassner, T., Zelnik-Manor, L.: Approximate nearest subspace search. *Trans. Pattern Anal. Mach. Intell.* **33**(2), 266–278 (2010)
5. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). *Comput. Vis. Image Underst.* **110**(3), 346–359 (2008)
6. Brox, T., Bregler, C., Malik, J.: Large displacement optical flow. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 41–48 (2009)
7. Bruhn, A., Weickert, J., Schnörr, C.: Lucas/kanade meets horn/schunck: combining local and global optic flow methods. *Int. J. Comput. Vis.* **61**(3), 211–231 (2005)
8. Edelman, A., Arias, T., Smith, S.: The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.* **20**, 303–353 (1998)
9. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, Cambridge (2004) [ISBN: 0521540518]

10. Hassner, T., Mayzels, V., Zelnik-Manor, L.: On SIFTs and their scales. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 1522–1528. IEEE, New York. <http://www.openu.ac.il/home/hassner/projects/siftscales> (2012)
11. Kokkinos, I., Yuille, A.: Scale invariance without scale selection. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 1–8. Available: <vision.mas.ecp.fr/Personnel/iasonas/code/distribution.zip> (2008)
12. Lindeberg, T.: Scale-space theory: A basic tool for analysing structures at different scales. *J. App. Stat.* **21**(2), 225–270 (1994)
13. Lindeberg, T.: Feature detection with automatic scale selection. *Int. J. Comput. Vis.* **30**(2), 79–116 (1998)
14. Lindeberg, T.: Principles for automatic scale selection. *Handb. Comput. Vis. Appl.* **2**, 239–274 (1999)
15. Liu, C., Yuen, J., Torralba, A., Sivic, J., Freeman, W.: SIFT flow: dense correspondence across different scenes. In: European Conference on Computer Vision, pp. 28–42. <people.csail.mit.edu/celiu/ECCV2008/> (2008)
16. Liu, C., Yuen, J., Torralba, A.: SIFT flow: dense correspondence across scenes and its applications. *Trans. Pattern Anal. Mach. Intell.* **33**(5), 978–994 (2011)
17. Lowe, D.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **60**(2), 91–110 (2004)
18. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proceedings of the International Conference on Computer Vision , vol. 2, pp. 416–423 (2001)
19. Mikolajczyk, K.: Detection of Local Features Invariant to Affine Transformations. Ph.D. thesis, Institut National Polytechnique de Grenoble, Grenoble (2002)
20. Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. *Int. J. Comput. Vis.* **60**(1), 63–86 (2004)
21. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. *Trans. Pattern Anal. Mach. Intell.* **27**(10), 1615–1630 (2005)
22. Morel, J., Yu, G.: Is sift scale invariant? *Inverse Problems Imaging (IPI)* **5**(1), 115–136 (2011)
23. Nowak, E., Jurie, F., Triggs, B.: Sampling strategies for bag-of-features image classification. In: European Conference on Computer Vision, pp. 490–503 (2006)
24. Qiu, W., Wang, X., Bai, X., Yuille, A., Tu, Z.: Scale-space sift flow. In: Proceedings of the Winter Conference on Applications of Computer Vision. IEEE, New York (2014)
25. Simon, I., Seitz, S.: A probabilistic model for object recognition, segmentation, and non-rigid correspondence. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 1–7 (2007)
26. Strecha, C., Tuytelaars, T., Gool, L.: Dense matching of multiple wide-baseline views. In: Proceedings of the International Conference on Computer Vision (2003)
27. Tau, M., Hassner, T.: Dense correspondences across scenes and scales. *IEEE Trans. Pattern Anal. Mach. Intell.* **PP**(99), 1 (2015)
28. Tola, E., Lepetit, V., Fua, P.: Daisy: an efficient dense descriptor applied to wide-baseline stereo. *Trans. Pattern Anal. Mach. Intell.* **32**(5), 815–830 (2009)
29. Varma, M., Garg, R.: Locally invariant fractal features for statistical texture classification. In: Proceedings of the International Conference on Computer Vision , pp. 1–8 (2007)
30. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms. In: Proceedings of the International Conference on Multimedia, pp. 1469–1472. Available: <www.vlfeat.org/> (2010)
31. Wolf, L., Hassner, T., Maoz, I.: Face recognition in unconstrained videos with matched background similarity. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 529–534 (2011)
32. Yang, H., Lin, W.Y., Lu, J.: Daisy filter flow: A generalized discrete approach to dense correspondences. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. IEEE, New York (2014)
33. Yao, J., Cham, W.: 3D modeling and rendering from multiple wide-baseline images by match propagation. *Signal Process. Image Commun.* **21**(6), 506–518 (2006)

Scale-Space SIFT Flow

Weichao Qiu, Xinggang Wang, Xiang Bai, Alan Yuille, and Zhuowen Tu

Abstract The SIFT flow algorithm has been widely used for the image matching/registration task and it is particularly effective in handling image pairs from similar scenes but with different object configurations. The way in which the dense SIFT features are computed at a fixed scale in the SIFT flow method might however limit its capability of dealing with scenes having great scale changes. In this work, we propose a simple, intuitive, and effective approach, Scale-Space SIFT flow, to deal with the large object scale differences. We introduce a scale field to the SIFT flow function to automatically explore the scale changes. Our approach achieves a similar performance as the SIFT flow method for natural scenes but obtains significant improvement for the images with large scale differences. Compared with a recent method that addresses a similar problem, our approach shows its advantage being more effective and efficient.

1 Introduction

The invention of dense image features [4, 17] has significantly improved the performance of image matching algorithms and they have been widely adopted in various computer vision applications, such as scene parsing [15, 20], image/video retrieval [16], motion estimation [16], and depth estimation [21]. Compared to the previous approaches using extracted sparse features, e.g., interest points [2, 18], dense features are better to preserve the intrinsic uncertainties so that more robust

W. Qiu • X. Wang • X. Bai

Department of Electronics and Information Engineering, Huazhong

University of Science and Technology, Hubei, China

e-mail: qiuwch@gmail.com; xgwang@hust.edu.cn; xbai@hust.edu.cn

A. Yuille

University of California, Los Angeles, Los Angeles, CA, USA

e-mail: yuille@stat.ucla.edu

Z. Tu (✉)

University of California, San Diego, La Jolla, CA, USA

e-mail: ztu@ucsd.edu

decisions can be made in the later stages; in practice, significant improvements over the previous method in image matching and classification have been widely observed by SIFT flow [16] and spatial pyramid matching [9], respectively.

Sparse features, e.g., the SIFT points [17] Harris corners [18], can somewhat automatically (to a certain degree) determine the feature scale. They work well for matching identical objects in the images but fail to deliver reliable results for semantic image alignment in the more general cases. In matching, it is hard to determine the right scales unless both the source and target images are observed.

When applying the dense features, the existing methods [16, 17] mostly compute the features at a fixed scale. In this work, we study the image matching/correspondence/registration problem and focus on the SIFT flow algorithm [16]. One popular method for computing the dense features is the dense SIFT (DSIFT) descriptor which extracts SIFT histogram at a single scale for all pixels with overlapping patches. Using DSIFT, SIFT flow [16] aligns two images by minimizing matching cost and keeping the flow field smooth. Since the DSIFT feature is only computed at one scale in SIFT flow, it requires that objects in two images share the same/similar scales. This makes SIFT flow problematic in dealing with images of large scale change, which was observed in [6]. To overcome this problem, a recent method called Scale-Less SIFT (SLS) was proposed in [6]. When performing the matching, SLS extracts a set of DSIFT features at multiple scales for every pixel and uses set-to-set distance to measure the matching cost between the corresponding pixels. More specifically, SLS uses Projection Frobenius Norm to compute the set-to-set distance. In practice, if there are 100,000 pixels in each image then a locality constraint is needed; running SLS with the Projection Frobenius Norm is both time and memory consuming. For an image of standard size 640×480 , it consumes more than 10G memory and takes hours to perform one matching, which makes the SLS algorithm nearly impractical to scale up. In addition, SLS tries to address the scale difference problem by feature fusion. Instead, we tackle this problem by creating a scale field to automatically explore the best SIFT matches at the right scale at each location. This simple but intuitive solution yields significant improvement over SLS in performance, speed, and memory consumption; it also demonstrates its advantage over the original SIFT flow method when dealing with images of large scale differences.

The “scale-space” theory was first introduced by Witkin in [22] for multiscale image representation and is now considered as a well-studied topic [10, 23]. Linear Gaussian scale space is the most widely used multiscale image representation method. Anisotropic diffusion [19] was proposed to obtain non linear smooth scale space by Perona and Malik. Nevertheless, it is beneficial to look back at some of the early work in computer vision whose essences are often inspiring to the modern vision algorithms. In [7], a spatial pyramid matching algorithm is embedded in the SIFT flow method to better deal with the scale change problem.

In our approach, we associate each pixel with a scale factor; scale factors on the image lattice form a scale field; dense feature for each pixel is estimated for a

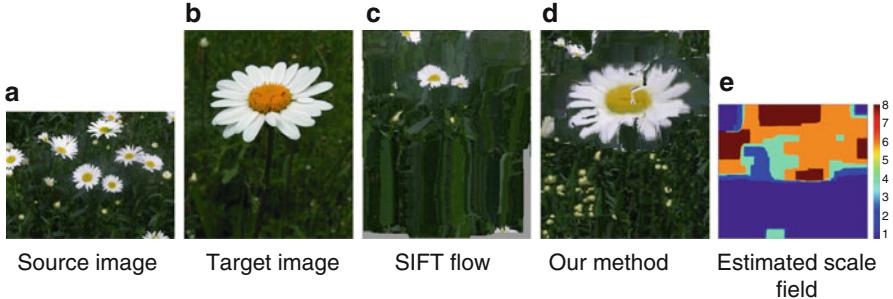


Fig. 1 We find dense correspondence from (a) to (b), then use dense correspondence to warp the source image to the target image. (c, d) are warped image from SIFT flow and our method, respectively. (e) shows the estimated scale field

particular scale during matching. The “best” match is estimated with the selection of the right scale factors through (1) minimizing feature matching cost, (2) keeping the flow field smooth, and (3) keeping the scale field smooth. Figure 1 shows the clear advantage of introducing the scale field of our method which naturally explores the scale space for matching image pairs of different scales. We give an iterative solution for minimizing the objective function.

2 Related Work

The related work can be roughly divided into categories on scale invariant image representations and those performing image matching/registration. To build scale invariant image representations, most methods are based on automatic scale selection which seeks for each feature point a stable and characteristic scale. Automatic scale selection methods usually search for local extrema in the 3D scale-space representation of an image (x , y , and scale). This idea was introduced in the early 1980s by Crowley [3]. Based on this idea, Laplacian of Gaussian (LoG) [11] and Difference of Gaussian (DoG) [17] are widely used. Please see [18] for a more comprehensive survey for automatic scale selection. In [8], a scale invariant image descriptor is built without scale selection. Scale invariance of SID is guaranteed in the Fourier Transform Modulus. There are a large number of image matching/alignment/registration methods in computer vision and medical imaging, etc. Some recent works include: the dense graph matching method in [12], the vector field learning method in [24], and factorized graph matching [25]. Our method is built upon the SIFT flow [13, 16] method, which does not consider the scale change problem.

3 Approach

3.1 SIFT Flow Review

We first review the SIFT flow formulation in [16]. In SIFT flow, it is assumed that image matching is performed without much scale change. Let $\mathbf{p} = (x, y)$ be the grid coordinate of images, $\mathbf{w}(\mathbf{p}) = (u(\mathbf{p}), v(\mathbf{p}))$ be the flow vector at \mathbf{p} , and s_1 and s_2 be two SIFT images that we want to match. $s_1(\mathbf{p})$ denotes a SIFT descriptor at position \mathbf{p} of SIFT image s_1 . Set ϵ contains all spatial neighborhoods (usually a four-neighbor system is used). The energy function for SIFT flow is defined as

$$\begin{aligned} E(\mathbf{w}) = & \sum_{\mathbf{p}} \min (\| s_1(\mathbf{p}) - s_2(\mathbf{p} + \mathbf{w}(\mathbf{p})) \|_1, t) \\ & + \sum_{\mathbf{p}} \eta (|u(\mathbf{p})| + |v(\mathbf{p})|) \\ & + \sum_{(\mathbf{p}, \mathbf{q}) \in \epsilon} \min(\alpha |\mathbf{w}(\mathbf{p}) - \mathbf{w}(\mathbf{q})|, d) \end{aligned} \quad (1)$$

which contains a *data term*, a *small displacement term*, and a *smoothness term* (a.k.a spatial regularization). In Eq. (1), $\| s_1(\mathbf{p}) - s_2(\mathbf{p} + \mathbf{w}(\mathbf{p})) \|_1$ is the *data term* which constrains the SIFT descriptors to be matched along with the flow vector $\mathbf{w}(\mathbf{p})$. $|u(\mathbf{p})| + |v(\mathbf{p})|$ is the *small displacement term* which constrains the flow vectors to be as small as possible. $|\mathbf{w}(\mathbf{p}) - \mathbf{w}(\mathbf{q})| = |u(\mathbf{p}) - u(\mathbf{q})| + |v(\mathbf{p}) - v(\mathbf{q})|$ is the *displacement smoothness term* which constrains the flow vectors of adjacent pixels to be similar. In this objective function, truncated ℓ_1 norms are used in both the data term and the displacement smoothness term to account for matching outliers and flow discontinues, with t and d as threshold, respectively.

In [16], the optimization problem in Eq. (1) was solved using a dual layer loopy belief propagation; a coarse-to-fine matching scheme is further adapted which can both speed up matching and obtain a better solution. There is no scale factor in Eq. (1), while in many dense feature matching applications, images are at different scales. In SIFT flow, dense SIFT feature computed in fixed grids and fixed patch size in image can handle small scale variation. From our experimental experience, SIFT flow can handle scale variation with the ratio in [1/1.5, 1.5]. In the following section, we propose a dense feature matching formulation by simply associating each pixel in image with a scale factor.

3.2 Scale-Space SIFT Flow

When matching two images, we keep the second image at its own scale. In the first image, we assign a scale factor for every position which is denoted by $\sigma(\mathbf{p}) \in \mathbf{R}$.

We denote $s_1(\mathbf{p}, \sigma(\mathbf{p}))$ as a SIFT feature computed at scale $\sigma(\mathbf{p})$ at position \mathbf{p} in the first image. For simplicity, we omit the scale factor if it is 1. Our new energy function is given as follows:

$$\begin{aligned} E(\mathbf{w}, \sigma) = & \sum_{\mathbf{p}} \min (||s_1(\mathbf{p}, \sigma(\mathbf{p})) - s_2(\mathbf{p} + \mathbf{w}(\mathbf{p}))||_1, t) \\ & + \sum_{\mathbf{p}, \mathbf{q} \in \epsilon} \min(\alpha |\mathbf{w}(\mathbf{p}) - \mathbf{w}(\mathbf{q})|, d) \\ & + \sum_{\mathbf{p}, \mathbf{q} \in \epsilon} \min(\beta |\sigma(\mathbf{p}) - \sigma(\mathbf{q})|, \tau) \end{aligned} \quad (2)$$

In Eq. (2), besides a *flow field* \mathbf{w} , we have another *scale field* σ . We require both fields to be smooth. In Eq. (2), $||\sigma(\mathbf{p}) - \sigma(\mathbf{q})||$ is the *scale smoothness term* which constrains the scale factors of adjacent pixels to be similar. Different from the energy function in SIFT flow, here we remove the small displacement term. This is due to the fact that the displacements are no longer small when objects of interest to be matched are at different scales. Another advantage of removing the small displacement term is that it allows objects of interest to appear in any position of the second image. This gives more flexibility than before. Parameters β and τ are used for adjusting the weight of scale smoothness term and truncating scale smoothness term. To deal with potential outliers in the scale field, we use ℓ_1 norm for scale smoothness term.

Our formulation gives a very natural and intrinsic approach to deal with the scale variation issue in dense feature matching. Rather than computing multiscale DSIFT descriptors and matching them using set-to-set distance in [6], we aim at computing feature at a proper scale. We allow different positions to have different scales and constrain the final scale field to be smooth. This setting is very reasonable.

3.3 Optimization for (2)

In this subsection, we give our optimization method for our energy function in (2). A straightforward solution is to put the scale factor $\sigma(\mathbf{p})$ in the dual layers loopy belief propagation framework. However, because the coarse-to-fine scheme needs to re-scale the images, it is not suitable for optimizing the scale factor. Alternatively, we propose an iterative solution. When the scale field σ is fixed, we optimize the flow field \mathbf{w} ; when the flow field \mathbf{w} is obtained, we optimize the scale field σ . To tackle the problem of optimizing scale field, we discrete the scale space into N states, denoted as $\zeta = \{\sigma_1, \sigma_2, \dots, \sigma_N\}$. The estimated scale factor for every position should be in the scale space, denoted as $\sigma(\mathbf{p}) \in \zeta$. To clearly illustrate our algorithm, we also define an index map $\mathbf{m} = \{m(\mathbf{p}) \in [1, \dots, N]\}$ for the scale field, thus $\sigma(\mathbf{p}) = \sigma_{m(\mathbf{p})}$.

Initialization: For $n \in [1, \dots, N]$, we compute flow field by solving:

$$\hat{\mathbf{w}}_n = \arg \min_{\mathbf{w}} \sum_{\mathbf{p}} \min \left(\|s_1(\mathbf{p}, \sigma_n) - s_2(\mathbf{p} + \mathbf{w}(\mathbf{p}))\|_1, t \right) + \sum_{\mathbf{p}, \mathbf{q} \in \mathcal{E}} \min(\alpha |\mathbf{w}(\mathbf{p}) - \mathbf{w}(\mathbf{q})|, d) \quad (3)$$

Then, we compute scale field by solving:

$$\begin{cases} \hat{\mathbf{m}} = \arg \min_{m(\mathbf{p}) \in [1, \dots, N]} \sum_{\mathbf{p}} \min \left(\|s_1(\mathbf{p}, \sigma_{m(\mathbf{p})}) - s_2(\mathbf{p} + \hat{\mathbf{w}}_m(\mathbf{p}))\|_1, t \right) + \sum_{\mathbf{p}, \mathbf{q} \in \mathcal{E}} \min(\beta |\sigma_{m(\mathbf{p})} - \sigma_{m(\mathbf{q})}|, \tau) \\ \hat{\sigma} = \{\sigma_{m(\hat{\mathbf{p}})}\} \end{cases} \quad (4)$$

Run the following two procedures for a desired number of iterations.

Optimize flow field \mathbf{w} :

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \sum_{\mathbf{p}} \min \left(\|s_1(\mathbf{p}, \sigma_{m(\hat{\mathbf{p}})}) - s_2(\mathbf{p} + \mathbf{w}(\mathbf{p}))\|_1, t \right) + \sum_{\mathbf{p}, \mathbf{q} \in \mathcal{E}} \min(\alpha |\mathbf{w}(\mathbf{p}) - \mathbf{w}(\mathbf{q})|, d) \quad (5)$$

Optimize scale field σ :

$$\begin{cases} \hat{\mathbf{m}} = \arg \min_{m(\mathbf{p}) \in [1, \dots, N]} \sum_{\mathbf{p}} \min \left(\|s_1(\mathbf{p}, \sigma_{m(\mathbf{p})}) - s_2(\mathbf{p} + \hat{\mathbf{w}}(\mathbf{p}))\|_1, t \right) + \sum_{\mathbf{p}, \mathbf{q} \in \mathcal{E}} \min(\beta |\sigma_{m(\mathbf{p})} - \sigma_{m(\mathbf{q})}|, \tau) \\ \hat{\sigma} = \{\sigma_{m(\hat{\mathbf{p}})}\} \end{cases} \quad (6)$$

Output: The estimated flow field $\mathbf{w}^* = \hat{\mathbf{w}}$ and scale field $\sigma^* = \hat{\sigma}$.

Fig. 2 Optimization algorithm for Eq. (2)

The whole solution is outlined in Fig. 2. For image 2, we always compute SIFT image at its original scale. In the initialization stage, for every scale in the scale space, we compute SIFT image for the first image at this scale, match the SIFT image to the SIFT image 2 using SIFT flow formulated in Eq. (3), and obtain the data term for every position at current scale. Then we initialize scale factor for every position by looking for smaller data term and keeping scale smooth at the same time; these two objectives are formulated together in a MRF framework given in Eq. (4). After the scale field is initialized, we iteratively optimize the flow field and the scale field. Now, the idea of optimizing flow field is fixing SIFT feature as $s_1(\mathbf{p}, \sigma_{m(\mathbf{p})})$ in the first image and computing SIFT flow; this is formulated in Eq. (5). The idea of optimizing scale field is fixing flow as $\hat{\mathbf{w}}(\mathbf{p})$ and looking for a smaller data term across scales and keeping the scale space smooth at the same time; this is formulated in Eq. (6).

The problems of optimizing flow field [in Eqs. (3) and (5)] and optimizing scale field [in Eqs. (4) and (6)] are easy to tackle using existing optimizing techniques. We directly use optimization method in SIFT flow [16] to optimize flow field.

We use the efficient belief propagation algorithm in [5] for scale field; coarse-to-fine scheme is not used when optimizing the scale field, since the scale space is very small compared to the displacement space.

4 Experiment

In our experiments, we report the image matching results by our method on benchmark data sets and compare with SIFT flow [16] and SLS [6], which are most related. For a quantitative evaluation, we use the standard Middlebury data set [1], which has been widely used as a benchmark for optical flow algorithms. Then we quantitatively and qualitatively compare the matching performance to [6] on the identical images used in [6].

Through our experiment, the scale space of the dense SIFT is set to $\zeta = \{1, 2, 4, 6, 8\}$; the weight parameters $\alpha = 3$, $\beta = 60$; the threshold for flow field $d = 60$; for the threshold of scale field $\tau = 120$; the threshold of data term t is set following [15]. We run our scale-space SIFT flow following our algorithm description in Fig. 2.

4.1 Quantitative Result

The original Middlebury data set [1] images are of identical scale. To perform image matching across different scales, we rescale images in this data set, following the protocol in [6]. Source images are scaled to 0.7 and target images are scaled to 0.2. The ground truths are also scaled accordingly. Quantitative image matching performance is measured by the angular error and the endpoint error which are the same as in [1]. Specifically, the angular error is the angle between estimated flow and ground truth. It is defined as $AE = \cos^{-1} \left(\frac{1.0 + u*u_{GT} + v*v_{GT}}{\sqrt{1.0 + u^2 + v^2} \sqrt{1.0 + u_{GT}^2 + v_{GT}^2}} \right)$. Endpoint error is defined as $EE = \sqrt{(u - u_{GT})^2 + (v - v_{GT})^2}$. Average values and standard deviations of both angular error and endpoint error ($\pm SD$) for three methods are reported in Table 1.

Table 1 shows that original SIFT flow fails to deal with large scale difference. Both SLS and our method can handle large scale difference and our method consistently outperforms SLS in both angular error and endpoint error measures.

We also test our method on the original Middlebury data set. From Table 2 it is obvious that our method can perform equally well as the original SIFT flow, which has been successfully applied to scene parsing [14].

Table 1 Quantitative results for SIFT flow [16], SLS [6], and our method on scaled version of Middlebury data set [1]

Data	Angular error			Endpoint error		
	SIFT flow [16]	SLS [6]	Our method	SIFT flow [16]	SLS [6]	Our method
Dimetrodon	25.99 ± 38.28	0.17 ± 0.42	0.13 ± 0.18	59.06 ± 37.98	0.83 ± 0.40	0.52 ± 0.27
Grove2	5.12 ± 12.89	0.15 ± 0.26	0.11 ± 0.23	25.89 ± 39.60	0.80 ± 0.36	0.48 ± 0.34
Grove3	5.73 ± 12.80	0.16 ± 0.32	0.12 ± 0.27	69.24 ± 69.84	0.91 ± 0.47	0.62 ± 0.83
Hydrangea	5.98 ± 13.82	0.18 ± 0.37	0.26 ± 1.77	24.52 ± 34.14	0.79 ± 0.40	0.63 ± 1.18
Rubber Whale	20.73 ± 30.05	0.15 ± 0.23	0.12 ± 0.17	63.29 ± 41.46	0.85 ± 0.50	0.52 ± 0.28
Urban2	6.24 ± 13.32	0.32 ± 1.28	0.14 ± 0.18	42.59 ± 55.57	1.53 ± 5.40	0.65 ± 0.53
Urban3	9.65 ± 14.98	0.66 ± 1.39	0.19 ± 0.37	58.25 ± 52.05	18.20 ± 35.11	0.79 ± 0.81
Venus	5.51 ± 15.21	0.23 ± 0.47	0.22 ± 0.55	15.94 ± 29.94	0.78 ± 0.39	0.62 ± 0.73

Table 2 Quantitative results of original SIFT flow [16] and our method on the original version of Middlebury dataset [1]

Data	Angular error		Endpoint error	
	SIFT flow [16]	Our method	SIFT flow [16]	Our method
Dimetrodon	9.82 \pm 8.55	10.71 \pm 11.24	0.43 \pm 0.28	0.47 \pm 0.37
Grove2	8.29 \pm 12.52	6.65 \pm 8.37	0.57 \pm 0.64	0.50 \pm 0.50
Grove3	12.44 \pm 19.18	10.69 \pm 17.13	1.10 \pm 1.66	0.98 \pm 1.51
Hydrangea	8.96 \pm 18.74	8.23 \pm 17.53	0.60 \pm 1.17	0.56 \pm 1.10
Rubber Whale	11.46 \pm 15.78	10.59 \pm 14.87	0.37 \pm 0.42	0.35 \pm 0.40
Urban2	10.77 \pm 17.40	8.34 \pm 12.29	1.51 \pm 3.57	0.77 \pm 1.74
Urban3	14.48 \pm 35.09	8.28 \pm 21.29	1.46 \pm 3.23	0.97 \pm 2.04
Venus	7.17 \pm 22.07	10.93 \pm 34.00	0.55 \pm 1.05	1.25 \pm 4.14

4.2 Qualitative Results

To further illustrate the ability of our method, some visual comparisons are shown in this section. For our method and other compared methods, we compute dense feature correspondences between the source image and the target image; the result image is the warped source image according to the computed dense correspondences.

At first, we show the matching results as the scale factor change in Fig. 3. As shown in Fig. 3, from left to right, the scale of the source image reduces gradually. The original SIFT flow can handle small scale variation, but will inevitably fail when scale differences become large. Both our method and SLS can effectively handle large scale variation.

We take the identical images presented in [6] for a fair comparison. Figure 4 shows the case that scale difference is caused by scene motion. As shown in the figure, the original SIFT flow method manages to lock onto single scale, which causes large distortion on the result image. Both SLS and our method can find correct correspondences between the source images and the target images. Since our method does not have a region of interest (ROI) strategy as used in SLS, the warped source image becomes more flexible in deformation.

Figure 5 shows the image matching results with scale difference in different scenes. For the flower and butterfly cases, our method can tackle scale difference to produce appealing results. For the racing car case, though each car in the source image found its correspondence on the target image, the road becomes more cluttered than the other method. So we consider it as a “failure” case.

4.2.1 Computational Cost

In this part, we compare computational time and memory cost needed by SLS and our method. According to our algorithm in Fig. 2, the computational time of our algorithm is roughly number of scale times execution time of original SIFT flow.

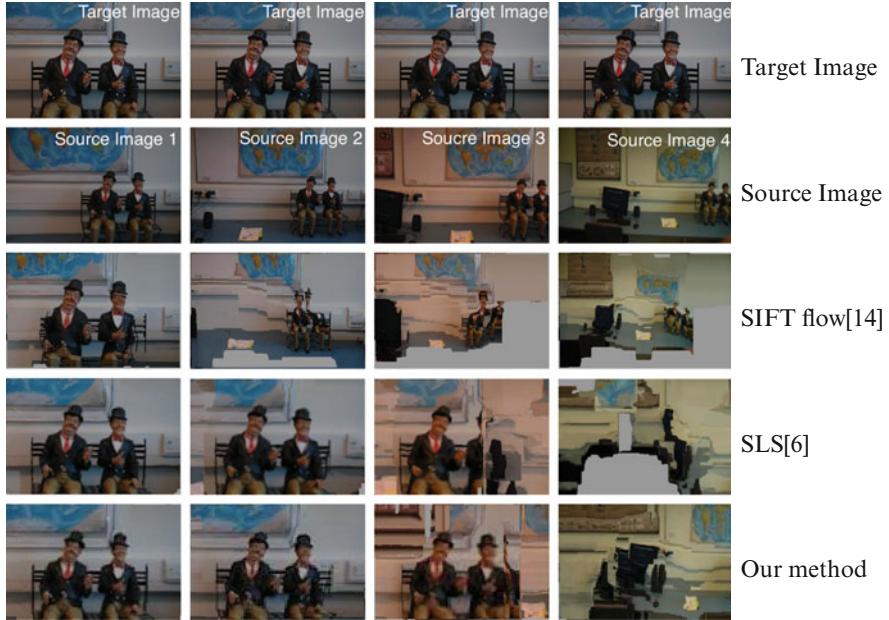


Fig. 3 Matching results of SIFT flow [16], SLS [6], and our method on different scales. SIFT flow can only handle small scale variance. Our method and SLS can tolerate much bigger scale difference

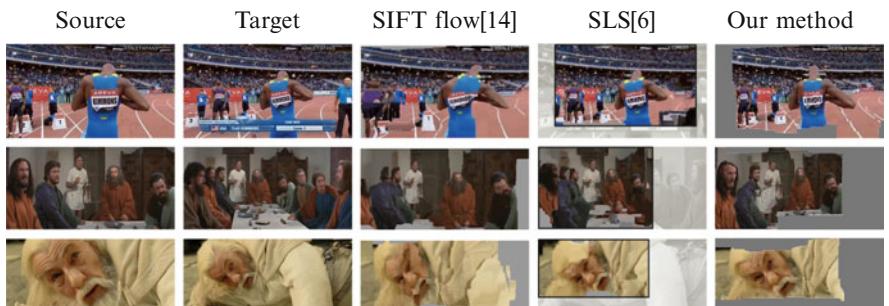


Fig. 4 Matching results of SIFT flow [16], SLS [6], and our method on different scale images caused by camera and scene motion

But in SLS, computing set-to-set distance for multiscale SIFTS is very time and memory consuming. We report average computational time and memory cost for matching one pair of images in the experiments in Sect. 4.1 in Table 3. In addition, our platform is a Xeon 3.07 GHz server with 64G memory.

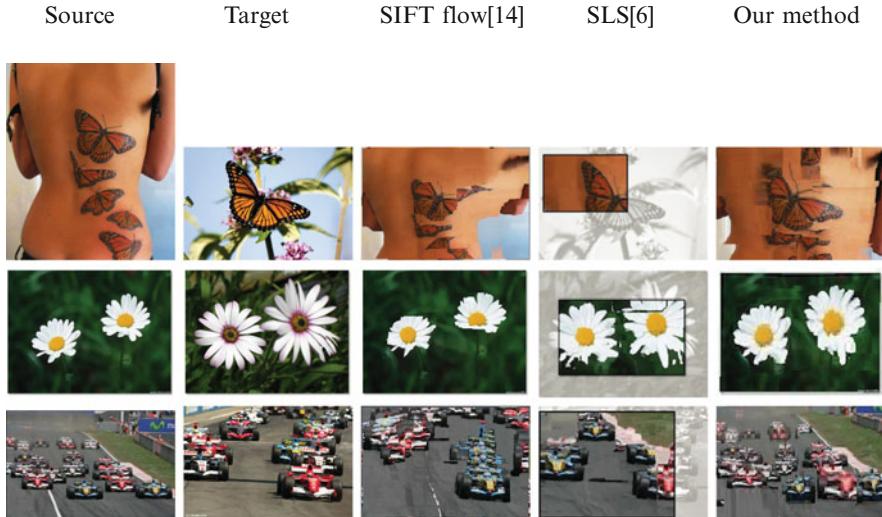


Fig. 5 Matching results of SIFT flow [16], SLS [6], and our method on multiscale images in different scenes

Table 3 Comparison of computational cost of SLS [6] and our method

	SLS [6]	Ours
Computational time	45 min	2 min
Memory cost	6G	1G

From those previous experiments, we observe that both SLS and the proposed method are able to cope with scale difference problem in matching. The significantly improved quantitative performance with much less computation burden illustrates the evident advantage of our method over SLS.

Acknowledgements This work is supported by the National Natural Science Foundation of China (grant No. 61222308) and the NSF awards IIS-1216528 (IIS-1360566), IIS-0844566 (IIS-1360568), and IIS-0917141.

References

1. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M.J., Szeliski, R.: A database and evaluation methodology for optical flow. *Int. J. Comput. Vis.* **92**(1), 1–31 (2011)
2. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. *Trans. Pattern Anal. Mach. Intell.* **24**(4), 509–522 (2002)
3. Crowley, J.: A Representation for Visual Information. Ph.D. thesis, Carnegie Mellon University (1981)

4. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 886–893. IEEE, New York (2005)
5. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient belief propagation for early vision. *Int. J. Comput. Vis.* **70**(1), 41–54 (2006)
6. Hassner, T., Mayzels, V., Zelnik-Manor, L.: On sifts and their scales. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 1522–1528. IEEE, New York (2012)
7. Kim, J., Liu, C., Sha, F., Grauman, K.: Deformable spatial pyramid matching for fast dense correspondences. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 2307–2314. IEEE, New York (2013)
8. Kokkinos, I., Yuille, A.: Scale invariance without scale selection. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE, New York (2008)
9. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 2169–2178. IEEE, New York (2006)
10. Lindeberg, T.: Scale-Space Theory in Computer Vision. Springer Science & Business Media, New York (1993)
11. Lindeberg, T.: Feature detection with automatic scale selection. *Int. J. Comput. Vis.* **30**(2), 79–116 (1998)
12. Liu, H., Yan, S.: Common visual pattern discovery via spatially coherent correspondences. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 1609–1616. IEEE, New York (2010)
13. Liu, C., Yuen, J., Torralba, A., Sivic, J., Freeman, W.T.: Sift flow: Dense correspondence across different scenes. In: European Conference on Computer Vision, pp. 28–42. Springer, New York (2008)
14. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing: label transfer via dense scene alignment. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 1972–1979. IEEE, New York (2009)
15. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing via label transfer. *Trans. Pattern Anal. Mach. Intell.* **33**(12), 2368–2382 (2011)
16. Liu, C., Yuen, J., Torralba, A.: Sift flow: dense correspondence across scenes and its applications. *Trans. Pattern Anal. Mach. Intell.* **33**(5), 978–994 (2011)
17. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **60**(2), 91–110 (2004)
18. Mikolajczyk, K., Schmid, C.: Scale & affine invariant interest point detectors. *Int. J. Comput. Vis.* **60**(1), 63–86 (2004)
19. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. *Trans. Pattern Anal. Mach. Intell.* **12**(7), 629–639 (1990)
20. Rubinstein, M., Liu, C., Freeman, W.T.: Annotation propagation in large image databases via dense image correspondence. In: European Conference on Computer Vision, pp. 85–99. Springer (2012)
21. Tola, E., Lepetit, V., Fua, P.: Daisy: An efficient dense descriptor applied to wide-baseline stereo. *Trans. Pattern Anal. Mach. Intell.* **32**(5), 815–830 (2010)
22. Witkin, A.: Scale space filtering. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1019–1022 (1983)
23. Yuille, A.L., Poggio, T.A.: Scaling theorems for zero crossings. *Trans. Pattern Anal. Mach. Intell.* **15**(1), 15–25 (1986)
24. Zhao, J., Ma, J., Tian, J., Ma, J., Zhang, D.: A robust method for vector field learning with application to mismatch removing. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 2977–2984. IEEE, New York (2011)
25. Zhou, F., De la Torre, F.: Factorized graph matching. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 127–134. IEEE, New York (2012)

Dense Segmentation-Aware Descriptors

Eduard Trulls, Iasonas Kokkinos, Alberto Sanfeliu,
and Francesc Moreno-Noguer

Abstract Dense descriptors are becoming increasingly popular in a host of tasks, such as dense image correspondence, bag-of-words image classification, and label transfer. However, the extraction of descriptors on generic image points, rather than selecting geometric features, requires rethinking how to achieve invariance to nuisance parameters. In this work we pursue invariance to occlusions and background changes by introducing segmentation information within dense feature construction. The core idea is to use the segmentation cues to downplay the features coming from image areas that are unlikely to belong to the same region as the feature point. We show how to integrate this idea with dense SIFT, as well as with the dense scale- and rotation-invariant descriptor (SID). We thereby deliver dense descriptors that are invariant to background changes, rotation, and/or scaling. We explore the merit of our technique in conjunction with large displacement motion estimation and wide-baseline stereo, and demonstrate that exploiting segmentation information yields clear improvements.

1 Introduction

Dense descriptors can be understood as replacing the convolution operations used in traditional image filterbanks [4, 23] with local descriptors, such as SIFT [20], that are better suited to tasks such as image correspondence, classification, and labeling. Starting from [25], which demonstrated the merit of replacing sparse with regularly sampled SIFT descriptors for the task of image classification, a line of subsequent works [11, 18, 40] quickly established dense descriptors as a versatile, efficient, and high-performing front end for vision tasks. In particular for dense correspondence, the seminal work of SIFT-flow [18] demonstrated that replacing

E. Trulls • A. Sanfeliu • F. Moreno-Noguer

Institut de Robòtica i Informàtica Industrial (UPC/CSIC), C/ Llorens i Artigas 4-6,
08028 Barcelona, Spain

e-mail: etrulls@iri.upc.edu; sanfeliu@iri.upc.edu; fmoreno@iri.upc.edu

I. Kokkinos (✉)

Ecole Centrale Paris, Grande Voie des Vignes, 92295 Chatenay-Malabry, France
e-mail: iasonas.kokkinos@ecp.fr

the common “brightness constancy” constraint of optical flow with a SIFT based notion of similarity facilitates novel applications, such as scene correspondence and label transfer.

Using dense descriptors however requires rethinking how to achieve invariance. Sparse descriptor techniques [2, 20, 24, 30, 34, 35, 46] first use an interest point detector that finds stable scale- and rotation-invariant points and then pool over coordinate systems adapted around these points to extract scale- and rotation-invariant descriptors. This strategy however is impossible in the dense setting, as image scale and orientation cannot be reliably estimated in most image areas: for instance defining scale is problematic around 1D edges while defining orientation is problematic on flat image areas.

Several recent works have addressed the scale- and/or rotation-invariance issue in the dense setting, either by adapting global image registration techniques to local image descriptors [13, 14, 19, 31] or by searching for invariant subspaces, as in the scale-less SIFT (SLS) descriptor of [12]. We will elaborate on such techniques in Sect. 2, where we present in more detail the scale-invariant descriptor (SID) [13, 14] which our work builds on.

In this chapter we push this line of works a step further, achieving invariance to “background changes.” In particular, some of the measurements used to construct a descriptor around a point may often come from distinct, independent objects, and will therefore differ in a new instantiation of the same point. One such case is illustrated in Fig. 1, where the object is roughly planar, but its background changes and therefore descriptors computed near its boundaries differ. Another case is shown in Fig. 2, where the scene stays the same, but due to its complicated geometry we have self-occlusions yielding again different descriptors across the two scenes. We will henceforth be using the term “background variability” as a common term for these two, and other cases (e.g., occlusion by objects lying closer to the camera), where observations that do not stem from the same planar region as the feature point may affect the point’s descriptor.

While this is an unavoidable problem when constructing local descriptors, we are aware of only a few works addressing it. In an early work in this direction,¹ Stein and Hebert [36] introduced a local membership function to eliminate background information from (sparse) SIFT descriptors; the image gradients that are pooled for the per-bin histograms were pre-multiplied by this function, resulting in a shunning of the background image gradients. In [40] the authors reported substantial performance improvements in multiview stereo by treating occlusion as a latent variable in an iterative stereo matching algorithm: at every iteration, each pixel chooses a (discretized) orientation variable and discards the feature measurements coming from the half-disk lying opposite to it. When interleaved with successive rounds of stereo matching this yields increasingly refined depth estimates.

Interestingly in image processing the idea of blocking the interference between different regions is a long-established idea that can be traced back to the

¹We were unaware of this work when first publishing [43].

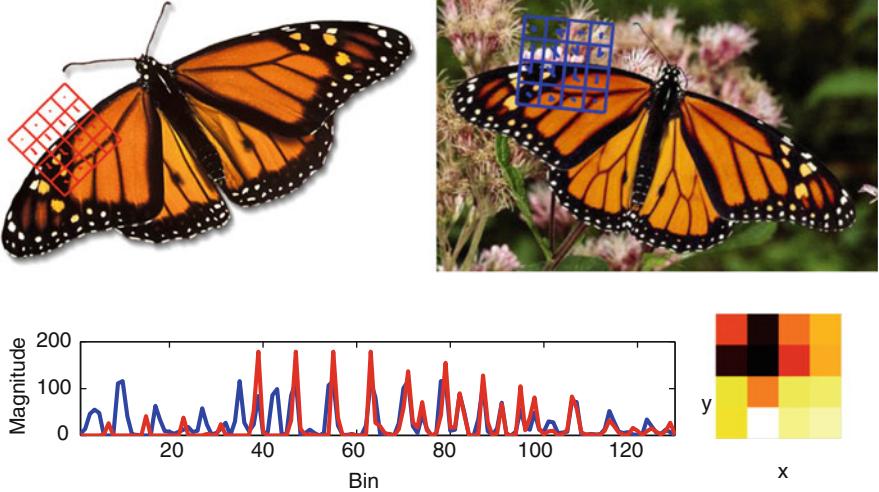


Fig. 1 Matching with background interference. We show images for two monarch butterflies, strikingly similar in appearance and pose (accounting for rotation), one over a *white background* and another in the wild. We plot the SIFT descriptors for two corresponding points found by hand, in *red* (left image) and *blue* (right image), close to object boundaries. On the *bottom left*, we plot the descriptor values, ordered: orientation bin first, x -bin second, y -bin third. On the *bottom right*, we plot the similarity between descriptors, averaged over the orientation bins (*white* is better, *red* is worse). Notice that the orientation histograms match well for the cells which lie on the foreground, but not for the background cells. This hurts correspondences around object boundaries

structure-preserving filtering used in nonlinear diffusion [26, 29], the bilateral filter [41], and the segmentation-sensitive normalized convolution of [28], while also bearing some resemblance to the self-similarity descriptor of [32]. In this light, our work can be understood as bringing to the problem of descriptor construction the insight of blocking the flow of information across distinct regions by using the image to construct a local region membership function.

In particular, our aim is to eliminate, or at least reduce, the effects of background changes when extracting descriptors. For this we use a “mid-level” segmentation module to reason about which pixels go together: as illustrated in Fig. 3, we incorporate segmentation information through a soft “gating” mask that modulates local measurements, effectively shunning those parts of the image which apparently do not belong to the same object/surface as the center of the descriptor. In particular, we use segmentation cues (Fig. 3b) to compute the affinity of a point with its neighbors (Fig. 3c), and downplay image measurements most likely to belong to different regions or objects (Fig. 3d).

We argue that our approach has the following favorable aspects: first, it is fairly *general*. We apply it to two different descriptors (SIFT and SID), with three different segmentation techniques (spectral clustering [21], generalized boundaries [17], structured edge forests [9]), for two different applications (motion and stereo). In all cases we demonstrate that the introduction of segmentation results in systematically

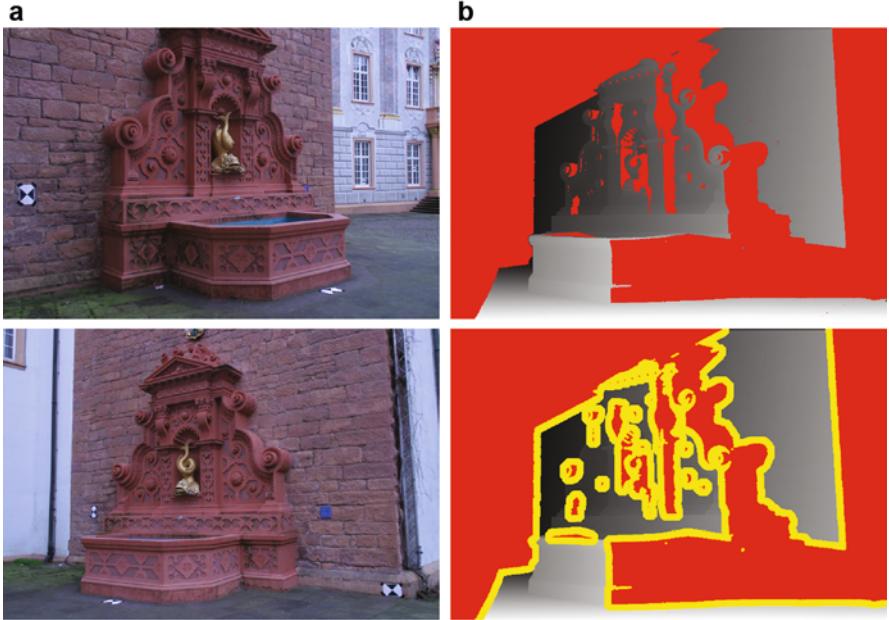


Fig. 2 Matching with occlusions. **(a)** Two images from wide-baseline stereo, with large occluded areas. **(b)** On the *top row*, we show the ground truth depth map, from the view farthest to the right. Occlusions, determined from ground truth visibility maps, are marked in *red*. On the *bottom row*, we dilate the occlusion map by 25 pixels and plot the result in *yellow*. We can think of these as the coordinates of the pixels that will be affected if we compute descriptors over circular areas of radius 25 pixels, as occluded areas creep into their domain—pixels closer to occlusion boundaries will suffer more. As the baseline increases, more and more pixels are beset by this problem: over 30 % of visible pixels in this case

better results over the respective baselines. Second, it is *simple*. It requires no training and simply modifies the values of an existing feature descriptor. As such, it can be used in any application that relies on descriptors. Third, it incurs *minimal overhead*. The affinity masks can be computed and applied efficiently, in the order of a few seconds [17] or even a fraction of a second [9], for *dense* descriptors. Fourth, our method has a *single parameter*, which can be used to adjust the “hardness” of the masks. We fix it once and use it throughout our experiments—even across different applications.

In Sect. 2 we describe the SID descriptor, which achieves scale and rotation invariance in dense descriptor construction and serves as the starting point of our work. In Sect. 3, we present how we extract segmentation information from images in a manner that makes it straightforward to extract dense soft segmentation masks and, in turn, dense segmentation-aware descriptors. Finally, we benchmark our descriptors on two different applications: large displacement motion estimation and wide-baseline stereo. We demonstrate that the introduction of segmentation cues yields systematic improvements.

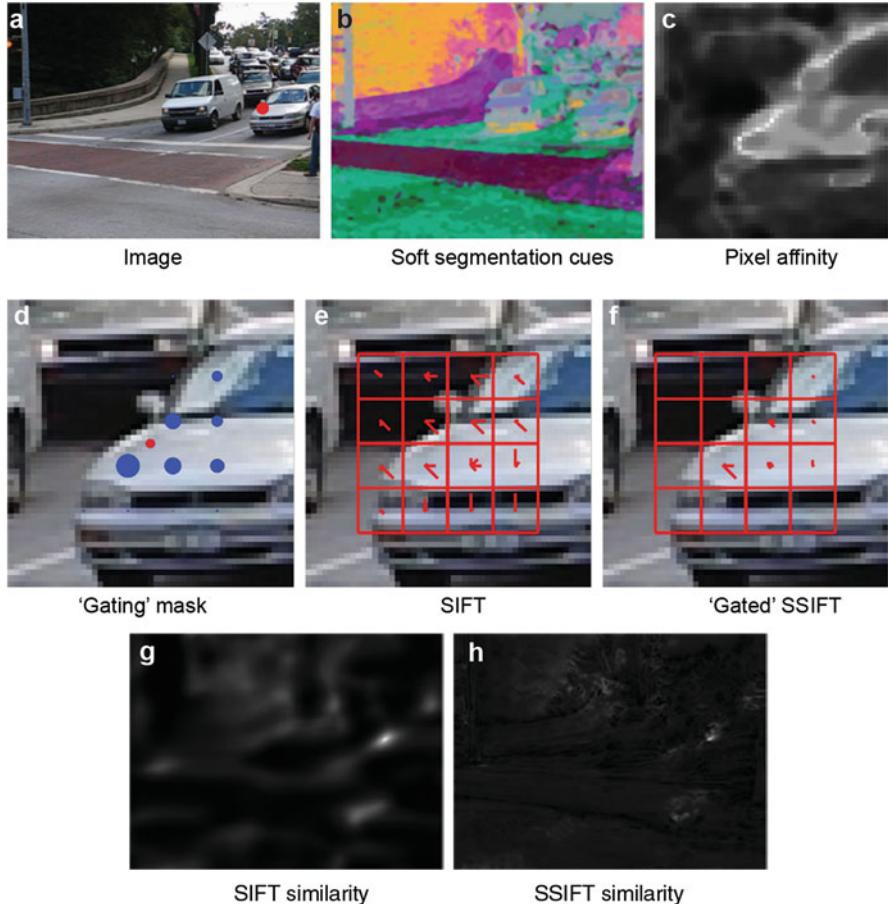


Fig. 3 We exploit segmentation information to construct feature descriptors that are robust to background variability. (a) Input image: we want to compute a descriptor for the pixel indicated by the red dot. (b) Segmentation embeddings, visualized in terms of their first three coordinates in RGB space; pixels with similar segmentation embeddings are likely to belong to the same region. (c) Affinity between the pixel represented by the red dot and its neighbors, measured in terms of the Euclidean distance in embedding space. (d) A “gating” mask that encodes the reliability, i.e., region similarity, of the locations used to compute the SIFT descriptor (e). We can thus construct a segmentation-aware SIFT (SSIFT) descriptor (f) by “gating” the descriptor features with the mask. (g, h) show the distance between the descriptors shown in (e)–(f) and respective dense SIFT/SSIFT descriptors over the whole image domain; we note that the SSIFT similarity function peaks more sharply around the pixel, indicating its higher distinctiveness

2 SID: A Dense Scale- and Rotation-Invariant Descriptor

In several applications it can be desirable to construct a scale-invariant descriptor densely, for instance when establishing dense image correspondences in the presence of scale changes. In such cases scale selection is not appropriate, as it is only reliably applicable around a few singular points (e.g., blobs or corners).

We argue that one can instead adapt the Fourier Transform Modulus-based image registration technique [6, 27, 47] to the construction of descriptors and thereby guarantee scale and rotation invariance at any image point, without requiring scale selection. Starting with an illustration of the technique for a one-dimensional signal, we will then briefly present how it applies to image descriptors; a more extensive presentation and evaluation is contained in [14], while we had originally presented this technique for sparse descriptors in [13].

We first consider describing a one-dimensional signal $f(x)$, $x > 0$ in a manner that will not change when the signal is scaled as $f(x/a)$, $a > 0$. Using the domain transformation $x' = \log(x)$ we can define a new function f' such that

$$f'(x') \doteq f(x), \quad \text{where } x' = \log x, \quad (1)$$

which is what we will be referring to as the “logarithmically transformed” version of f ; this is illustrated also in the left column of Fig. 4. For this particular transformation, dilating f by a will amount to translating f' by a constant, $\log(a)$:

$$f'(x' - \log(a)) = f(x/a), \quad (2)$$

We can thus extract a scale-invariant quantity based on the fact that if $g(x)$ and $G(\omega)$ are a Fourier transform pair, $g(x - c)$ and $G(\omega)e^{-j\omega c}$ will be a transform pair as well (by the shifting-in-time property). Defining $f_a(x') = f'(x' - \log(a))$ and denoting by $\mathcal{F}_a(\omega)$ the Fourier Transform of $f_a(x)$ we then have

$$\mathcal{F}_a(\omega) = \mathcal{F}_1(\omega)e^{-j\log(a)\omega} \quad \text{or} \quad (3)$$

$$|\mathcal{F}_a(\omega)| = |\mathcal{F}_1(\omega)|. \quad (4)$$

From Eq. (4) we conclude that changing a will not affect the Fourier Transform Modulus $|\mathcal{F}_a(\omega)|$ of f_a , which can thus be used as a scale-invariant descriptor of f .

As shown in the next columns of Fig. 4, a 2D scaling and rotation can similarly be converted into a translation with a log-polar transformation of the signal—and then eliminated with the FTM technique. The principle behind this approach is commonly used in tasks involving global transformations such as image registration [6, 47] and texture classification [27].

Adapting the FTM technique to the construction of local descriptors requires first a discrete formulation. We construct a descriptor around a point $\mathbf{x} = (x_1, x_2)$ by sampling its neighborhood along K rays leaving \mathbf{x} at equal angle increments $\theta_k = 2\pi k/K$, $k = 0, \dots, K - 1$. Along each ray we use N points whose distances

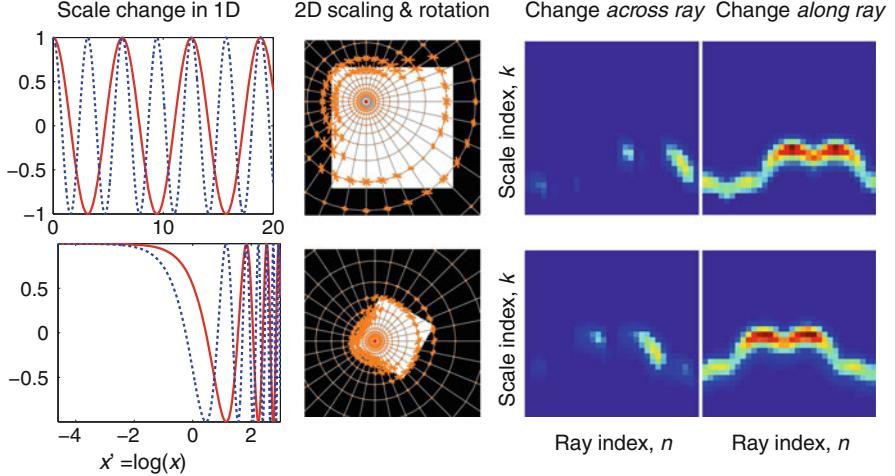


Fig. 4 Turning scaling into translations for 1D and 2D signals: The *left column* demonstrates for a 1D signal how the logarithmic transformation $x' = \log(x)$ turns scaling into translation: the *red solid* ($f(x) = \cos(x)$) and *blue dashed* ($g(x) = \cos(2x)$) functions differ by a scale factor of two; the transformation $f'(x) = f(\log(x))$ delivers $f'(x) = \cos(\log(x))$, $g'(x) = \cos(\log(x) - \log(2))$, which differ by a translation. The next columns illustrate the same effect for 2D signals. The *second column* shows the descriptors computed on a point before and after scaling and rotating an image; the needle length indicates directional derivative magnitude. The next *two columns* show the respective magnitudes across and along the ray direction, demonstrating that image scaling and rotation are turned into translations. The point is arbitrary (i.e., not a corner/junction/blob center), and therefore scale selection around it would not be reliable, or even feasible

from \mathbf{x} form a geometric progression $r_n = c_0 a^n$. The signal measurements on those points provide us with a $K \times N$ matrix:

$$h[k, n] = f[x_1 + r_n \cos(\theta_k), x_2 + r_n \sin(\theta_k)]. \quad (5)$$

By design, image scalings and rotations of the image amount to translations over the radial and angular dimensions, respectively, of this descriptor. From the time-shifting property of the Discrete-Time Fourier Transform (DTFT) we know that if $h[k, n] \xleftrightarrow{\mathcal{F}} H(j\omega_k, j\omega_n)$ are a DTFT pair, we will then have

$$h[k - c, n - d] \xleftrightarrow{\mathcal{F}} H(j\omega_k, j\omega_n) e^{-j(\omega_k c + \omega_n d)}; \quad (6)$$

therefore taking the absolute of the DTFT yields a scale- and rotation-invariant quantity.

We can apply the Fourier Transform only over scales to obtain a scale-invariant but rotation-dependent quantity (and vice versa, for a rotation-invariant and scale-sensitive quantity). This can be useful in scenes with scaling changes but no rotations, where we would be discarding useful information. In our evaluations we

will refer to the scale- and rotation-invariant descriptor as **SID** and to the scale-invariant but rotation-sensitive descriptor as **SID-Rot**.

Apart from a discrete formulation, we also need to preprocess the image so as to (a) discount illumination changes, (b) allow for efficient dense computation, and (c) account for sampling effects. Regarding (a), for invariance to additive illumination changes we use the *directional derivatives* of the signal along a set of orientations offset by the current ray’s orientation (see, e.g., Fig. 4 for the components along and perpendicular to the ray). We extract features at H' orientations, preserving polarity, so that the effective number of orientation histogram bins is $H = 2 \cdot H'$. To discount multiplicative illumination changes we normalize the features to unit L_2 norm. For (b), memory- and time-efficient dense computation we combine Daisy [40] with steerable filtering [10] and recursive Gaussian convolutions [7]. Finally for (c), dealing with sampling effects, as proposed in [13, 40], we use a “foveal” smoothing pattern, with a smoothing scale that is linear in the distance from the descriptor’s center.

In Fig. 5 we show the values of the lowest-frequency coefficients of densely computed descriptors on two images related by scaling and rotation. We see that the descriptor values are effectively invariant, despite a scaling factor in the order of 2.

Before proceeding we note that apart from SID, the SLS descriptor was introduced in [12], also with the goal of achieving scale invariance for dense descriptors. The SLS approach is to compute a set of dense SIFT descriptors at different scales for each point and project them into an invariant low-dimensional subspace that

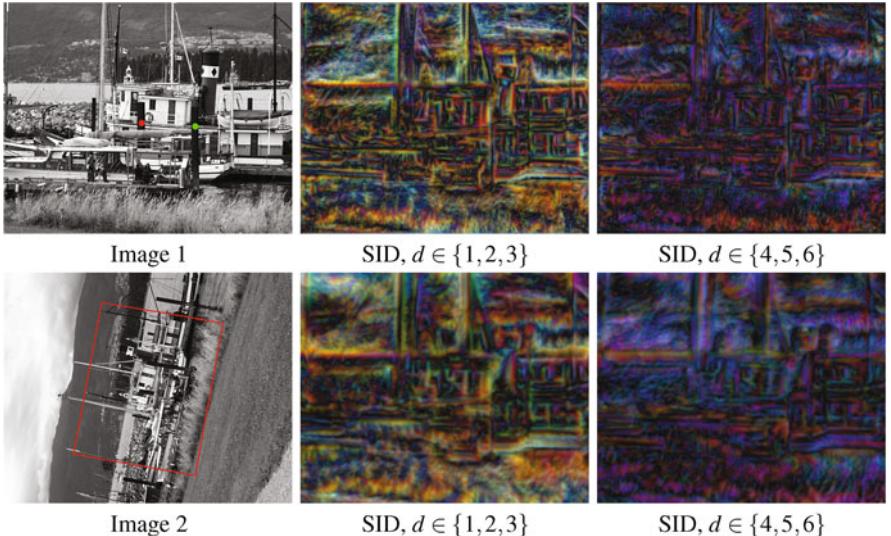


Fig. 5 Visualization of dense SID: the location of image 1 within image 2 is indicated by the red box; the scaling transformation amounts to an area change in the order of four. We align the descriptors for the *bottom row* (red box) with the *top row* image after computing them and visualize three of their dimensions in RGB space—demonstrating that they are effectively invariant

elicits the scale-invariant aspects of these descriptors. SLS gives clearly better results than dense SIFT in the presence of scaling transformations, but SID can also be rotation invariant and is substantially faster to compute. We include this state-of-the-art descriptor in our benchmarks in Sect. 4 and refer to the chapter on SLS for details on its construction.

3 Dense Segmentation-Aware Descriptors

In this section we turn to discarding background variability by exploiting segmentation. Our goal is to construct feature descriptors that are contained within a single surface/object (“region” from now on). In this way, changes in the background, e.g., due to layered motion, will not affect the description of a point in the interior of a region. Similarly, when a region is occluded by another region in front of it, even though we cannot recover its missing information, we can at least ignore irrelevant occluders.

Achieving this goal can benefit SIFT as well as any other descriptor, but its merit is most pertinent to SID. In particular, image scaling does not necessarily result in a cyclic permutation of the SID elements: the finest- and coarsest-level entries can change. As such the (circular) shifting relationship required to obtain the DTFT pair in Eq. (6) does not strictly hold. To remedy this issue SID typically uses large image patches and many rings, so that the percentage of points where this change happens eventually becomes negligible; this however limits SID’s applicability, since background structures and occlusions can easily creep into its construction.

If we were able to use information only from the region containing a point, we could make a descriptor invariant to background changes: as shown in the first column of Fig. 6, given the support of the region containing a pixel we can identify the descriptor elements that come from different regions and set them to zero. However, since segmentation is far from solved we turn to algorithms that do not strongly commit to a single segmentation, but rather determine the affinity of a pixel to its neighbors in a soft manner. This soft affinity information is then incorporated into descriptor construction, in the form of a “gating” signal.

Namely, when constructing a descriptor around a point \mathbf{x} we measure an affinity $w[k, n] \in [0, 1]$ between \mathbf{x} and every other grid coordinate $\mathbf{x}[k, n]$ and multiply the respective measurements \mathbf{d} extracted around $[k, n]$ with it:

$$\mathbf{d}'[k, n] = w[k, n]\mathbf{d}[k, n]. \quad (7)$$

In Eq. (7), $\mathbf{d}[k, n]$ represents for SID the concatenation of the H polarized-smoothed derivatives at $[k, n]$ and for SIFT the respective eight-dimensional orientation histogram. Multiplying by $w[k, n]$ effectively shuns measurements which come from the background; as such, the descriptor extracted around a point is affected only by points belonging to its region and remains robust to background variability. As our results indicate, replacing \mathbf{d} by \mathbf{d}' yields noticeable performance improvements.

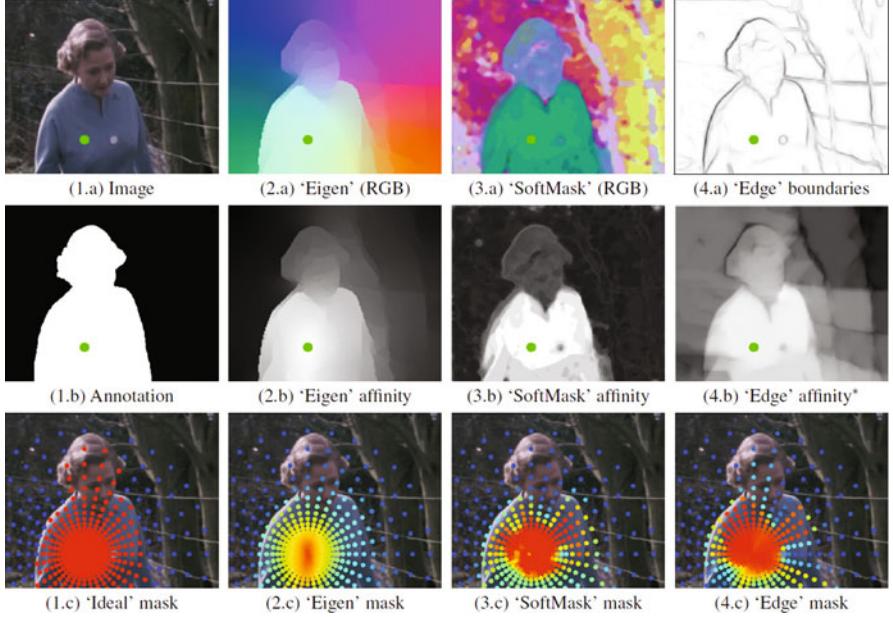


Fig. 6 Segmentation-aware descriptor construction. *Column 1:* given image (1.a) and a “perfect” figure-ground segmentation (1.b), separating foreground and background measurements would be trivial: (1.c); grid points are marked in *red* if enabled and *blue* if disabled. This is unattainable—we propose alternative solutions based on different segmentation cues. *Columns 2–4:* given segmentation cues [(2–4).a], we can measure the “affinity” between pairs of pixels: in [(2–4).b], we show the affinity between the point represented by the *green dot* and the rest of the image. We use this per-pixel affinity to design “gating” masks [(2–4).c]. We present procedures to leverage the normalized-cut eigenvectors of [21] (“Eigen,” *column 2*), the generalized Pb soft segmentations of [17] (“SoftMask,” *column 3*), and the structured forests boundaries of [9] (“Edge,” *column 4*). Notice that (2.b) and (3.b) are for illustration: in practice we do this only for grid coordinates $\mathbf{x}[k, n]$ (pictured in the *bottom row*), smoothing the embeddings with filters of size increasing with n , prior to sampling. For the “Edge” masks we use an affinity measure computed over the radial coordinates n , which does not lend itself to an image-based representation—for illustration purposes in (4.b) we show a distance transform instead [3]. Please refer to Sect. 3.3 for details

Having provided the general outline of our method, we now describe three alternative methods to obtain the affinity function $w[k, n]$ used in Eq. (7).

3.1 “Eigen” Soft Segmentations (*gPb Detector*)

First, we use the image segmentation approach of [21]: we treat the image as a weighted graph, with nodes corresponding to pixels and weights corresponding to the low-level affinity between pixels. The latter is obtained in terms of the *intervening contour cue* [33], which measures the presence of strong boundaries between

two pixels. One can phrase the segmentation problem as a global optimization of the *normalized cut* [33] objective defined on the (discrete) labeling of this graph, which is NP hard. However, relaxing this problem yields a generalized eigenvector problem of the form

$$(D - W)\mathbf{v} = \lambda D\mathbf{v}, \quad (8)$$

where $\mathbf{v} \in R^P$ is the relaxed solution for the P image pixels, W is a $P \times P$ affinity matrix encoding the low-level affinity between pixels, and D is a diagonal degree matrix with $D_{ii} = \sum_j W_{ij}$.

Even though this eigenvector problem can be solved exactly (albeit slowly), turning the computed eigenvectors into a segmentation is not obvious. Several post-processing techniques have been proposed (e.g., eigenvector-level clustering, by Shi and Malik [33], or eigenvector-based features in [22]), but do not necessarily stem from optimizing the original problem. Instead, we propose to use the eigenvectors directly, as continuous *pixel embeddings*; namely, every point \mathbf{x} is mapped to a higher-dimensional space \mathbf{y} where Euclidean distances indicate the probability of two pixels falling in different regions. This is closer in spirit to “Laplacian eigenmaps” [1].

In particular we construct $\mathbf{y}(\mathbf{x})$ by weighting the first $M = 10$ eigenvectors by a quantity dependent on their corresponding eigenvalues,

$$\mathbf{y}(\mathbf{x}) = \left[\frac{1}{\sqrt{\lambda_1}} \mathbf{v}_1(\mathbf{x}), \dots, \frac{1}{\sqrt{\lambda_M}} \mathbf{v}_M(\mathbf{x}) \right]^T, \quad (9)$$

so that lower-energy eigenvectors (global structures) have a stronger weight. Indicative examples of the first three dimensions of $\mathbf{y}(\mathbf{x})$ can be seen in Fig. 7b.

Based on the assumption that Euclidean distance in \mathbf{y} indicates how likely two points are to belong to the same region, we measure the descriptor-level affinity, w , between two points \mathbf{x}, \mathbf{x}' as

$$w = \exp(-\lambda \|\mathbf{y}(\mathbf{x}) - \mathbf{y}(\mathbf{x}')\|_2). \quad (10)$$

Here λ is a single scalar design parameter determining the sharpness of the affinity masks, which we set experimentally in Sect. 4. We show some pixel affinities (Euclidean distances over the embedded space) and segmentation masks in Fig. 6. For simplicity, we use ‘Eigen’ to refer to the embeddings and masks we derive from this approach.

3.2 “SoftMask” Soft Segmentations (*Gb Detector*)

As an alternative to the “Eigen” embeddings, we also explore the soft segmentations employed in the *generalized* boundary detector *Gb* of [17]. Their method uses local color models, built around each pixel, to construct a large set of figure-ground

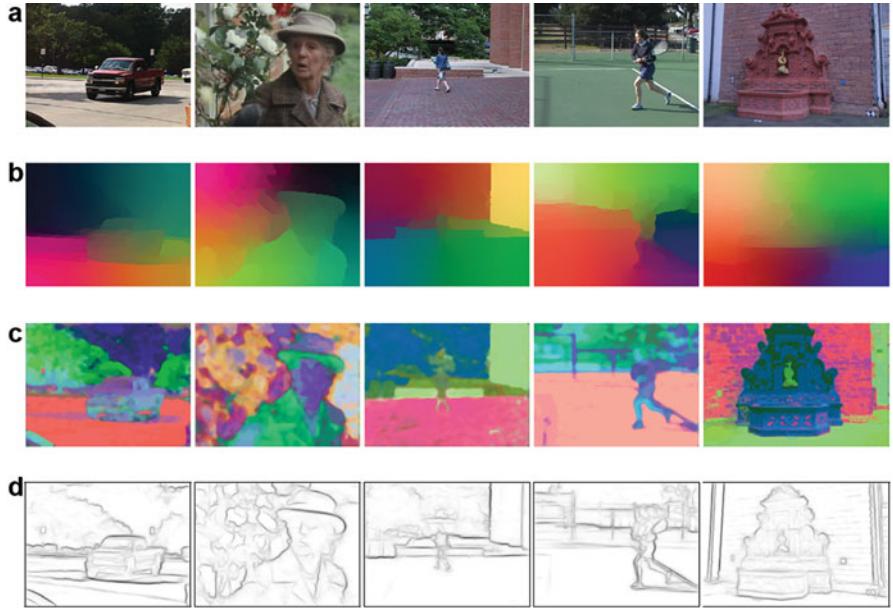


Fig. 7 Segmentation cues used in this work. **(a)** Source image, **(b)** top three “Eigen” embeddings in RGB space, **(c)** top three “SoftMask” embeddings in RGB space, **(d)** structured forest “Edge” boundaries

segmentations. These segmentations are then projected onto a lower-dimensional subspace through PCA. As before, we can take the top $M = 8$ components, which provides us again with low-dimensional pixel embeddings.

The main advantage of these features is that they are obtained at a substantially smaller computational cost, whereas building and solving for the eigenvectors in Eq. (8) is expensive for any but the smallest images. We refer to these embeddings, and their associated masks as “SoftMask.”

Figure 7c shows the G_b soft segmentations for several images. We notice that the “SoftMask” embeddings have higher granularity than the “Eigen” embeddings: on the one hand this makes them more noisy, but on the other hand this also allows them to better capture small features.

3.3 “Edge” Boundaries (Structured Forest Detector)

The third method we explore uses the state-of-the-art Structured Forest boundary detector of [9], which has excellent detection performance while operating at multiple frames per second. Unlike the previous two methods, this method does not provide an embedding, but rather measures the probability that two adjacent pixels

may belong to different regions. In order to efficiently extend this measurement beyond adjacent pixels we adapt the “intervening contour” technique of [33] to the descriptor coordinate system.

We start by sampling the boundary responses on the log-polar grid of SID; we use smoothing prior to sampling, so as to achieve scale-invariant processing. This sampling provides us with a boundary strength signal $B[k, n]$ that complements the descriptor features in Eq. (5). We then obtain the affinity function $w[k, n]$ in Eq. (7) in terms of the running sum of $w[k, n]$ along the radial coordinate k :

$$w[k, n] = \exp\left(-\lambda' \sum_{k'=0}^{k-d} B[k, n]\right). \quad (11)$$

We have introduced an additional quantity, d , in Eq. (11), which acts like a “mask dilation” parameter. Namely, this allows us to postpone (by d) the decay of the affinity function w around region boundaries, thereby letting the descriptor profit from the shape information contained around boundaries. We have empirically observed that setting $d = 2$ or $d = 1$ yields a moderate improvement over $d = 0$. We refer to the segmentation masks computed in this manner as “Edge,” for convenience. Figure 7d shows some boundaries obtained with the Structured Forest detector.

4 Experimental Evaluation

We consider two scenarios: video sequences with multilayered motion and wide-baseline stereo. We explore the use of the different segmentation cues described in Sect. 3 and several dense descriptors [SID, segmentation-aware SID, dense SIFT (DSIFT), segmentation-aware dense SIFT, SLS, and Daisy]. We use the “S” prefix to indicate “segmentation-aware,” so that for instance “SSID” stands for our variant of SID.

4.1 Large Displacement, Multilayered Motion

In this experiment we estimate the motion of objects across time over the image plane, i.e. optical flow. This problem is usually formulated as an optimization over a function that combines a data term that assumes constancy of some image property (e.g. intensity) with a spatial term that models the expected variation of the flow fields across the image (e.g., piecewise smoothness).

Traditional optical flow methods rely on pixel data to solve the correspondence problem. We use SIFT flow [18], which follows a formulation similar to that of optical flow but exploits densely sampled SIFT descriptors rather than raw pixel values. SIFT flow is designed for image alignment, and unlike optical flow it is amenable not only to different views of the same scene but also to different instances of scenes belonging to the same category—hence the use of dense SIFT, which has proven successful in image registration. Moreover, this approach can be applied to any feature descriptor that can be computed densely and was previously paired with SLS in [12]. We use this framework to estimate the motion between pairs of frames, using different descriptors as features.

We test our approach on the Berkeley Motion Dataset (MOSEG) [5], which itself is an extension of the Hopkins 155 dataset [42]. The MOSEG dataset contains 10 sequences of outdoor traffic taken with a handheld camera, three sequences of people in movement, and 13 sequences from the TV series *Miss Marple*. All of them exhibit multilayered motion. For these experiments we consider only the traffic sequences, as in many of the others, the “objects” in the scene (e.g., people) disappear or occlude themselves (e.g., turn around)—the dataset is geared towards long-term *tracking*. Ground truth object annotations (segmentation masks) are given for a subset of frames in every sequence—roughly one annotation every ten frames.

For each sequence, we pair the first frame with all successive frames for which we have ground truth annotations, yielding 31 frame pairs. The images are resized to 33 %, in particular to permit comparison with SLS, which has high memory requirements. We design an evaluation procedure based on the segmentation annotations, proceeding as follows:

1. We compute flow fields for each descriptor type.
2. We use the flow estimates to warp the annotations for every frame $I_j, j > 0$ over the first frame (I_0).
3. We compute the overlap between the annotations for frame I_0 and the warped annotations for every frame $I_j, j > 0$ using the Dice coefficient [8] as a metric.

We consider DSIFT [45], SLS, SID, and SSID with “Eigen,” “SoftMask,” and “Edge” embeddings. We use SLS both in its original form [12] and a PCA variant developed afterwards: we refer to them as SLS-“paper” and SLS-PCA. A SLS descriptor is size 8256, whereas its PCA variant is size 528. The code for both was made publicly available by the authors.

For SID construction we use the dense implementation of [14]. We take $K = 28$ rays, $N = 32$ points per ray, and $H' = 4$ oriented derivatives, preserving the polarity as in [40], so that the effective number of orientations is $H = 8$. We exploit the symmetry of the Fourier Transform Modulus to discard two quadrants, as well as

the DC component, which is affected by additive lighting changes. The size of the descriptor is 3328 for SID and 3360 for SID-Rot. We refer to the publicly available code for further details.²

For the SID-based descriptors we consider only their rotation-sensitive version, SID-Rot, as the objects in the MOSEG sequences do not contain significant rotations—discarding them in such a case entails a loss of information and a decrease in performance. We use the same parameters for both SID and SSID unless stated otherwise. We use this experiment to determine the values for the λ parameter of SSID of Eq. (10): $\lambda = 0.7$ for “Eigen” and $\lambda = 37.5$ for “SoftMask”; and Eq. (11): $\lambda = 27.5$ for “Edge.”

Figure 8 plots the results for every descriptor. Each bin shows the average overlap for all frame pairs under consideration. The results are accumulated, so that the first bin includes all frame pairs ($j \geq 10$), the second bin includes frame pairs with a displacement of 20 or more frames ($j \geq 20$), and so on. We do so to prioritize large displacements; the sequences have varying lengths, so that the samples are skewed towards smaller displacements. As expected, SSID outperforms SID, in particular

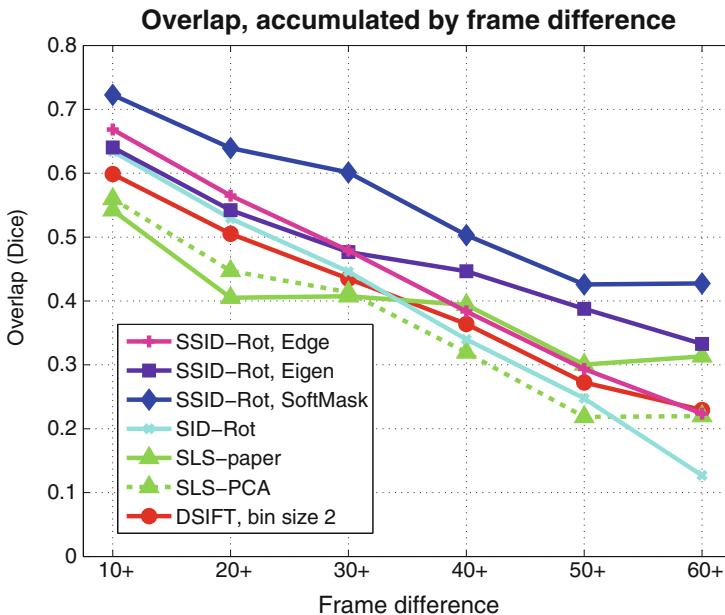


Fig. 8 Overlap results over the MOSEG dataset, for all the dense descriptors considered. Each bin shows the average overlap for all frame pairs under consideration. The results are accumulated, so that the first bin (“10+”) includes all frame pairs and subsequent bins (“ $j+$ ”) include frame pairs with a difference of j or more frames. For DSIFT we show only the results corresponding to the best scale

²<https://github.com/etrullls/softseg-descriptors-release>.

for large displacements, which are generally correlated with large j . The best overall results are obtained by SSID-Rot with “SoftMask” embeddings, followed by SSID-Rot with “Eigen” embeddings—the “SoftMask” variant does better, despite its reduced computational cost. The “Edge” boundaries also provide a boost over the segmentation-agnostic SID—while they do not perform as well as the “SoftMask” embeddings, they reduce the cost of extracting the segmentation cues even more drastically.

Additionally, we use the flow fields to warp each image I_j , over I_0 . Some large displacement warps are pictured in Fig. 9—again, SSID outperforms the other descriptors.

4.2 Segmentation-Aware SIFT

The application of soft segmentation masks over SID is particularly interesting because it alleviates its main shortcoming: fine sampling over large image areas to achieve invariance. But its success suggests that this approach can be applied to other standard grid-based descriptors—namely, SIFT. We extend the formulation to SIFT’s 4×4 grid, using the “SoftMask” embeddings which give us consistently better results with SSID. Figure 10 shows the increase in performance over four different scales. The gains are systematic, but as expected the optimal λ is strongly correlated with the spatial size of the descriptor grid. Figure 11 displays the performance gains; note that the variability could be potentially accounted for by the low number of samples (31 image pairs). This merits further study, in particular with regards to its application to the multiple scales considered in the construction of SLS descriptors.

4.3 Wide-Baseline Stereo

For a second experiment, we consider stereo reconstruction—i.e., the problem of computing a 3D representation of a scene given images extracted from different viewpoints. Stereo is one of the classical computer vision problems and has been studied for several decades. While *narrow-baseline* stereo (usually defined as two cameras separated by a short distance, pointing in the same direction) is well understood, the same cannot be said for its *wide-baseline* counterpart.

Narrow-baseline stereo is often addressed with simple similarity measures such as pixel differencing, or block-wise operations such as the sum of square differences (SSD) or normalized cross correlation (NCC). As the viewpoint increases, perspective distortion and occlusions become a problem and we cannot rely on these simple metrics—feature descriptors are more robust. Wide-baseline stereo has often

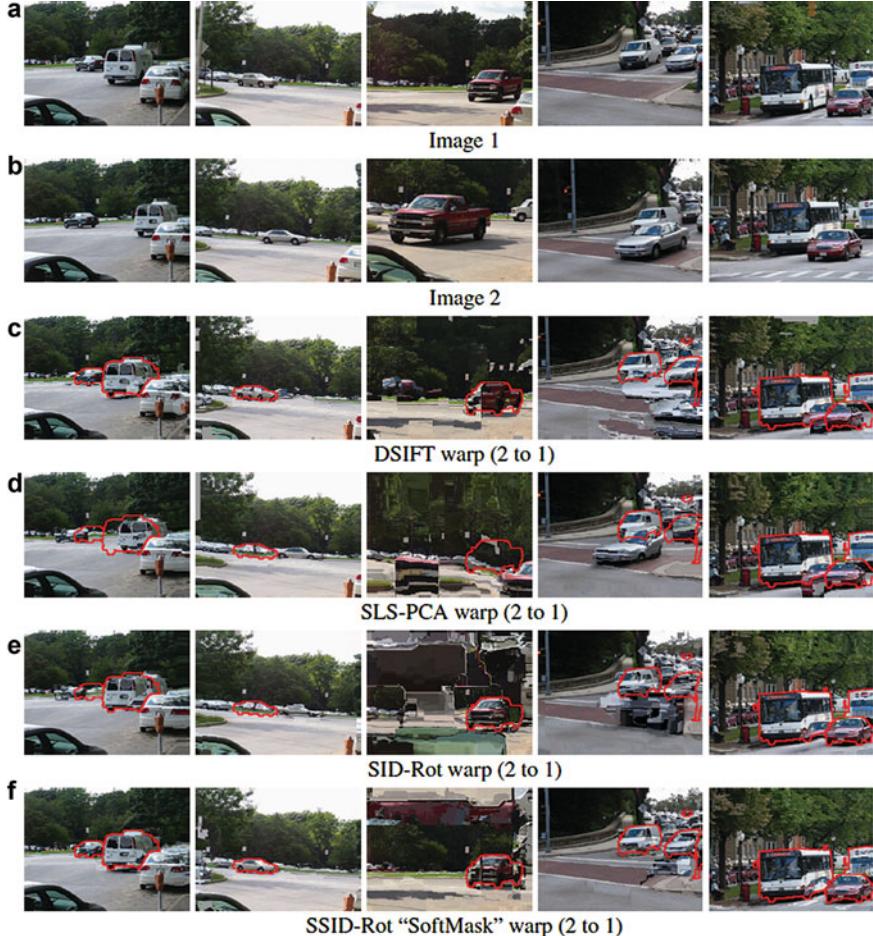


Fig. 9 Large displacement motion with SIFT flow, for some of the descriptors considered in this work. We warp image “2” to “1,” using the estimated flow fields. The ground truth segmentation masks are overlaid in red—a good registration should bring the object in alignment with the segmentation mask. We observe that segmentation-aware variant SSID does best—particularly over its baseline, SID. Similar improvements were observed for SDSIFT over DSIFT

been addressed as a multistep process, using sparse matches as anchors or seeds [37, 48], which can result in gross reconstruction errors if the first matching stage is inaccurate. Dense correspondences are a preferable method.

Modern dense stereo algorithms use local features to estimate the similarity between points and then impose global shape constraints to enforce spatial consistency. This problem is naturally formulated in terms of discrete energy minimization. For our experiments we use a setup similar to [40]:

1. We discretize 3D space into $L = 50$ depth bins, from the reference frame of the camera furthest to the right.

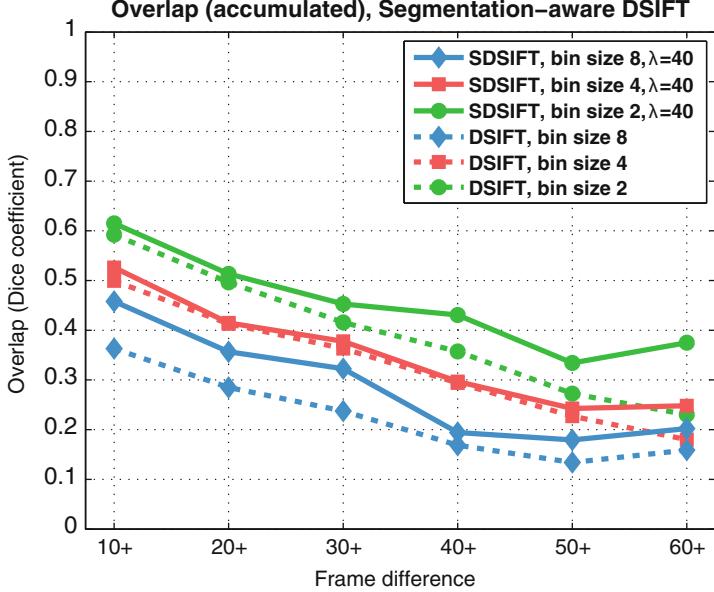


Fig. 10 Overlap results over the MOSEG dataset for segmentation-aware DSIFT and its baseline, at different scales

2. Given a calibrated stereo system, we compute the distance between every pixel in one image and all the possible matching candidates over the other image, subject to epipolar constraints, within the scene range.
3. We store the distance for the best match at every depth bin.
4. We feed the costs (distances) $N_w \times N_h \times L$, where N_w and N_h are the width and height of the image, to a global regularization algorithm, to enforce piecewise smoothness. Each pixel is assigned a label (depth bin) in $\mathcal{L} \in \{1, \dots, L\}$.

For the last step we use Tree-Reweighted Message Passing [16] with Potts pairwise costs; i.e., a constant penalty if adjacent pixels are assigned different depth labels and no penalty otherwise. We add an additional label with a constant cost, to model occlusions.

We use the wide-baseline dataset of [38], which contains two multiview sets of high-resolution images with ground truth depth maps. We consider the “fountain” set, as it contains much wider baselines in terms of angular variation than the “herzjesu” set, which exhibits mostly fronto-parallel displacements. As in [40], we use a much smaller resolution, in our case 460×308 .

First, we evaluate the accuracy of each descriptor. We compute depth maps using the algorithm we just described, and evaluate the error on every visible pixel, using the ground truth visibility maps from [38], without accounting for occlusions. We consider DSIFT, SLS, and Daisy, as well as SID and SSID. For DSIFT, SLS and Daisy we align the descriptors with the epipolar lines, to enforce rotation invariance,

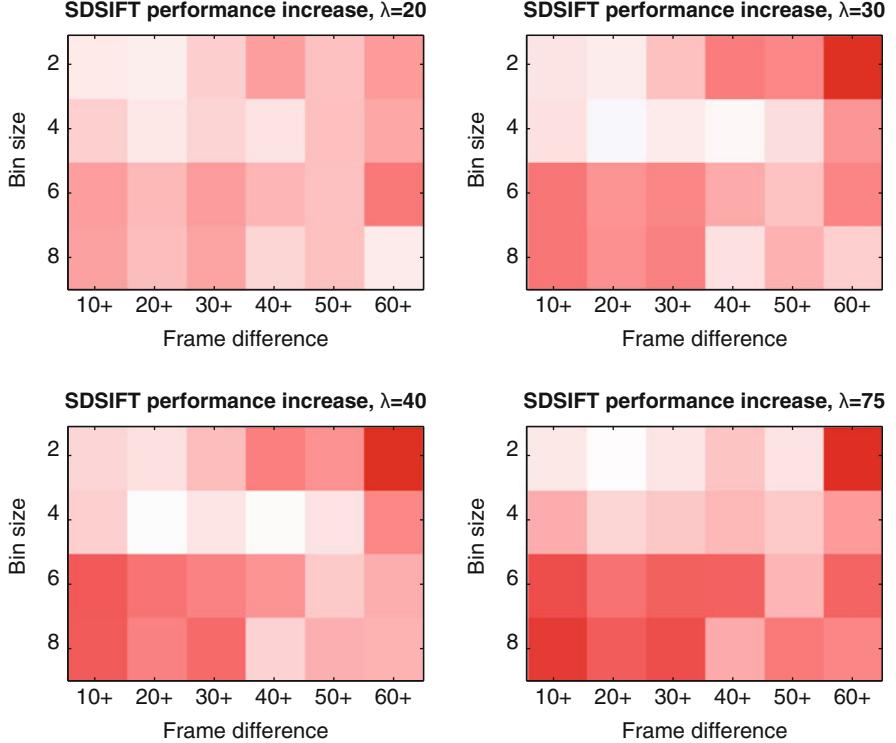


Fig. 11 Increase in average overlap for the segmentation-aware SIFT over its baseline, for several SIFT scales, difference in frames j (accumulated), and λ values. White signals no difference in overlap, with shades of red marking an increase (the largest increase in overlap is 0.14). For clarification, note the correspondence between the bottom left figure ($\lambda = 40$) and Fig. 10. As expected, high λ values produce more aggressive segmentation masks and more discriminating descriptors, but the optimal λ varies with the SIFT scale

as in [40]. For SID and SSID we consider only the *fully invariant* descriptors and omit this step. We use SLS-PCA rather than SLS-“paper”, which has much lower dimensionality: matching descriptors is the costliest step in dense stereo and is correlated with descriptor size. We show the results in Fig. 12. Our SID-based segmentation-aware descriptors outperform the others, except for SLS—but our approach does not require rotating the patch.

Most of the performance gains on wide-baseline stereo reported in [40] stem not from the Daisy descriptor but from their handling of occlusions. For this Tola et al. introduce a novel approach to latent occlusion estimation for iterative stereo reconstruction. Their technique exploits a set of binary masks, half-disks at different orientations, that disable image measurements from occluded areas. These are similar to our segmentation masks $w[k, n]$, but binary (i.e., $w' \in \{0, 1\}$) rather than soft ($w \in [0, 1]$) and with a predetermined spatial structure. The most appropriate

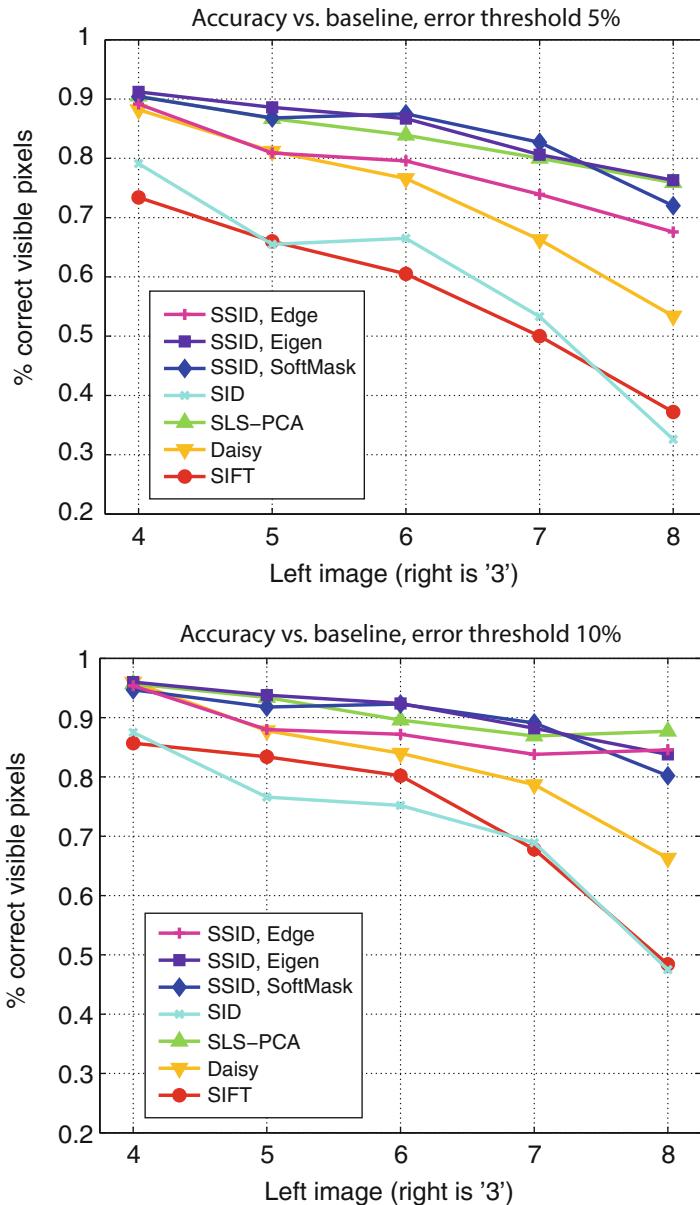


Fig. 12 Accuracy at different baselines, for visible pixels only, for two error thresholds (expressed as a fraction of the scene range). Occlusions are not taken into account

mask is determined on a per-pixel basis, using the current depth estimates around the pixel to prioritize masks that disable regions with heterogenous label distributions. Subsequent iterations apply the highest-scoring masks to the descriptors as in Eq. (7), dropping the measurements likely affected by occlusions from the similarity measure. A downside of this approach is that errors in the first iteration, which do not account for occlusions, can be hard to recover from.

The previous experiment did not take occlusions into account. In a second experiment, we pitch this state-of-the-art iterative technique against our segmentation-aware, single-shot approach. We let the Daisy stereo algorithm run for 5 iterations and show the results in Fig. 13. The performance of SSID with “Eigen” embeddings is comparable or superior to that of Daisy for most baselines—we achieve this in a single step, and without relying on calibration data to enforce rotation invariance. Additionally, note that we set the λ parameter of Eq. (10) on the motion experiments, and do not adjust them for a different problem: stereo. Figure 14 shows the depth estimates at two different baselines (image pairs {5,3} and {7,3}, reconstructed over 3)—the reference frame (3) is shown in Fig. 7.

4.4 Computational Requirements

The cost of computing dense SIFT descriptors [45] for an image of size 320×240 is under 1 second (MATLAB/C code). SLS (MATLAB) requires ~ 21 min. SID (a non-optimized MATLAB/C hybrid) requires ~ 71 s. SSID requires ~ 81 s, in addition to the extraction of the masks. Note that for all the experiments in this chapter we compute the “Eigen”/“SoftMask” embeddings at the original resolution (e.g., 640×480) before downscaling the images. The “SoftMask” embeddings (MATLAB) require ~ 7 s per image and the “Eigen” embeddings (MATLAB/C hybrid) ~ 280 s. Structured forest boundaries can be computed at multiple frames per second. The computational cost of matching two images with the SIFT-flow framework depends on the size of the descriptors, varying from ~ 14 s for SIFT (the smallest) to ~ 80 s for SID/SSID, and ~ 10 min for SLS-“paper” (the largest).

5 Summary and Future Work

In this work we propose a method to address background variability at the descriptor level, incorporating segmentation data into their construction. Our method is general, simple, and carries a low overhead. We use it to obtain segmentation-aware descriptors with increased invariance properties, which are of the same form as their original counterparts, and can thus be plugged into any descriptor-based application with minimal adjustments. We apply our method to SIFT and to SID descriptors, obtaining with the latter dense descriptors that are simultaneously invariant to scale, rotation, and background variability.

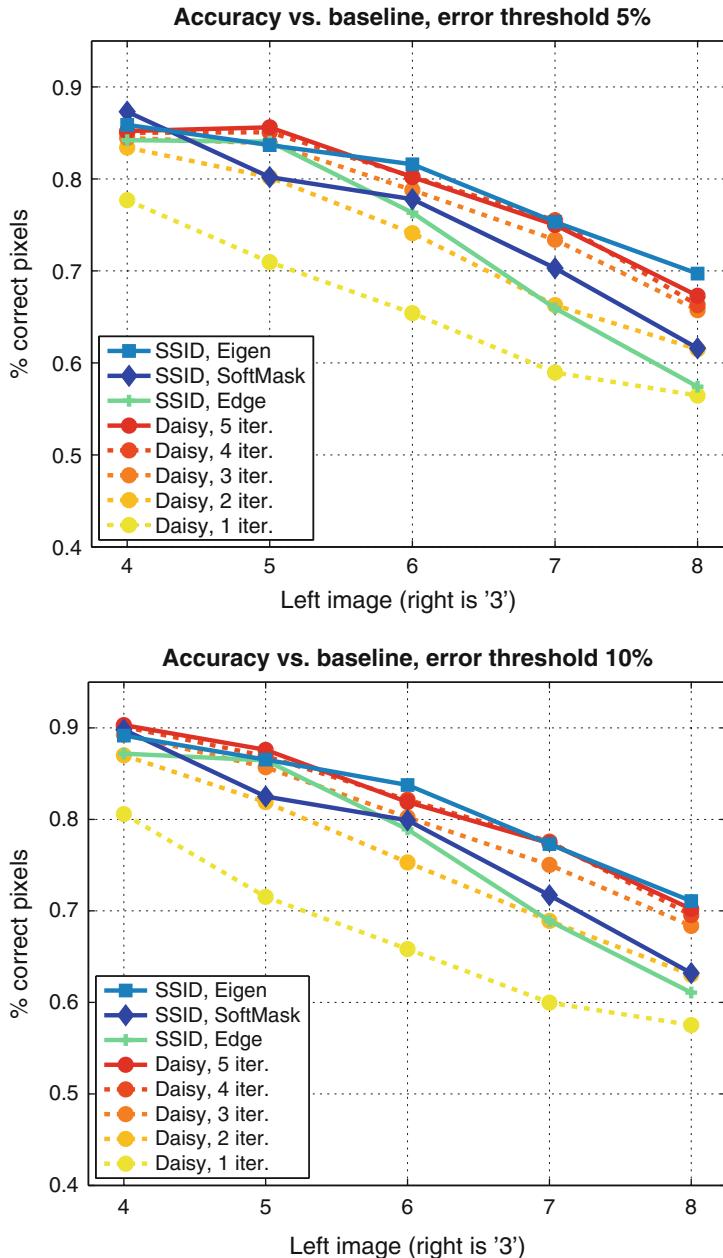


Fig. 13 Accuracy of the iterative approach to occlusion estimation of [40] and our segmentation-based, single-shot approach at different baselines for two error thresholds (expressed as a fraction of the scene range)

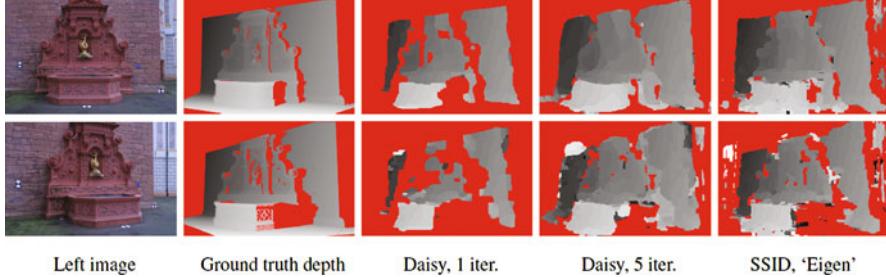


Fig. 14 *First column:* we compute depth maps for image pairs {5, 3} (top) and {7, 3} (bottom) of [38], over the “right” viewpoint. Occluded pixels are marked in red. We show images 5 or 7–3 is pictured in Fig. 7. *Second column:* ground truth depth maps. *Third and fourth columns:* iterations #1 and #5 for Daisy with latent occlusion estimation. *Fifth column:* single-shot reconstruction for SSID with “Eigen” embeddings

We demonstrate that our approach can deal with background changes in large displacement motion, and with occlusions in wide-baseline stereo. For stereo, we obtain results with SID comparable to the mask-based, state-of-the-art latent occlusion estimation of Daisy [40]—we do so without relying on calibration data to enforce rotation invariance; and in a single step, rather than with iterative refinements. While similar in spirit (both “gate” the features to achieve invariance against background variability), our method is also applicable to the case where a *single image* is available.

Regarding future work, we can identify at least two directions for extending our work. First, the segmentation-aware SID suffers from high dimensionality, but is likely very redundant. This shortcoming could be addressed with spectral compression and also with metric learning [39]. The latter proved able to both drastically reduce dimensionality problems and increase the discriminative power of descriptors at the same time. Second, both of the applications where we assess our segmentation-aware descriptors involve shots of the same scene which differ in time, or viewpoint, but contain identical object instances. Our more recent work in [44] extends the segmentation-aware feature extraction technique to Histogram-of-Gradient (HOG) features and Deformable Part Models by relying on superpixels, but we believe this is only a starting point for leveraging segmentation in feature extraction for recognition—the introduction of segmentation in convolutional network classifiers is one of our current research directions.

Acknowledgements This work has been partially funded by the Spanish Ministry of Economy and Competitiveness under project RobInstruct TIN2014-58178-R and ERA-Net Chisera project ViSen PCIN-2013-047; by EU projects AEROARMS H2020-ICT-2014-1-644271, MOBOT FP7-ICT-2011-600796, and RECONFIG FP7-ICT-600825; and by grant ANR-10-JCJC-0205 (HiCoRe).

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* **15**(6), 1373–1396 (2003)
2. Berg, A.C., Malik, J.: Geometric blur for template matching. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, vol. 1. IEEE, New York (2001)
3. Borgefors, G.: Distance transformations in digital images. *Comput. Vis. Graphics Image Process.* **34**(3), 344–371 (1986)
4. Bovik, A.C., Clark, M., Geisler, W.S.: Multichannel texture analysis using localized spatial filters. *Trans. Pattern Anal. Mach. Intell.* **12**(1), 55–73 (1990)
5. Brox, T., Malik, J.: Berkeley motion segmentation dataset. http://imb.informatik.uni-freiburg.de/resources/datasets/moseg_en.html (2010)
6. Casasent, D., Psaltis, D.: Position, rotation, and scale invariant optical correlation. *Appl. Opt.* **15**(7), 1795–1799 (1976)
7. Deriche, R.: Using Canny's criteria to derive a recursively implemented optimal edge detector. *Int. J. Comput. Vis.* **1**(2), 167–187 (1987)
8. Dice, L.R.: Measures of the amount of ecologic association between species. *Ecology* **26**(3), 297–302 (1945)
9. Dollár, P., Zitnick, C.L.: Structured forests for fast edge detection. In: Proceedings of the International Conference on Computer Vision, pp. 1841–1848. IEEE, New York (2013)
10. Freeman, W.T., Adelson, E.H.: The design and use of steerable filters. *Trans. Pattern Anal. Mach. Intell.* **13**(9), 891–906 (1991)
11. Fulkerson, B., Vedaldi, A., Soatto, S.: Localizing objects with smart dictionaries. In: European Conference on Computer Vision, pp. 179–192. Springer, New York (2008)
12. Hassner, T., Mayzels, V., Zelnik-Manor, L.: On SIFTs and their scales. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 1522–1528. IEEE, New York (2012)
13. Kokkinos, I., Yuille, A.: Scale invariance without scale selection. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE, New York (2008)
14. Kokkinos, I., Bronstein, M., Yuille, A.: Dense scale invariant descriptors for images and surfaces (2012). INRIA Research Report 7914
15. Kokkinos, I., Bronstein, M.M., Litman, R., Bronstein, A.M.: Intrinsic shape context descriptors for deformable shapes. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 159–166. IEEE, New York (2012)
16. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. *Trans. Pattern Anal. Mach. Intell.* **28**(10), 1568–1583 (2006)
17. Leordeanu, M., Sukthankar, R., Sminchisescu, C.: Efficient closed-form solution to generalized boundary detection. In: European Conference on Computer Vision, pp. 516–529. Springer, New York (2012)
18. Liu, C., Yuen, J., Torralba, A.: SIFT flow: Dense correspondence across scenes and its applications. *Trans. Pattern Anal. Mach. Intell.* **33**(5), 978–994 (2011)
19. Liu, K., Skibbe, H., Schmidt, T., Blein, T., Palme, K., Brox, T., Ronneberger, O.: Rotation-invariant HOG descriptors using Fourier analysis in polar and spherical coordinates. *Int. J. Comput. Vis.* **106**(3), 342–364 (2014)
20. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **60**(2), 91–110 (2004)
21. Maire, M., Arbeláez, P., Fowlkes, C., Malik, J.: Using contours to detect and localize junctions in natural images. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. IEEE, New York (2008)
22. Maire, M., Yu, S.X., Perona, P.: Object detection and segmentation from joint embedding of parts and pixels. In: Proceedings of the International Conference on Computer Vision, pp. 2142–2149. IEEE, New York (2011)
23. Mallat, S.: Zero-crossings of a wavelet transform. *Trans. Inf. Theory* **37**(4), 1019–1033 (1991)
24. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Van Gool, L.: A comparison of affine region detectors. *Int. J. Comput. Vis.* **65**(1–2), 43–72 (2005)

25. Nowak, E., Jurie, F., Triggs, B.: Sampling strategies for bag-of-features image classification. In: European Conference on Computer Vision, pp. 490–503. Springer, New York (2006)
26. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. Trans. Pattern Anal. Mach. Intell. **12**(7), 629–639 (1990)
27. Porat, M., Zeevi, Y.Y.: The generalized Gabor scheme of image representation in biological and machine vision. Trans. Pattern Anal. Mach. Intell. **10**(4), 452–468 (1988)
28. Ren, X., Malik, J.: Learning a classification model for segmentation. In: Proceedings of the International Conference on Computer Vision, pp. 10–17. IEEE, New York (2003)
29. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Phys. D: Nonlinear Phenom. **60**(1), 259–268 (1992)
30. Schmid, C., Mohr, R.: Local grayvalue invariants for image retrieval. Trans. Pattern Anal. Mach. Intell. **19**(5), 530–534 (1997)
31. Schmidt, U., Roth, S.: Learning rotation-aware features: from invariant priors to equivariant descriptors. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 2050–2057. IEEE, New York (2012)
32. Shechtman, E., Irani, M.: Matching local self-similarities across images and videos. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE, New York (2007)
33. Shi, J., Malik, J.: Normalized cuts and image segmentation. Trans. Pattern Anal. Mach. Intell. **22**(8), 888–905 (2000)
34. Simonyan, K., Vedaldi, A., Zisserman, A.: Descriptor learning using convex optimisation. In: European Conference on Computer Vision, pp. 243–256. Springer, New York (2012)
35. Simonyan, K., Vedaldi, A., Zisserman, A.: Learning local feature descriptors using convex optimisation. Trans. Pattern Anal. Mach. Intell. **12**, 25–70 (2014)
36. Stein, A., Hebert, M.: Incorporating background invariance into feature-based object recognition. In: Application of Computer Vision, vol. 1, pp. 37–44. IEEE, 7th IEEE Workshop on Applications of Computer Vision (WACV), New York (2005)
37. Strecha, C., Tuytelaars, T., Van Gool, L.: Dense matching of multiple wide-baseline views. In: Proceedings of the International Conference on Computer Vision, pp. 1194–1201. IEEE, New York (2003)
38. Strecha, C., von Hansen, W., Van Gool, L., Fua, P., Thoennessen, U.: On benchmarking camera calibration and multi-view stereo for high resolution imagery. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 1–8. IEEE, New York (2008)
39. Strecha, C., Bronstein, A.M., Bronstein, M.M., Fua, P.: LDA-hash: improved matching with smaller descriptors. Trans. Pattern Anal. Mach. Intell. **34**(1), 66–78 (2012)
40. Tola, E., Lepetit, V., Fua, P.: Daisy: An efficient dense descriptor applied to wide-baseline stereo. Trans. Pattern Anal. Mach. Intell. **32**(5), 815–830 (2010)
41. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proceedings of the International Conference on Computer Vision, pp. 839–846. IEEE, New York (1998)
42. Tron, R., Vidal, R.: Hopkins 155 dataset. <http://www.vision.jhu.edu/data.htm> (2007)
43. Trulls, E., Kokkinos, I., Sanfeliu, A., Moreno-Noguer, F.: Dense segmentation-aware descriptors. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 2890–2897. IEEE, New York (2013)
44. Trulls, E., Tsogkas, S., Kokkinos, I., Sanfeliu, A., Moreno-Noguer, F.: Segmentation-aware deformable part models. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 168–175. IEEE, New York (2014)
45. Vedaldi, A., Fulkerson, B.: An Open and Portable Library of Computer Vision Algorithms, <http://www.vlfeat.org> (2008)
46. Winder, S., Hua, G., Brown, M.: Picking the best daisy. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, pp. 178–185. IEEE, New York (2009)
47. Wolberg, G., Zokai, S.: Robust image registration using log-polar transform. In: International Conference on Pattern Recognition, vol. 1, pp. 493–496. IEEE, New York (2000)
48. Yao, J., Cham, W.K.: 3D modeling and rendering from multiple wide-baseline images by match propagation. Signal Process. Image Commun. **21**(6), 506–518 (2006)

SIFTpack: A Compact Representation for Efficient SIFT Matching

Alexandra Gilinsky and Lihi Zelnik-Manor

Abstract Computing distances between large sets of SIFT descriptors is a basic step in numerous algorithms in computer vision. When the number of descriptors is large, as is often the case, computing these distances can be extremely time consuming. We propose the SIFTpack: a compact way of storing SIFT descriptors, which enables significantly faster calculations between sets of SIFTs than the current solutions. SIFTpack can be used to represent SIFTs densely extracted from a single image or sparsely from multiple different images. We show that the SIFTpack representation saves both storage space and run time, for both finding nearest neighbors and computing all distances between all descriptors. The usefulness of SIFTpack is demonstrated as an alternative implementation for K -means dictionaries of visual words and for image retrieval.

1 Introduction

In numerous applications in computer vision a basic building block is the computation of distances between sets of SIFT descriptors [32]. Some applications require finding the nearest match for each descriptor, e.g., image alignment, scene classification, and object recognition [16, 29, 36]. In other cases one needs to compute all the distances between all the SIFTs, e.g., when constructing affinity matrices for image segmentation [4], co-segmentation [23, 24], or self-similarity [46]. As the number of descriptors increases, the computation time of these distances becomes a major consideration in the applicability of the algorithm.

Previous solutions to reduce the runtime can be categorized into four different approaches. The first approach reduces the dimensionality of the descriptors, thus decreasing the computation time of each distance calculation [25, 35, 55]. This comes at the cost of loss of accuracy due to the reduced dimension. The second approach goes further by hashing the descriptors into a binary code, or directly creating binary descriptors [10, 38, 44, 48, 51]. Using binary descriptors enables

A. Gilinsky • L. Zelnik-Manor (✉)
Technion Israel Institute of Technology, Haifa 32000, Israel
e-mail: sashkagil@gmail.com; lihi@ee.technion.ac.il

matching with a Hamming distance which is much faster to calculate than the standard L_2 distance. Some of these methods result in descriptors which are not invariant to rotation or scale changes [10, 44], others use supervised learning, thus requiring a training set and designed to a specific task [48, 51]. The third approach reduces the number of descriptors by using sparse sampling of the image [49, 56]. This could compromise accuracy, as it was repeatedly shown that dense sampling yields better results in recognition [16, 29, 36]. Furthermore, in many applications, the obtained set of descriptors could still be very large, e.g., when reconstructing 3D scenes using thousands of images. Hence, efficient methods for computing the distances are still required. Finally, the fourth approach is based on approximate solutions [3, 14, 22]. These reduce runtime significantly; however, they are relevant only for the nearest-neighbor case and become inefficient when all distances need to be computed. Algorithms for approximate matching across images are also highly efficient [6, 19, 28, 37], but are limited to dense matching across a pair of images and most of them cannot be applied for matching SIFTs.

As an alternative solution, we propose the SIFTpack: a compact form for storing a set of SIFT descriptors that reduces both storage space and runtime when comparing sets of SIFTs. Our key idea is to exploit redundancies between descriptors to store them efficiently in an image-like structure. Redundancies can occur, for example, when two descriptors are computed over overlapping image regions. In this case the descriptors have a shared part that does not need to be stored twice. The SIFTpack construction identifies such redundancies and hence saves in storage space. We show how SIFTpack can be used to compactly represent descriptors extracted densely from a single image. We further suggest an algorithm for constructing SIFTpacks for descriptors extracted sparsely from one or many images. Such SIFTpacks can serve as an efficient alternative to SIFT dictionaries.

The key contribution of this work is a significant reduction in runtime when computing distances between sets of SIFTs. The speedup is due to two reasons. First, we avoid repeated calculations of the same distances. Second, since the SIFTs are stored in an image-like form, we can utilize existing efficient algorithms for fast matching between images. We suggest such solutions for both nearest-neighbor matching and all-distances calculation. The proposed solutions are shown to be useful for bag-of-words (BoW) models and for image retrieval. Although we demonstrate our method on SIFT descriptors only, it can be easily adjusted to other, similar to SIFT descriptors, like HoG [13] or SURF [8].

The rest of the chapter is organized as follows. Related work is reviewed in Sect. 2. Next we describe the SIFTpack construction for dense-SIFT and for an arbitrary set of SIFTs in Sect. 3. Efficient algorithms for matching sets of SIFTs, based on SIFTpack, are presented in Sect. 4. The usefulness of SIFTpack as an efficient alternative to K -means dictionaries of visual words is presented in Sect. 5 and its application to image signatures is outlined in Sect. 6. Finally, we conclude in Sect. 7.

This chapter extends the earlier work appeared in [18].

2 Related Work

In this section we review previous work related to our work. Since our research deals with a compact way of representing SIFT features which enables faster calculations between them, we start by reviewing applications that could benefit from the proposed representation. The applications can be roughly divided into three categories: (1) applications that compute dense matching between a pair of images (or between an image to itself), (2) applications that match dense descriptors of one image to a general set of descriptors (or a dictionary), and (3) applications that require matching between two general sets of descriptors (or two dictionaries). Part of the reviewed applications compute all (or multiple) distances between the descriptors while others settle for finding only the nearest neighbor. Our algorithms for efficient SIFT matching rely on recent developments in dense matching between a pair of images. Hence, we further review related work on nearest-neighbor field computation. Finally, we complete the review by discussing related work on compact representations of image descriptors.

2.1 Applications of SIFT Matching

2.1.1 Dense Matching Between Images

Many image segmentation (and co-segmentation) algorithms rely on computation of an affinity matrix, which contains the affinities between all pairs of descriptors in an image (or a pair of images) [23, 24]. Since these methods compute distances between dense descriptors, reducing runtime of this step is highly important. In [4] a different approach is proposed for image segmentation based on a definition for a “good segment” as one that is easy to compose from its own parts while hard to compose from the rest of the image. This definition as well requires many calculations of distances between descriptors in a dense manner. Finally the calculation of all (or multiple) distances between dense sets of descriptors is also required for building “Self-Similarity” descriptors [15, 46].

Finding nearest neighbors between dense descriptors of a pair of images is also a very common step in many applications. For example, in [6, 7] the nearest-neighbor field is utilized for denoising, clone detection, symmetry detection, and object detection. In [31] the nearest-neighbor field between SIFTs is used for alignment, motion hallucination, and image retrieval. Many of these applications should run in real time; hence, the computation time of the nearest-neighbor field is crucial.

Previous attempts to reduce the runtime amount to dimensionality reduction [50] and settling for sparse matching [49, 56]. Both approaches result in a reduction in matching accuracy.

2.1.2 Matching an Image to a Dictionary

A very common approach to many computer vision applications is using the BoW model, e.g., for recognition and classification [1, 17, 29, 30, 34, 36] or for image retrieval [40, 41, 52]. The BoW approach involves two main steps: (1) creating a dictionary from a set of descriptors and (2) representing each image by the corresponding histogram over this dictionary. To generate the histogram, it is necessary to find for each descriptor in the image its nearest neighbor in the dictionary. Therefore, the histogram construction step could be highly time consuming when the dictionary is large. This problem is even more prominent when using dense descriptors, as recommended for various scenarios [16, 17, 29, 36, 54]. To address this and reduce the run time, in this work we present an approach for constructing and representing SIFT dictionaries such that faster matching can be performed.

2.1.3 Matching Dictionaries

In image retrieval typically each image is represented by a compact signature, and retrieval is performed by comparing the query signature to the database signatures. In [20, 26, 27, 45] it is proposed to use a dictionary of features as the image signature. The distance between two dictionaries is computed as the sum of distances between each word in the query dictionary and its nearest neighbor in the database image signature. Since the database is typically very large, reducing the runtime of matching dictionaries is essential. Furthermore, as mentioned in [42], storing a database of dictionaries could require a lot of memory space which may be an obstacle. Hence, a compact way of storing such dictionaries is needed. Ramakrishnan et al. [42] present an approach to reducing the number of dictionaries to be stored for classification purposes. In this work we propose instead a compact representation for storing dictionaries more efficiently (in terms of space).

2.2 Algorithms for Nearest-Neighbor Matching Between Images

As previously discussed in Sect. 2.1, many applications require matching of SIFT features. As a result, several approaches for speeding up the matching have been proposed in recent years. These approaches vary in their applicability. When the goal is nearest-neighbor matching between arbitrary sets of descriptors, the popular approaches are based on kd-trees [3], Locality-Sensitive Hashing [14], or the more recent Product Quantization technique [22]. More efficient solutions exist when the descriptors are taken densely from an image, i.e., for computing nearest neighbor fields. In this case the coherency between nearby patches can be harnessed to reduce

the matching runtime [6, 7]. It is important to note that several of the methods for computing the nearest-neighbor field are limited to matching image patches and cannot be easily adapted to matching SIFT descriptors, e.g., [19, 28]. This limitation is due to their reliance on the Walsh–Hadamard decomposition of the patches, which does not apply to SIFT.

The case of computing all distances between all patches of an image has been addressed by Olonetsky and Avidan [37]. There an efficient solution has been proposed based on the integral image representation [11]. The algorithm of [37] is highly efficient, but it is limited to dense matching between patches of two images, and cannot be applied to matching descriptors.

2.3 *Dictionaries of Image Descriptors*

The common approach to storing SIFTs is straightforward: the descriptors are simply stored in an array. When a more compact representation is sought the popular solution is to quantize the descriptors using K -means [12, 20, 29, 45]. The quantized descriptors are then stored in an array. A key observation we make is that storing SIFTs in an array is wasteful and more compact solutions should be designed.

The SIFTpack representation we propose is inspired by the “Image Signature” framework of [2, 9]. There, the patches of an image are represented by a sparsifying dictionary that is forced to have an image form. Similarly to [2, 9], we construct an image-like dictionary. Differently from them, we use this to represent SIFT descriptors. We show that such dictionaries are significantly more efficient in terms of space and time.

3 The SIFTpack Representation

The SIFTpack construction that we propose is based on two observations. The first is that different SIFT descriptors could still have shared components. This could occur, for example, when the regions over which they were computed have a highly similar appearance. Storing these shared components more than once is redundant and inefficient. The second observation we make is that highly efficient solutions have been proposed for computing nearest-neighbor fields across images. This motivates us to seek an image-like representation for a set of SIFTs. In what follows we suggest a representation that builds upon these two observations. We first describe how SIFTpack is constructed for dense-SIFT and then continue to present an algorithm for constructing a SIFTpack for an arbitrary set of SIFTs.

3.1 SIFTpack for Dense-SIFT

We start by briefly reviewing the structure of the SIFT descriptor [32]. The SIFT at a pixel is based on the gradients within a neighborhood around it. The neighborhood is divided into $4 \times 4 = 16$ subregions, and an 8-bin histogram of weighted gradients is computed for each subregion. The weights are set according to the distance from the center pixel. The 16 histograms are then concatenated into a 128-dimensional vector. Typically, SIFTS are stored in a $128 \times M$ array, where M is the number of descriptors.

To efficiently store dense-SIFT descriptors [53] we start by computing the SIFTS on a grid with an n pixel gap. We set the scale such that each descriptor is computed over a region of $4n \times 4n$ pixels, i.e., each 8-bin histogram is computed over a sub-region of size $n \times n$ pixels. As illustrated in Fig. 1, in this construction the spatial regions of neighboring descriptors overlap and the subregions are aligned. To simplify the presentation we will assume for the time being that no weighting is applied to the pixels when constructing the descriptors. Later on we will remove this assumption. Without the weighting, two descriptors with overlapping regions will have at least one shared histogram. In fact, all the descriptors including a certain subregion will share its corresponding gradient histogram. While the standard approach stores this histogram multiple times, once for each descriptor, we wish to store it only once.

To enable a compact and image-like representation we start by reshaping the extracted SIFT descriptors. As illustrated in Fig. 2, we reshape each 128-dimensional SIFT vector into a 3D array of dimensions $4 \times 4 \times 8$. We think of

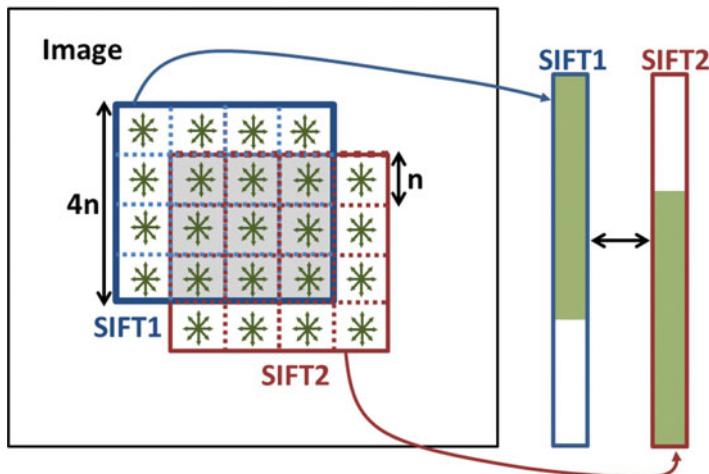


Fig. 1 Redundancy in overlapping SIFTs: when two SIFT descriptors are computed over overlapping image regions the shared region (marked in gray) could result in common SIFT values in the descriptors (marked in green). The joint values will appear at different entries of the vectors and will be stored twice by the standard approach of keeping SIFTS in an array

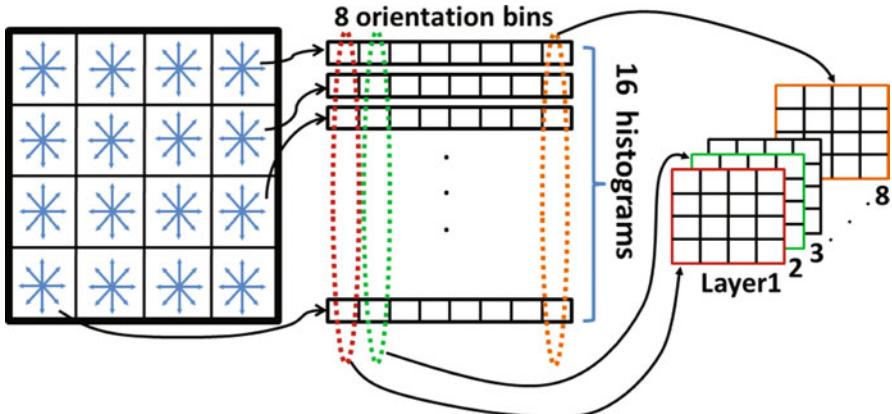


Fig. 2 Reshaping SIFT: SIFTpack requires reshaping the 128-dimensional SIFT descriptor into 8 layers of 4×4 pixels each

this 3D array as an image with 4×4 pixels and 8 layers. Each layer corresponds to one of the 8 bins of the histograms, i.e., each layer captures one of the eight gradient orientations considered by SIFT. Each “pixel” corresponds to one of the 16 histograms used to construct the descriptor.

The key idea behind our construction is that after the reshape step, the shared histograms of two neighboring descriptors correspond to shared “pixels.” Therefore, to store the descriptors compactly, we simply place them next to each other, with overlaps, according to their original position in the image. This is illustrated in Fig. 3. The SIFTpack storage space is 16 times smaller than that of the conventional approach of saving all SIFTs in an array. The construction time is linear in the number of descriptors.

To complete our construction, we next remove the temporary assumption of no pixel weighting. In practice, Gaussian weighting is applied to the region over which the descriptor is computed, thus two overlapping SIFTs do not share the exact same entry values. Therefore, we average the values of all SIFTs that overlap, i.e., we store only one, averaged version, of all SIFT entries that describe the same spatial bin. Figure 4 displays the layers of a SIFTpack constructed from dense-SIFT.

To examine the implications of averaging the SIFT values, we performed the following experiment. We used SIFT flow [31] to compute the flow field between the pairs of images of the Middlebury dataset [5] and on the “light” dataset of “Mikolajczyk” [35]. We performed this twice, once with the original SIFT descriptors and once while averaging overlapping descriptors. As shown in Table 1, not only does the averaging not interfere but it also slightly improves the correspondence accuracy and reduces errors. This is probably due to the enhanced robustness to noise when smoothing. For this experiment we used normalized values of SIFT. While the averaging described above alters the normalization a bit, this is not significant as matching accuracy is not harmed.

Fig. 3 *SIFTpack*: an 8-layer image, where each 4×4 patch corresponds to a single SIFT descriptor. Histograms that are shared between descriptors are stored only once in this construction, e.g., the gray area is mutual to the blue and red SIFTS

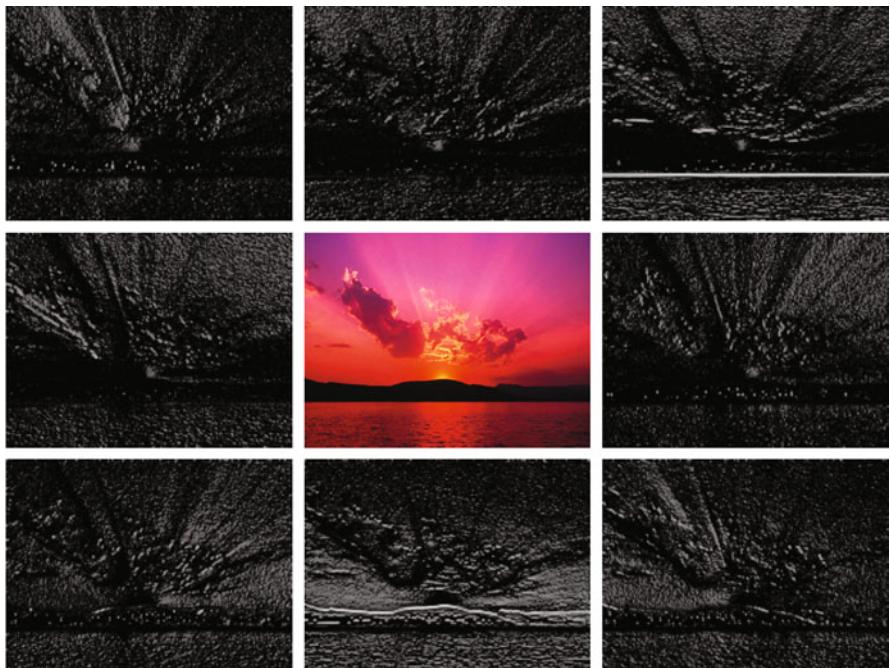
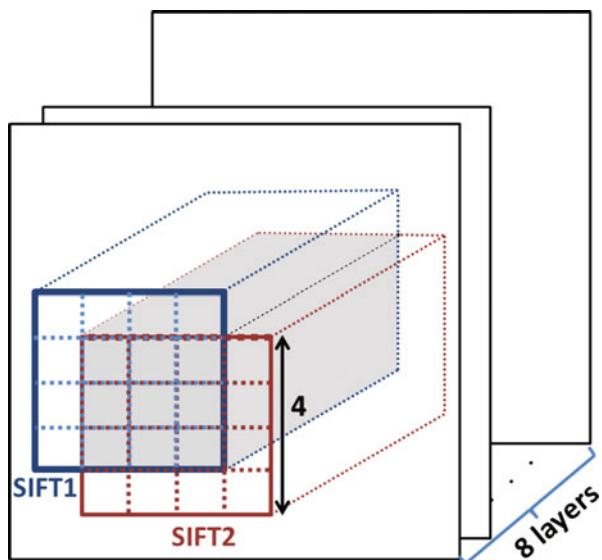


Fig. 4 Visualizing *SIFTpack*: the eight grayscale images are the layers of the *SIFTpack* of the center image. As expected, they capture gradients at different orientations. The *SIFTpack* layers are rescaled to the image size for visibility, but their original size is n times smaller in each axis

Table 1 Flow-field accuracy on Middlebury (a) and Mikolajczyk dataset (b): as can be seen from both tables, the flow fields computed by SIFT flow [31] are consistently more accurate when applied to descriptors packed in SIFTpack than the original SIFTs

(a)		
Name	Without averaging Error (pixels)	With averaging (SIFTpack) Error (pixels)
<i>Dimetrodon</i>	1.6104	1.16089
<i>Grove2</i>	1.7314	1.7297
<i>Grove3</i>	1.9034	1.8883
<i>Hydrangea</i>	0.6690	0.6446
<i>RubberWhale</i>	1.2247	1.2211
<i>Urban2</i>	1.5638	1.5302
<i>Urban3</i>	141.1214	22.5310
<i>Venus</i>	1.2661	1.2435

(b)		
Num	Without averaging Error (pixels)	With averaging (SIFTpack) Error (pixels)
1	1.5397	1.5239
2	1.9062	1.8774
3	1.9281	1.8884
4	1.2238	1.1979
5	2.0961	2.0459

This suggests that averaging overlapping SIFTs reduces noise and hence matching accuracy is improved

Bold values indicate the lower values among the two columns

Algorithm 1 SIFTpack for dense-SIFT

Input: Image

Compute dense-SIFT, each over a $4n \times 4n$ neighborhood, on a grid with n pixels spacing.

for all SIFTs **do**

- Reshape into a $4 \times 4 \times 8$ array.
- Place SIFT of pixel i,j at SIFTpack location $i/n, j/n$ (these are integer values due to the grid).
- Average all overlapping values.

end for

Output: SIFTpack

To summarize, our method for constructing SIFTpack for dense-SIFT is outlined in Algorithm 1.

3.1.1 SIFTpack for SIFTs Extracted at Each Pixel

Our construction described by Algorithm 1 is useful for SIFTs extracted on a grid with an n pixel gap. Suppose we want to pack SIFTs on a more dense grid, for instance each pixel. In this case we can use Algorithm 1 to pack sets of SIFTs with different offsets as follows. We create a SIFTpack for SIFTs extracted at each n^{th} pixel starting from the first pixel, then a SIFTpack for SIFTs extracted

Table 2 Flow-field accuracy on Middlebury when using SIFTpack for dense-SIFT taken at each pixel: the flow fields computed by SIFT flow [31] are consistently more accurate

<i>Name</i>	Without averaging	With averaging (SIFTpack)
	Error (pixels)	Error (pixels)
<i>Dimetrodon</i>	0.4188	0.4158
<i>Grove2</i>	0.5280	0.5237
<i>Grove3</i>	1.0591	1.0456
<i>Hydrangea</i>	0.4026	0.3759
<i>RubberWhale</i>	0.3868	0.3726
<i>Urban2</i>	1.0764	1.0710
<i>Urban3</i>	268.5910	53.2182
<i>Venus</i>	0.4857	0.4650

Bold values indicate the lower values among the two columns

at each n^{th} pixel starting from the second pixel, and so on. We create a SIFTpack for each offset in both axes, for example, if our cell size is 5×5 , then we will create 25 SIFTpacks, for five offsets in each direction. Table 2 shows the flow-field results for this case on the Middlebury dataset. Since the SIFTS are extracted at each pixel, the results are better than those of Table 1. Still, weighting the SIFTS improves the accuracy.

After we create a SIFTpack for each offset, we can obtain a single combined SIFTpack by merging these SIFTpacks together via interlacing. This way we can increase the resolution of each of the eight layers. An example result is shown in Fig. 5. As can be seen, when we create one SIFTpack from all the SIFTpacks of different offsets, the images in the layers remain similar to those shown in Fig. 4. The size of the SIFTpack layers though is now identical to the size of the image.

3.2 SIFTpack for a Set of SIFTS

Next, we suggest a similar construction for the case where SIFTS are extracted at isolated locations (typically interest points), possibly from multiple different images, of different scales and orientations. When the number of descriptors is large we expect to find many similarities between parts of them, e.g., when multiple descriptors include sub-regions of similar appearance. However, unlike the dense case, here we cannot tell a priori which descriptors have corresponding components.

To identify these repetitions and enable the construction of a compact SIFTpack we build upon the dictionary optimization framework of [2]. Given a set of M SIFT descriptors y_i , $i = 1, \dots, M$ we wish to find a SIFTpack S , of size $m \times m \times 8$, that forms a sparsifying dictionary for them. S is obtained by optimizing the following objective:

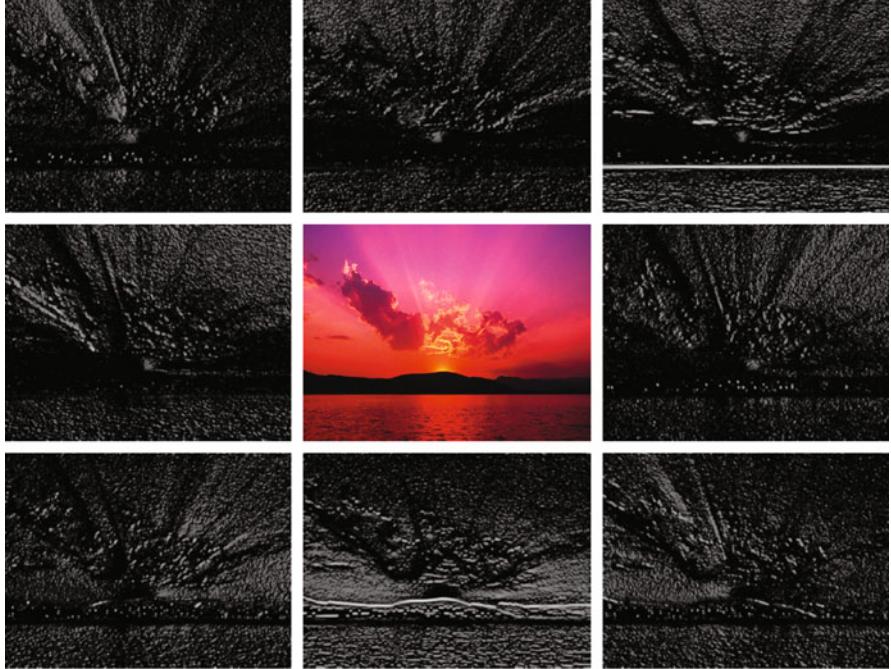


Fig. 5 Visualizing SIFTpack: the eight grayscale images are the layers of the SIFTpack of the center image. The SIFTs are extracted at each pixel; hence, here the SIFTpack layers are of the same size as the image

$$\begin{aligned} \min_{S,x} \quad & \sum_{i=1}^M \left\| y_i - \sum_{k=1}^m \sum_{l=1}^m x_{i[k,l]} C_{[k,l]} \hat{S} \right\|_F^2 \\ \text{s.t. } & \|x_i\|_0 \leq \theta, \quad i = 1, \dots, M, \end{aligned} \quad (1)$$

where \hat{S} is a vector obtained by column stacking the SIFTpack S , $C_{[k,l]}$ is an indicator matrix that extracts the 128-dimensional SIFT descriptor in location $[k, l]$ in S , and $x_{i[k,l]}$ are the corresponding sparse coding coefficients.

Problem (1) is not convex with respect to both S and x ; however, when fixing either one of them it is convex with respect to the other. Therefore, to find a local minimum of Problem (1) we iterate between optimizing for the sparse coefficients x via orthogonal matching pursuit (OMP) [33, 39] and optimizing for the SIFTpack S . In all our experiments we set the sparsity $\theta = 1$ and $x_{i[k,l]} = 1$ for only a single SIFTpack location $[k, l]$ and all other values are 0. In practice, this implies that the OMP finds for each original SIFT y_i the closest SIFT within the current SIFTpack. The SIFTpack update step is

$$\hat{S} = \mathbf{R}^{-1} p, \quad (2)$$

Algorithm 2 SIFTpack for a set of SIFTS**Input:**

- A set of M SIFTS
- The desired SIFTpack size m

Initialization

- Construct a SIFTpack of size $m \times m \times 8$ using Algorithm 1 on an arbitrary image.

repeat

- Assign each SIFT in the input set to its most similar SIFT in the current SIFTpack.
- Update the SIFTpack by averaging SIFTS assigned to overlapping SIFTpack entries.

until Objective of Eq. (1) converges.

Output: SIFTpack

where

$$\mathbf{R} = \sum_{i=1}^M \left(\sum_{k=1}^m \sum_{l=1}^m x_{i[k,l]} C_{[k,l]} \right)^T \left(\sum_{k=1}^m \sum_{l=1}^m x_{i[k,l]} C_{[k,l]} \right),$$

$$p = \sum_{i=1}^M \left(\sum_{k=1}^m \sum_{l=1}^m x_{i[k,l]} C_{[k,l]} \right)^T y_i.$$

It can be shown that since we use sparsity $\theta = 1$ and set $x_{i[k,l]} = 1$, this is equivalent to the averaging of all SIFTS that were assigned to the same location in S .

The above optimization requires initialization. When the SIFTpack represents a set of SIFTS extracted at interest points of multiple different images we have found empirically that best results were obtained when initializing with a SIFTpack constructed from a dense-SIFT of a randomly chosen image, as described in the previous section. Note that the initial image should be textured; otherwise, the results will be inferior.

Our method for constructing SIFTpack for any set of SIFTS is summarized in Algorithm 2. As can be noticed, the set of SIFTS is treated as image patches (with eight channels), and the SIFTpack is a dictionary in the form of an image. This is similar to packing image patches into an image signature [2] or epitome [9].

Figure 6a visualizes a SIFTpack constructed by Algorithm 2. The final SIFTpack is of size $m \times m \times 8$, where m is a user-defined parameter. The smaller the m is, the more storage space is saved. However, this comes at the cost of lower accuracy, as illustrated in Fig. 6b. When the SIFTS are taken from multiple images, and $m \ll M$, a SIFTpack constructed by Algorithm 2 can be viewed as a dictionary of visual words. It differs from the standard dictionaries in exploiting redundancies between visual words to store the dictionary more compactly. Note the resemblance between Algorithm 2 and the K -means algorithm, as their complexity is the same and they both are usually intended to be executed offline, during the dictionary construction phase.

We should also point that theoretically, during the optimization process, some cells can stay inactive. This can happen when a cell was not found as a nearest

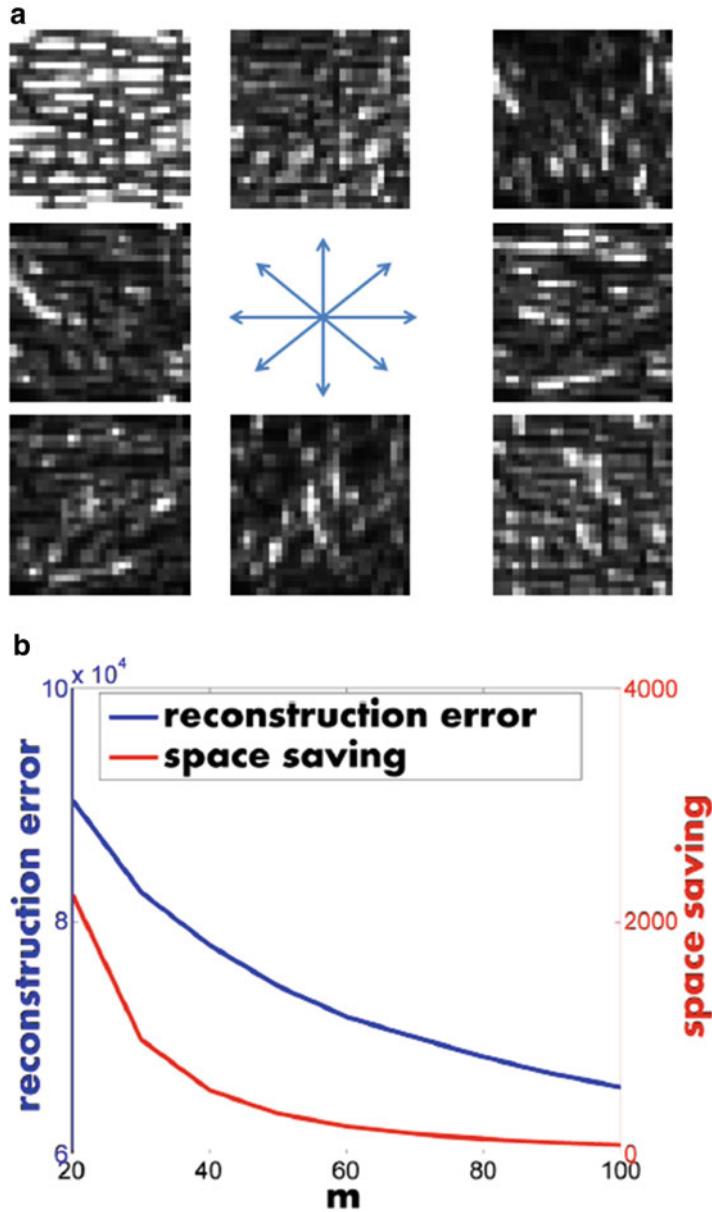


Fig. 6 SIFTpack for a set of SIFTs: (a) the eight layers of a SIFTpack constructed of $\sim 50,000$ SIFTs extracted at interest points from different images with different scales. (b) The average representation error (blue) and saving in storage space (red) as a function of the SIFTpack size m . The smaller the SIFTpack, the more space we save, at the price of a larger representation error

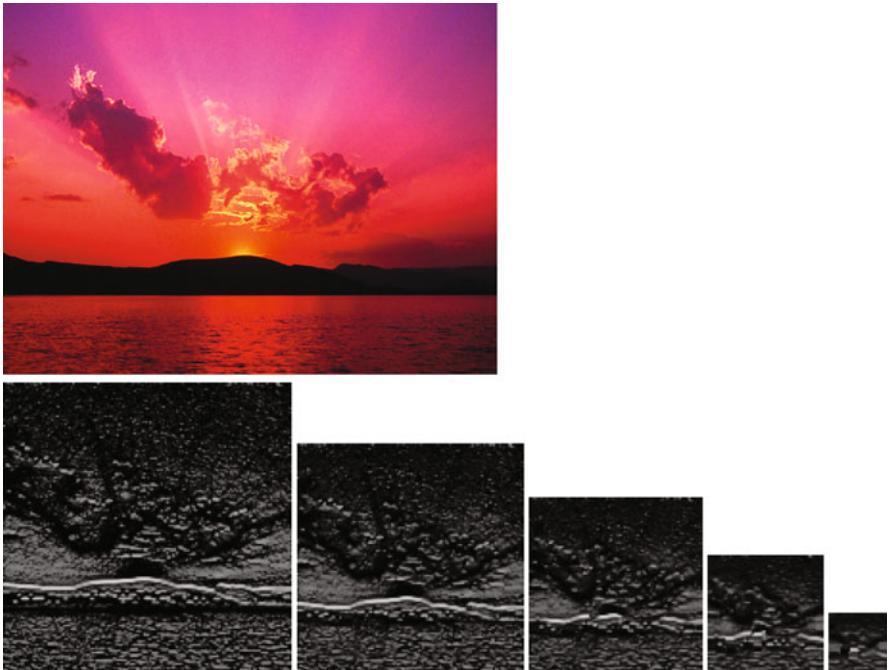


Fig. 7 An illustration of gradual resizing for one layer of SIFTpack, on a “sunset” image

neighbor to anyone of the SIFTs in the set. Practically, in all our experiments, when $m \ll M$, this phenomenon never happened. This fact confirms our assumption that there are spatial redundancies even in a random large set of SIFTs, hence they can be put in an image like structure.

Since the proposed optimization framework is applicable to any arbitrary set of SIFTs it is also applicable to dense-SIFT, leading to further reduction in the required storage space, as compared to that obtained by Algorithm 1. Inspired by Simakov et al. [47], when the input is dense-SIFT we iteratively reduce the size of the SIFTpack to 95 % of its previous dimension, initializing each iteration with a resized version of the previous one. The iterations continue until the desired size is reached. This procedure is illustrated in Fig. 7 for one of the 8 layers. When performing gradual resizing, the initialization of Algorithm 2 is always very close to the input SIFTpack, resulting in a convergence to a better result. Hence, the advantage of the gradual resizing is a more compact SIFTpack for the same representation error (or a better representation error for the same SIFTpack size). On the down side, applying Algorithm 2 with gradual resizing is slower, and hence is not always preferable.

It is important to note that while Algorithm 2 is described for the L_2 distance between SIFTs, our framework is generic and can be applied to many other distance metrics. Exchanging the distance metric amounts to replacing the update step of Eq. (2) by an appropriate calculation for averaging overlapping SIFTs and exchanging stage 1 with regards to the new metric.

4 Efficient Matching Solutions

So far we have described algorithms for constructing a space efficient representation for multiple SIFTs, extracted from either one or multiple images. While saving in storage space is a desirable property, our main goal is to obtain a significant reduction in computation time as well. A main advantage of the SIFTpack is that it can be viewed as an 8-layer image; therefore, one can employ existing algorithms for efficient matching across images.

4.1 Computing All Distances

In applications such as image segmentation, co-segmentation, and self-similarity estimation one needs to compute all (or multiple) distances between all (or multiple) pairs of descriptors within a given set or across two sets of SIFTs. When the number of descriptors is large, computing all distances naively is highly time consuming: $O(M^2)$, where M is the number of descriptors.

Storing the descriptors in a SIFTpack enables a more efficient computation since we avoid redundant calculations. As described in Sect. 3, the SIFTpack stores joint descriptor parts only once; hence, they are used only once when computing distances. We next present an efficient algorithm, which makes use of the special image-like structure of the SIFTpack to avoid redundant calculations.

Let $S1$ and $S2$ denote the SIFTpacks constructed for two sets of SIFTs. In applications where distances are to be computed between the descriptors of a single set, we assign $S2 = S1$. We further denote by $S_{i,j}$ the SIFT descriptor in location i,j in S and define $S^{[k,l]}$ as the shift of S by k, l pixels in x, y directions, respectively.

Our approach for efficiently computing all distances between the descriptors of $S1$ and $S2$ is adopted from [37], where it was used to compare image patches. We use the integral image [11] to compute the distance between all pairs of descriptors in $S1$ and $S2$ that have the same location i,j . To compute distances between descriptors at different locations we loop through all shifts k, l of $S2$. Our algorithm can be summarized as follows:

Algorithm 3 loops through all possible shifts k, l , similarly to the naive solution. However, it computes the distances between all pairs of descriptors with a location shift k, l faster than the standard solution. The speedup is due to the integral image approach that is possible here since our SIFTpack has an image-like structure.

Figure 8 ascertains this via empirical comparison between the runtime (in seconds) of the naive approach and Algorithm 3. For this experiment we first extract the dense-SIFT [53] descriptors of images of varying sizes. We then compute the distances between all pairs of SIFTs within a radius R of each other, using the naive approach. Next, we construct a SIFTpack for each image, using Algorithm 1, and compute the distances between the same pairs of SIFTs using Algorithm 3. We use the “RetargetMe” dataset [43] together with some images collected by us that together consist of 90 images. We used different sizes for each image by applying

Algorithm 3 All distances between SIFTS

Input: SIFTpacks $S1$ and $S2$

for all shifts k, l do

- Compute the element-wise square difference
 $\Delta = (S1 - S2^{[k,l]})^2$
- Compute the Integral Image $F(\Delta)$, summing the 8 layers.
- The distance between $S1_{i,j}$ and $S2_{i,j}^{k,l}$ is equal to:
 $F(i,j) + F(i+3,j+3) - F(i+3,j) - F(i,j+3)$

end for

Output: Distances between all SIFTS

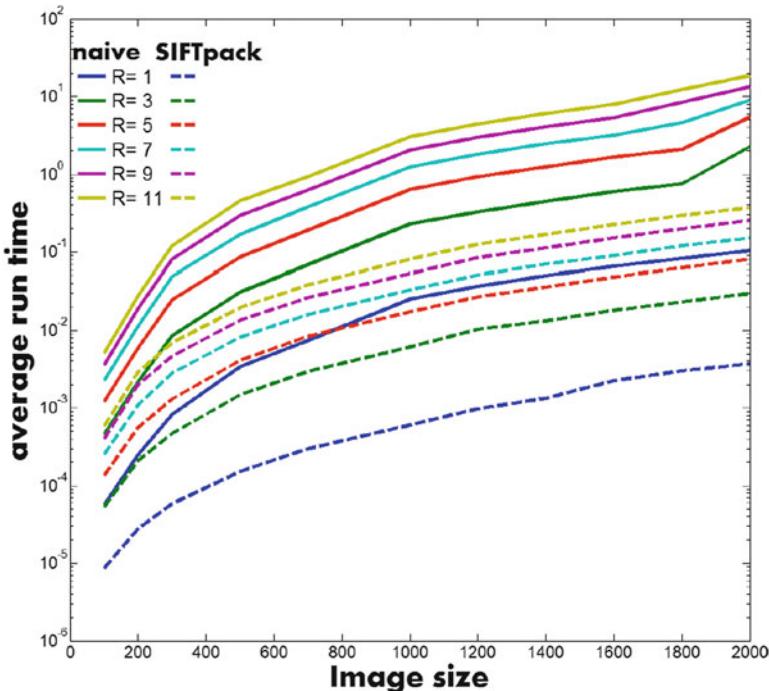


Fig. 8 Runtime saving by SIFTpack when computing multiple distances: the curves represent the average runtime for computing distances between all pairs of SIFTS of an image, within a radius R of each other. Solid curves correspond to the naive approach and dotted lines correspond to Algorithm 3. As can be seen, the standard approach is an order of magnitude slower than using SIFTpack and Algorithm 3. This holds for varying image sizes and different R values

resizing. For each image we repeated the distances computation 100 times and averaged the results. As can be seen, for all image sizes and for all radii R the reduction in runtime of the SIFTpack approach varies between one and two orders of magnitude. This experiment, as well as all others presented here, was performed on an i7 2.53 GHz Linux machine with 16Gb RAM, using one core. Similar results were obtained on i7 3.4 GHz, 16Gb Windows machine.

The above experiment shows the speedup that can be obtained for applications that require computing distances between dense descriptors, e.g., image segmentation and self-similarity. When one needs to compute all distances between two arbitrary sets of SIFTs, the SIFTpack construction needs to be performed via Algorithm 2, which is time consuming on its own. In such cases using our approach makes sense when the SIFTpack can be computed and stored a priori.

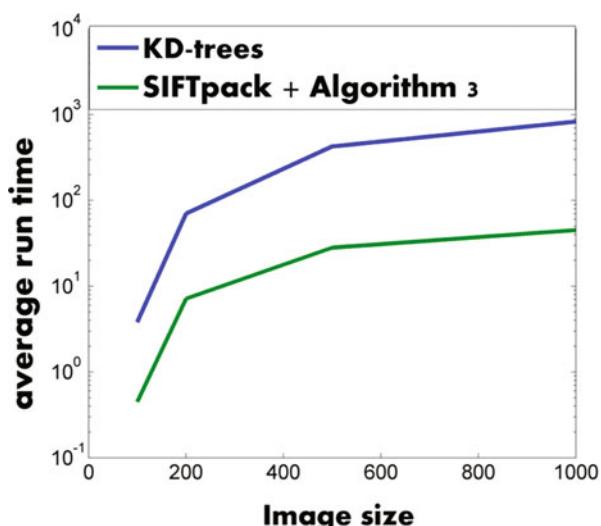
4.2 Exact Nearest-Neighbor Matching

When a single nearest neighbor is needed, the naive solution is to compute all distances and take the minimum. Alternatively, one could use more efficient tree-based methods such as that proposed in [3]. We propose to use SIFTpacks and Algorithm 3, with the slight modification that only the nearest neighbor is stored for each descriptor. We performed an experiment on the “VidPairs” dataset [28] which consists of 133 pairs of images. We used resized versions of each pair to test on images of different sizes. We computed the exact NN field between each pair of different sizes. Figure 9 shows that our solution is much faster than that of [3].

4.3 Approximate Nearest-Neighbor Matching

Due to the high computational demands of finding the exact nearest neighbor, more efficient approaches have been proposed that settle for finding the approximate nearest neighbor (ANN) [3, 14, 22]. These methods provide significant runtime

Fig. 9 Runtime saving by SIFTpack for exact nearest neighbor: computing the exact NN using SIFTpacks and Algorithm 3 is significantly faster than using kd-trees [3]



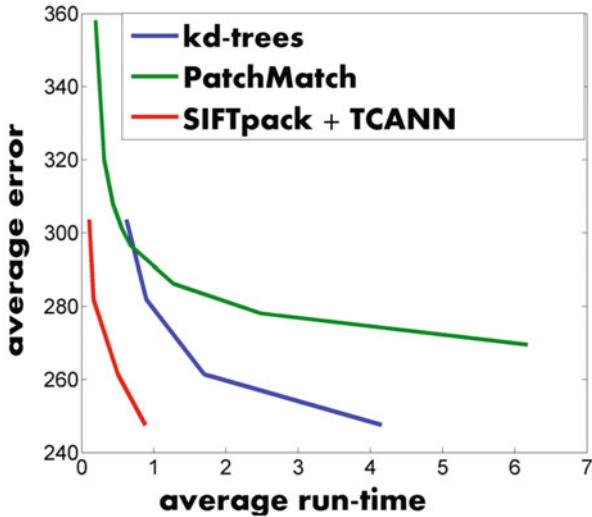


Fig. 10 Runtime saving by SIFTpack for ANN: computing the ANN using SIFTpack and TreeCANN leads to significantly lower errors and faster run time (in seconds) than both kd-trees and PatchMatch. These graphs are for images of size 800×800 , but similar results were obtained for other image sizes as well

reduction at the price of loss in accuracy. More recently it has been shown that even faster algorithms can be developed when computing ANN densely across images [6, 19, 28, 37]. These algorithms utilize the coherency between nearby image patches to speed-up the matching. Since the SIFTpack can be viewed as an 8-layer image, where each patch represents a SIFT descriptor, we can apply these algorithms to find ANN between SIFTS.

Figure 10 presents an empirical evaluation of the benefits of SIFTpack for computing ANN. Given a random pair of images and their corresponding dense-SIFT descriptors, we find for each descriptor in one image its ANN in the second image, using four methods: (1) SIFT flow [31], (2) kd-trees [3] computed over the set of SIFTS, (3) PatchMatch [6] applied to standard dense-SIFT, and (4) TreeCANN [37] applied to the SIFTpack. TreeCANN is one of the fastest algorithms for ANN that can be applied only for finding ANN between image patches. Note that the standard approaches to date are (1)–(3), while our proposed SIFTpack, being an image, enables method (4). Similarly to the exact NN experiment, we test the performances on the “VidPairs” dataset [28], repeating 100 times each NN field calculation and averaging. As can be seen, SIFTpack+TreeCANN significantly outperforms both Kd-Trees and PatchMatch in both accuracy and runtime. We have omitted the results of SIFT-flow as their quality was too low, probably since this approach is designed only for pairs of highly similar images.

5 SIFTpack as a Bag-of-Words

Another advantage of our construction is that a SIFTpack built from SIFTs extracted from multiple different images, using Algorithm 2, can be viewed as an alternative for the highly popular dictionaries of visual words [12], which are typically constructed by K -means clustering. The dictionary construction process is performed offline; hence, its computation time is of lesser interest. Our main objectives are thus twofold. First, we wish to evaluate the benefits in storage space and representation error of SIFTpack in comparison with the standard K -means dictionary. Second, we wish to examine the contribution in runtime when using the SIFTpack to compute the histogram of word frequencies. The histogram is obtained by finding for each given SIFT the most similar SIFT word in the SIFTpack/dictionary.

5.1 Storage Space Saving

To assess the benefits of SIFTpack in terms of storage space we performed the following experiment. Given a set of SIFTs we use Algorithm 2 to construct the corresponding SIFTpack of size $m \times m \times 8$. We then compute the representation error of the SIFTpack as the average L_2 difference between each original SIFT and its nearest neighbor in the SIFTpack. In addition, we also construct a standard dictionary using K -means, setting K such that the same (as much as possible) representation error is obtained. Finally, we compare the array size of the SIFTpack and of the K -means dictionary. We have repeated this experiment for varying representation errors.

Figure 11 presents the obtained results for two setups. Figure 11a shows the results when the set of SIFTs was extracted at interest points of ~ 1700 images from the scene classification database [16] of 6 different scenes: “kitchen,” “bedroom,” “MITcoast,” “MIThighway,” “MITmountain,” and “MITstreet.” Figure 11b shows the results when representing dense-SIFT of a single image. The results are averaged over 80 repetitions of the experiment. We used $m = [10, 20, 30, \dots, 100]$. The units of space are the number of entries ($m \times m \times 8$ for SIFTpack and $K \times 128$ for K -means dictionary). As can be seen, the saving in storage space is tremendous for both setups. When packing dense-SIFTs the space reduction is more significant since the SIFTpack is more compact due to the gradual resizing during its construction.

5.2 Runtime Saving

Next, we assess the runtime benefits of SIFTpack when constructing the BoW representation for an image. The evaluation is done via the following experiment. As a testbed, we use the SIFTpack and K -means dictionaries constructed in the previous

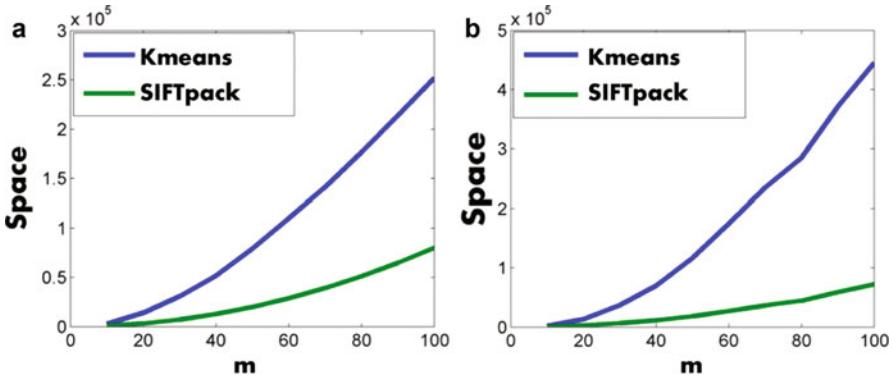


Fig. 11 Space saving by SIFTpack for BoW: the plots present the required storage space of SIFTpacks of varying representation errors, constructed with Algorithm 2, and of corresponding K -means dictionaries (with the same representation error). The storage space is calculated as the number of entries in the SIFTpack or K -means dictionary. **(a)** The results when representing a set of SIFTS from multiple different images. **(b)** The results when representing dense-SIFT of a single image. In all cases SIFTpack is significantly more space efficient than the standard K -means dictionary

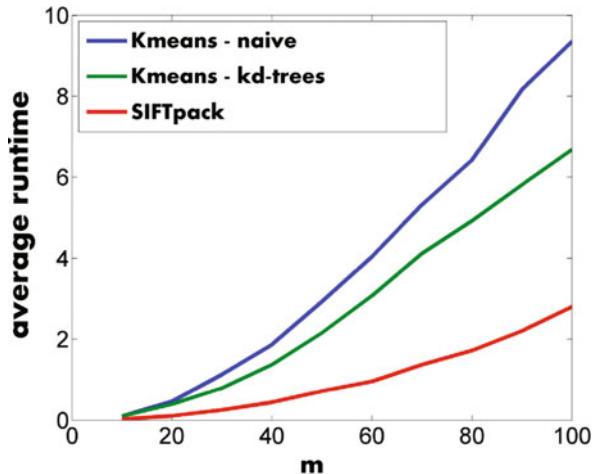
experiment from ~ 1700 images of six different scenes (corresponding pairs with the same representation error). Given an arbitrary input image we first extract dense-SIFT descriptors and construct the corresponding SIFTpack using Algorithm 1. Next, for each descriptor we find its nearest neighbor in the K -means dictionary and in the SIFTpack. We use the common naive approach as well as the kd-tree algorithm [3] for searching the K -means dictionary. We apply Algorithm 3 to find the exact nearest neighbor within the SIFTpack.

Figure 12 presents the obtained results, averaged over multiple experiments. As can be seen, using SIFTpack is significantly faster than using the popular K -means dictionary, even when kd-trees are used. We should note that as far as exact NN are concerned, the kd-tree algorithm does not outperform the naive one significantly.

5.3 Creating SIFTpacks with Larger Sparsity

As was explained in Sect. 3.2, Algorithm 2 is a simplified version of the general SIFTpack update steps through Orthogonal Matching Pursuit and Eq. (2). This simplification is achieved due to the constraint of sparsity = 1 with coefficient = 1. In general, a SIFTpack with larger sparsity can be created as an alternative to a dictionary. We performed several experiments that showed that such SIFTpacks are more compact in space than dictionaries with the same sparsity and reconstruction error. Nevertheless, the general formulation results in significantly longer SIFTpack construction time which puts a question mark on the worthwhileness of such a

Fig. 12 Runtime saving by SIFTpack for BoW: computing the bag-of-words model (constructing the histogram of frequencies) is significantly faster when using SIFTpack to represent the dictionary, instead of the standard k -means dictionary with the same representation error



structure. In addition, time saving can no longer be achieved with the integral image approach of Algorithm 3. Hence, we decided to focus in our experiments on the simplified case, demonstrating the time saving achieved by using the SIFTpack in this way.

6 SIFTpack as An Image Signature

In previous sections we showed the benefits of the SIFTpack for applications performing calculations between two dense-SIFTs (Sect. 4) and for applications performing calculations between a dense-SIFT and a dictionary (Sect. 5). Here we would like to test the third category of applications, those who perform calculations between two dictionaries. To achieve that, we will test the SIFTpack as an image signature for retrieval and measuring distances between different images.

It has recently been shown that robust retrieval can be achieved by constructing a dictionary for each image in the database and using dictionary-to-dictionary distance for finding similar images [20, 26, 45]. The distance between two dictionaries is taken as the sum of distances between nearest-neighbor pairs. This requires finding for each SIFT its nearest neighbor in the other dictionary. As was shown in Sect. 4 such matching can be performed significantly more efficiently using SIFTpack than the standard approaches.

To verify the usefulness of this retrieval approach we compare it against the standard K -means dictionaries on INRIA holidays dataset [21] that contains 1491 images, out of which 500 are queries. For each image we extract dense-SIFT and use Algorithm 2 to construct SIFTpacks of various sizes in a range that guarantees reasonable runtimes, $m \in \{20, 25, 30, 35, 40, 50, 60\}$. We further construct multiple K -means dictionaries, adjusting K such that each dictionary matches one of the

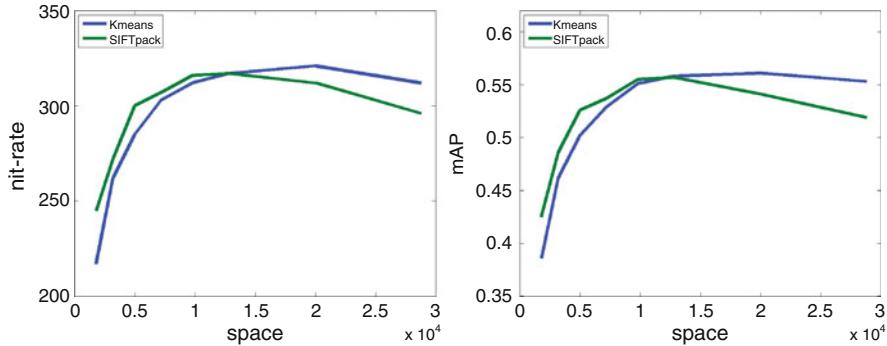


Fig. 13 Hit-rates and mean average precision results of the K -means dictionaries and the SIFTpacks with the same space in memory

Table 3 *Image retrieval results:* the table lists the hit rate and the mean average precision obtained by Spatial pyramid (SP) using L_2 and χ^2 metrics, together with the best results of K -means and SIFTpack dictionary comparison

	SP+ L_2	SP+ χ^2	K -means	SIFTpack
Hit rate	195	314	321	317
mAP	0.333	0.549	0.561	0.557

SIFTpacks, in terms of space, $K \in \{25, 39, 56, 77, 100, 156, 225\} = \frac{m \times m \times 8}{128}$. We retrieve for each query the most similar image in the dataset, using each of the SIFTpacks and K -means dictionaries. Similarity is measured according to the distance between dictionaries. In order to compare our method to the standard K -means, we compute two evaluation measures. The first one is hit rate, which is the amount of correctly retrieved images when taking only one most similar image to each query. The second is the more commonly used mean average precision (mAP) criteria. This evaluation measure comes with the INRIA holidays dataset and it calculates the mean of areas under recall precision graphs. The hit rate and the mAP are shown in Fig. 13. As can be seen, for small dictionaries the SIFTpack outperforms K -means, while for larger dictionaries K -means gives better results. Hence, when the space is limited, using SIFTpack dictionaries may be worthwhile. For larger K and m , the performance of both the K -means and the SIFTpack dictionaries deteriorates, due to over-fitting of each image by its dictionary.

Table 3 shows the retrieval results of the more popular spatial pyramid matching technique [29] with its default parameters (3 pyramid levels, 200 dictionary words). This technique represents an image by concatenating histograms of its descriptors, calculated over the entire image and on different regions of the image. The retrieval is performed by comparing these concatenated histograms using the L_2 and the χ^2 metrics. As can be seen, the retrieval results with L_2 metric are very low, while

using the χ^2 metric improves the results dramatically. We should note that the spatial pyramid matching technique compares histograms over a fixed dictionary, as opposed to dictionary comparison; thus, it works significantly faster. Hence, from a computational point of view, using the spatial pyramid matching may be worthwhile, even though at some working points, it is outperformed by the dictionaries comparison technique.

7 Conclusions

We have suggested an approach for compactly storing sets of SIFT descriptors. We have shown that the proposed SIFTpack saves not only in storage space, but more importantly, it enables huge reductions in runtime for matching between SIFTs. While our key idea is very simple, its implications could be highly significant as it can speed-up many algorithms. In particular, we have shown empirically the benefits of using SIFTpack instead of the traditional BoW dictionary. We have also shown the advantages of SIFTpack as an alternative image signature for retrieval purposes. We believe that storing SIFTs as an image could open the gate to using other algorithms on SIFTs that were restricted to only images before. In addition, our framework could be easily extended to other descriptors whose spatial properties are similar to SIFT.

Acknowledgements This research was supported in part by the Ollendorf Foundation and by the Israel Ministry of Science. We would also like to thank Prof. Michael Elad for useful conversations and good ideas.

References

1. Agarwal, A., Triggs, B.: Hyperfeatures - multilevel local coding for visual recognition. In: ECCV, pp. 30–43 (2006)
2. Aharon, M., Elad, M.: Sparse and redundant modeling of image content using an image-signature-dictionary. SIAM J. Imag. Sci. **1**(3), 228–247 (2008)
3. Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A.Y.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. J. ACM **45**(6), 891–923 (1998)
4. Bagon, S., Boiman, O., Irani, M.: What is a good image segment? A unified approach to segment extraction. In: ECCV, pp. 30–44 (2008)
5. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., Szeliski, R.: A database and evaluation methodology for optical flow. Int. J. Comput. Vis. **92**(1), 1–31 (2001)
6. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: a randomized correspondence algorithm for structural image editing. SIGGRAPH **28**(3), 24:1–24:11 (2009)
7. Barnes, C., Shechtman, E., Goldman, D.B., Finkelstein, A.: The generalized patchmatch correspondence algorithm. In: ECCV, pp. 29–43 (2010)
8. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Comput. Vis. Image Underst. **110**(3), 346–359 (2008)

9. Benoît, L., Mairal, J., Bach, F., Ponce, J.: Sparse image representation with epitomes. In: CVPR, pp. 2913–2920 (2011)
10. Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., Fua, P.: BRIEF: computing a local binary descriptor very fast. IEEE Trans. Pattern Anal. Mach. Intell. **34**(7), 1281–1298 (2012)
11. Crow, F.C.: Summed-area tables for texture mapping. SIGGRAPH **18**(3), 207–212 (1984)
12. Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: ECCV, vol. 1, pp. 1–22 (2004)
13. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: International Conference on Computer Vision & Pattern Recognition, vol. 2, pp. 886–893 (2005)
14. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: SoCG, pp. 253–262 (2004)
15. Deselaers, T., Ferrari, V.: Global and efficient self-similarity for object classification and detection. In: CVPR, pp. 1633–1640 (2010)
16. Fei-Fei, L., Perona, P.: A bayesian hierarchical model for learning natural scene categories. In: CVPR, pp. 524–531 (2005)
17. Furuya, T., Ohbuchi, R.: Dense sampling and fast encoding for 3d model retrieval using bag-of-visual features. In: CIVR, pp. 26:1–26:8 (2009)
18. Gilinsky, A., Zelnik-Manor, L.: Siftpack: a compact representation for efficient sift matching. In: IEEE International Conference on Computer Vision (ICCV). IEEE, New York (2013)
19. He, K., Sun, J.: Computing nearest-neighbor fields via propagation-assisted kd-trees. In: CVPR, pp. 111–118 (2012)
20. Janet, B., Reddy, A.: Image index model for retrieval using hausdorff distortion. In: ICCAIE, pp. 85–89 (2010)
21. Jegou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: ECCV, pp. 304–317 (2008)
22. Jégou, H., Douze, M., Schmid, C.: Product quantization for nearest neighbor search. IEEE Trans. Pattern Anal. Mach. Intell. **33**, 117–128 (2011)
23. Joulin, A., Bach, F., Ponce, J.: Discriminative clustering for image co-segmentation. In: CVPR, pp. 1943–1950 (2010)
24. Joulin, A., Bach, F., Ponce, J.: Multi-class cosegmentation. In: CVPR, pp. 542–549 (2012)
25. Ke, Y., Sukthankar, R.: Pca-sift: A more distinctive representation for local image descriptors. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 506–513 (2004)
26. Khapli, V.R., Bhattacharya, A.S.: Compressed domain image retrieval using thumbnails of images. In: CICSYN, pp. 392–396 (2009)
27. Khapli, V.R., Bhattacharya, A.S.: Image retrieval for compressed and uncompressed images. In: ICICS, pp. 1140–1143 (2009)
28. Korman, S., Avidan, S.: Coherency sensitive hashing. In: ICCV, pp. 1607–1614 (2011)
29. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: CVPR, pp. 2169–2178 (2006)
30. Li, Z., Imai, J., Kaneko, M.: Robust face recognition using block-based bag of words. In: ICPR, pp. 1285–1288 (2010)
31. Liu, C., Yuen, J., Torralba, A.: Sift flow: Dense correspondence across scenes and its applications. IEEE Trans. Pattern Anal. Mach. Intell. **33**(5), 978–994 (2011)
32. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60**(2), 91–110 (2004)
33. Mallat, S., Zhang, Z.: Matching pursuits with time-frequency dictionaries. IEEE Trans. Signal Process. **41**(12), 3397–3415 (1993)
34. Michael, N., Metaxas, D., Neidle, C.: Spatial and temporal pyramids for grammatical expression recognition of american sign language. In: ASSETS, pp. 75–82 (2009)
35. Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Trans. Pattern Anal. Mach. Intell. **27**(10), 1615–1630 (2005)

36. Nowak, E., Jurie, F., Triggs, B.: Sampling strategies for bag-of-features image classification. In: ECCV, pp. 490–503 (2006)
37. Olonetsky, I., Avidan, S.: Treecann k-d tree coherence approximate nearest neighbor algorithm. In: ECCV, pp. 602–615 (2012)
38. Ortiz, R.: Freak: Fast retina keypoint. In: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), CVPR ‘12, pp. 510–517 (2012)
39. Pati, Y.C., Rezaifar, R., Rezaifar, Y.C.P.R., Krishnaprasad, P.S.: Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In: Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers, pp. 40–44 (1993)
40. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR, pp. 1–8 (2007)
41. Philbin, J., Isard, M., Sivic, J., Zisserman, A.: Lost in quantization: Improving particular object retrieval in large scale image databases. In: CVPR, pp. 1–8 (2008)
42. Ramakrishnan, S., Rose, K., Gersho, A.: Constrained-storage vector quantization with a universal codebook. *IEEE Trans. Image Process.* **7**, 42–51 (1995)
43. Rubinstein, M., Gutierrez, D., Sorkine, O., Shamir, A.: A comparative study of image retargeting. *SIGGRAPH* **29**(5), 160:1–160:10 (2010)
44. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: Orb: An efficient alternative to sift or surf. In: Proceedings of the 2011 International Conference on Computer Vision, ICCV ‘11, pp. 2564–2571 (2011)
45. Schaefer, G.: Compressed domain image retrieval by comparing vector quantisation codebooks. In: Proc. SPIE, pp. 959–966 (2002)
46. Shechtman, E., Irani, M.: Matching local self-similarities across images and videos. In: CVPR, pp. 1–8 (2007)
47. Simakov, D., Caspi, Y., Shechtman, E., Irani, M.: Summarizing visual data using bidirectional similarity. In: CVPR, pp. 1–8 (2008)
48. Stecha, C., Bronstein, A.M., Bronstein, M., Fua, P.: LDAHash: improved matching with smaller descriptors. *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)* **34**, 66–78 (2012)
49. Strecha, C.: Dense matching of multiple wide-baseline views. In: ICCV, pp. 1194–1201 (2003)
50. Tola, E., Lepetit, V., Fua, P.: A fast local descriptor for dense matching. In: CVPR, pp. 1–8 (2008)
51. Trzcinski, T., Christoudias, M., Fua, P., Lepetit, V.: Boosting binary keypoint descriptors. In: Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR ‘13), pp. 2874–2881 (2013)
52. Umesh, K., Suresha, S.: Web image retrieval using visual dictionary. *Int. J. Web Serv. Comput.* **3**(3), 77–84 (2012)
53. Vedaldi, A., Fulkerson, B.: VLFeat: an open and portable library of computer vision algorithms. <http://www.vlfeat.org/> (2008)
54. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Action recognition by dense trajectories. In: CVPR, pp. 3169–3176 (2011)
55. Winder, S.A.J., Hua, G., Brown, M.: Picking the best daisy. In: CVPR, pp. 178–185. IEEE, New York (2009)
56. Yao, J., kuen Cham, W.: 3d modeling and rendering from multiple wide-baseline images by match propagation. *Sig. Proc.: Image Commu.* **21**(6), 506–518 (2006)

In Defense of Gradient-Based Alignment on Densely Sampled Sparse Features

Hilton Bristow and Simon Lucey

Abstract In this chapter, we explore the surprising result that gradient-based continuous optimization methods perform well for the alignment of image/object models when using densely sampled sparse features (HOG, dense SIFT, etc.). Gradient-based approaches for image/object alignment have many desirable properties—*inference* is typically fast and exact, and diverse constraints can be imposed on the motion of points. However, the presumption that gradients predicted on sparse features would be poor estimators of the true descent direction has meant that gradient-based optimization is often overlooked in favor of graph-based optimization. We show that this intuition is only partly true: sparse features are indeed poor predictors of the error surface, but this has no impact on the actual alignment performance. In fact, for general object categories that exhibit large geometric and appearance variation, sparse features are integral to achieving any convergence whatsoever. How the descent directions are predicted becomes an important consideration for these descriptors. We explore a number of strategies for estimating gradients, and show that estimating gradients via regression in a manner that explicitly handles outliers improves alignment performance substantially. To illustrate the general applicability of gradient-based methods to the alignment of challenging object categories, we perform unsupervised ensemble alignment on a series of nonrigid animal classes from ImageNet.

1 Notation

Before we begin, a brief aside to discuss notation in the sequel. Regular face symbols (i.e., n, N) indicate scalars, with lowercase variants reserved for indexing and uppercase for ranges/dimensions; lowercase boldface symbols (i.e., \mathbf{x}) indicate vectors; uppercase boldface symbols (i.e., \mathbf{J}) indicate matrices, and uppercase

H. Bristow
Queensland University of Technology, Brisbane, QLD, Australia
e-mail: hilton.bristow@gmail.com

S. Lucey (✉)
The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: sducey@cs.cmu.edu

calligraphic symbols (i.e., \mathcal{I}) indicate functions. We refer to images as functions rather than vectors or matrices to indicate that non-integer pixels can be addressed (by sub-pixel interpolation). This is necessary since the output coordinates of warp functions can be real valued. The notation $\mathcal{I} : \mathbb{R}^{D \times 2} \rightarrow \mathbb{R}^D$ indicates the sampling of D (sub-)pixels. To keep notation terse—and hopefully more readable—we often vectorize expressions. Therefore, in many instances, functions have vector-valued returns, though we endeavor to be explicit when this happens (as above).

2 Introduction

The problem of object or image alignment involves finding a set of parameters $\Delta\mathbf{x}$ that optimally align an input image \mathcal{I} to an object or image model,

$$\Delta\mathbf{x}^* = \arg \min_{\Delta\mathbf{x}} \mathcal{D}\{\mathcal{I}(\mathbf{x} + \Delta\mathbf{x})\} + \mathcal{A}\{\Delta\mathbf{x}\}. \quad (1)$$

Under this umbrella definition of alignment, we can instantiate particular models of optical flow, pose estimation, facial landmark fitting, deformable parts modelling, and unsupervised alignment commonly encountered in computer vision. $\mathcal{D} : \mathbb{R}^D \rightarrow \mathbb{R}$ is the continuous loss function which measures the degree of fit of the image observations to the model. $\mathcal{I} : \mathbb{R}^{D \times 2} \rightarrow \mathbb{R}^D$ is the image function which samples the (sub-)pixel values at the given locations, and we use the shorthand $\mathbf{x} = [\mathbf{x}_1^T, \dots, \mathbf{x}_D^T]^T$, $\mathcal{I}(\mathbf{x}) = [\mathcal{I}(\mathbf{x}_1 + \mathbf{x}_1), \dots, \mathcal{I}(\mathbf{x}_D + \mathbf{x}_D)]^T$ where $\mathbf{x}_i = [x_i, y_i]^T$ is the i th x - and y - discrete coordinates sampled on a regular grid at integer pixel locations within the continuous image function. $\mathcal{A} : \mathbb{R}^{2D} \rightarrow \mathbb{R}$ is the regularization function that will penalize the likelihoods of each possible deformation vector $\Delta\mathbf{x}$. For example, in optical flow, deformation vectors that are less smooth will attract a larger penalty. In the case of parametric warps (affine, similarity, homography, etc.) the regularization function acts as an indicator function which has zero cost if the deformation vector adheres to the desired parametric warp or infinite cost if it does not. In reality the alignment function \mathcal{D} can be as complicated as a support vector machine [7], mutual information [8], or deep network [24], or as simple as the sum of squared distances (SSD) between an input image and a fixed template.

Since pixel intensities are known to be poor estimators of object/part similarity, it is common in alignment strategies to instead use a feature mapping function,

$$\Delta\mathbf{x}^* = \arg \min_{\mathbf{x}} \mathcal{D}\{\Phi\{\mathcal{I}(\mathbf{x} + \Delta\mathbf{x})\}\} + \mathcal{A}\{\Delta\mathbf{x}\}, \quad (2)$$

where $\Phi : \mathbb{R}^D \rightarrow \mathbb{R}^{DK}$ is a nonlinear transformation from raw pixel intensities to densely sampled sparse features such as HOG [7] or SIFT [14]. K refers to the number of channels in the feature. In HOG and SIFT for example, these channels represent gradient orientation energies at each pixel location. In general one can attempt to solve either alignment objective [Eqs. (1) and (2)] in one of two ways:

(1) graph-, or (2) gradient-based search. The choice is largely problem specific, depending on the type of alignment and regularization function. We expand upon these considerations in the following subsections.

An especially important factor for gradient-based search strategies is the accuracy of the linearization matrix function of the representation (whether raw pixel intensities or densely sampled sparse features) with respect to the deformation vector. The linearization matrix function, or gradient as it is often referred to in computer vision, attempts to estimate an approximate linear relationship between the representation function and the deformation vector $\Delta\mathbf{x}$ over a restricted set of deformations. In this chapter we attempt to answer the question: *does the accuracy of this linearization reflect its utility in gradient search alignment?*

We argue that gradient search alignment strategies are often needlessly dismissed, as the linearization of $\Phi\{\mathcal{I}(\mathbf{x})\}$ of most natural images is poor in comparison with that obtained from $\mathcal{I}(\mathbf{x})$. We demonstrate empirically and with some theoretical characterization, that in spite of the poor linearization approximations of sparse features like SIFT and HOG, they actually enjoy superior gradient search alignment performance in comparison with raw pixel representations. We believe this result to be of significant interest to the computer vision community.

3 Search Strategies for Alignment

3.1 Graph-Based Search

If computation time was not an issue, one would simply exhaustively search all finite deformation vectors $\Delta\mathbf{x}$ in order to find the global minima. This brute force strategy is tractable for coarse-to-fine sliding window detectors such as Viola and Jones [22], but intractable for nearly all other deformations of interest within the field of computer vision. If the set of allowable displacements $\Delta\mathbf{x}$ is discretized and the function of parameters \mathcal{A} constrained to obey a particular graphical structure (e.g., tree, star, or chain), efficient graph optimization methods such as dynamic programming (i.e., belief propagation) can be applied to solve for the globally optimal deformation in polynomial time. A prominent example can be found in deformable parts models whose ensemble of local part detectors are restricted to search discrete translations and scales, but with additional constraints placed on the spatial layout of the parts. If the dependencies between the parts can be represented as a tree, one can search all possible deformations (in order to find a global minima) in a computationally tractable manner via dynamic programming [11].

Although graphical models have proven popular in the alignment literature, they still face a number of problems. Inference in graphical models is difficult and inexact in all but the simplest models such as tree- or star-structured graphs. For example, in the application of graph-based search to optical flow—termed SIFT Flow [13]—the regularization on the 2D smoothness of the flow prevents the allowable warps

from being factored into a tree structure. Instead, the authors employ an alternation strategy of enforcing the smoothness constraint in the x - and then y - directions (each of which independently can be represented as a tree structure) using dual-layer belief-propagation to find an approximate global solution. In many other cases, simplified tree- or star-structured models are unable to capture important dependencies between parts, so are not representative of the underlying structure or modes of deformation of the object being modelled [27]. The limited expressiveness of these simple models prevents many interesting constraints from being explored, which has led to the study of discrete but non-graphical models [17].

3.2 Gradient-Based Search

An alternative strategy for solving Eq.(1) in polynomial time is through nonlinear continuous optimization methods. This class of approaches linearize the system around the current estimate of the parameters, perform a constrained/projected gradient step, then update the estimate, iterating this procedure until convergence. We refer to this strategy in the sequel as *gradient-based*.

Gradient-based search methods can be categorized as deterministic or stochastic. Deterministic gradient estimation can be computed in closed form and is computationally efficient. This requires the alignment function to be continuous and deterministically differentiable. Stochastic gradient estimation involves the sampling of a function with respect to its parametric input in order to estimate a first or second order relationship between the function and the input and can be computationally expensive (especially when one is trying to establish second order relationships). Both methods, when applied to object or image alignment, employ stochastic gradient estimation methods at some level. Deterministic methods estimate stochastic gradients on the representation, and then leverage closed form first and second order derivative information of the alignment function. Stochastic methods, however, estimate stochastic gradients directly from the objective [25].

Deterministic gradient-based search methods have a long history in alignment literature [5, 10, 12, 26]. The most notable application of this concept is the classic Lucas & Kanade (LK) algorithm [15], which has been used primarily for image registration. The approach estimates gradients stochastically on the image representation, and then employs deterministic gradients of the objective of the SSD alignment function, resulting in an efficient quasi-Newton alignment algorithm. Many variations upon this idea now exist in computer vision literature [1, 4, 5] for applying deterministic gradient search to object registration.

A good example of stochastic gradient-based search for object/image alignment can be found in the constrained mean-shift algorithm for deformable part alignment (made popular for the task of facial landmark alignment [19]). In this approach, stochastic gradients around the alignment objective are estimated independently for each part detector, from which a descent direction is then found that adheres to the

learned dependencies between those parts. The focus in this chapter, however, will be solely on deterministic gradient-based methods due to their clear computational advantages over stochastic methods.

4 Linearizing Pixels and Sparse Features

As stated earlier, our central focus in this chapter is to first investigate how well sparse features like HOG and SIFT linearize compared to pixel intensities. To do this we first need to review how one estimates the representation's gradient estimate $\nabla \mathcal{R}(\mathbf{x}) : \mathbb{R}^{2D} \rightarrow \mathbb{R}^{D \times 2D}$ when performing the linearization,

$$\mathcal{R}(\mathbf{x} + \Delta \mathbf{x}) \approx \mathcal{R}(\mathbf{x}) + \nabla \mathcal{R}(\mathbf{x}) \Delta \mathbf{x}, \quad (3)$$

where $\mathcal{R}(\mathbf{x}) \in \{\mathcal{I}(\mathbf{x}), \Phi\{\mathcal{I}(\mathbf{x})\}\}$ is a representation function that is agnostic to the choice of representation: raw pixel intensities $\mathcal{I}(\mathbf{x})$, or densely sampled sparse features $\Phi\{\mathcal{I}(\mathbf{x})\}$.

4.1 Gradient Estimation as Regression

One can view the problem of gradient estimation naively as solving the following regression problem,

$$\nabla \mathcal{R}(\mathbf{x}) = \arg \min_{\mathbf{J}} \sum_{\Delta \mathbf{x} \in \mathbb{P}} \eta \{ \mathcal{R}(\mathbf{x} + \Delta \mathbf{x}) - \mathbf{J} \Delta \mathbf{x} \}, \quad (4)$$

where \mathbb{P} is the set of deformations over which we want to establish an approximately linear relationship between the representation $\mathcal{R}(\mathbf{x} + \Delta \mathbf{x})$ and the deformation vector $\Delta \mathbf{x}$. η is the objective function used for performing the regression, for example $\eta\{\cdot\} = \|\cdot\|_2^2$ would result in least-squares regression. This gradient estimation step can be made more efficient by considering each coordinate in $\mathbf{x} = [\mathbf{x}_1^T, \dots, \mathbf{x}_D^T]^T$ to be independent of each other. This results in a set of KD regression problems,

$$\nabla \mathcal{R}_i^k(\mathbf{x}_i) = \arg \min_{\mathbf{J}} \sum_{\delta \in \mathbb{L}} \{ \mathcal{R}_i^k(\mathbf{x}_i + \delta) - \mathbf{J} \delta \}, \quad \forall i = 1 : D, \quad k = 1 : K \quad (5)$$

where $\nabla \mathcal{R}_i^k(\mathbf{x}_i) : \mathbb{R}^2 \rightarrow \mathbb{R}^1$, \mathbb{L} is the local translation deformation set for each pixel coordinate (normally a small window of say 3×3 or 5×5 discrete pixel coordinates), D is the number of pixel coordinates, and K is the number of channels in the representation (e.g., for raw pixel intensities $K = 1$). We can then define $\nabla \mathcal{R}_i(\mathbf{x}_i) : \mathbb{R}^{2D} \rightarrow \mathbb{R}^{DK \times 2D}$ as

$$\nabla \mathcal{R}(\mathbf{x}) = \begin{bmatrix} \nabla \mathcal{R}_1^1(\mathbf{x}_1) \\ \vdots \\ \nabla \mathcal{R}_1^K(\mathbf{x}_1) \\ \ddots \\ \nabla \mathcal{R}_D^1(\mathbf{x}_D) \\ \vdots \\ \nabla \mathcal{R}_D^K(\mathbf{x}_D) \end{bmatrix}. \quad (6)$$

Of course, linear regression is not the only option for learning the gradient regressor. One could also consider using support vector regression (SVR) [9], which has better robustness to outliers. Intuitively, SVR predicts the gradient direction from a different weighted combination of pixels within a local region around the reference pixel. SVR has a clear performance advantage, with a commensurate increase in computation during training.

4.2 Gradients Estimation as Filtering

For a least-squares objective $\eta\{\cdot\} = \|\cdot\|_2^2$ the solution to each gradient matrix function can be computed in closed form,

$$\nabla \mathcal{R}_i^k(\mathbf{x}_i) = \left(\sum_{\delta \in \mathbb{L}} \boldsymbol{\delta} \boldsymbol{\delta}^T \right)^{-1} \left(\sum_{\delta \in \mathbb{L}} \boldsymbol{\delta} [\mathcal{R}_i^k(\mathbf{x}_i) - \mathcal{R}_i^k(\mathbf{x}_i + \boldsymbol{\delta})] \right). \quad (7)$$

There are a number of interesting things to observe about this formulation. The first term in the solution is independent of the representation—it depends only on the local deformations sampled, and so can be inverted once rather than for each \mathcal{R}_i^k . The second term is simply a sum of weighted differences between a displaced pixel and the reference pixel, i.e.,

$$\begin{bmatrix} \sum_{\Delta x} \sum_{\Delta y} \Delta x (\mathcal{R}_i^k(x_i + \Delta x, y_i + \Delta y) - \mathcal{R}_i^k(x, y)) \\ \sum_{\Delta x} \sum_{\Delta y} \Delta y (\mathcal{R}_i^k(x_i + \Delta x, y_i + \Delta y) - \mathcal{R}_i^k(x, y)) \end{bmatrix}. \quad (8)$$

If $\boldsymbol{\delta} = [\Delta x, \Delta y]^T$ is sampled on a regular grid at integer pixel locations, Eq. (8) can be cast as two filters—one each for horizontal weights Δx , and vertical weights Δy ,

$$f_x = \begin{bmatrix} x_{-n} & \dots & x_n \\ \vdots & & \\ x_{-n} & \dots & x_n \end{bmatrix} \quad f_y = \begin{bmatrix} y_{-n} & \dots & y_n \\ \vdots & & \\ y_n & \dots & y_n \end{bmatrix} \quad (9)$$

Thus, an efficient realization of Eq. (7) of the gradient at every pixel coordinate is

$$\nabla \mathcal{R}_i^k(\mathbf{x}_i) = \left(\sum_{\delta \in \mathbb{L}} \delta \delta^T \right)^{-1} \text{diag} \left(\begin{bmatrix} f_x * \mathcal{R}_i^k(\mathbf{x}) \\ f_y * \mathcal{R}_i^k(\mathbf{x}) \end{bmatrix} \right), \quad (10)$$

where $*$ is the 2D convolution operator. This is equivalent to blurring the image with a clipped quadratic and then taking the derivative. It is also possible to place weights on δ stemming from \mathbb{L} as a function of its distance from the origin. In the case of Gaussian weights this results in the classical approach to estimating image gradients by blurring the representation with a Gaussian and taking central differences. It is surprising that the two formulations make opposing assumptions on the importance of pixels, and as we show in our experiments section the clipped quadratic kernel induced by linear regression is better for alignment.

4.3 Pixels Versus Sparse Features

Considerable literature has been devoted to finding image features for general object classes that are discriminative of image semantics whilst being tolerant to local image contrast and geometric variation. The majority of existing feature transforms encode three components: (1) nonlinear combinations of pixels in local support regions, (2) multichannel outputs, and (3) sparsity. Prominent image features that exhibit these properties include HOG [7] and densely sampled SIFT descriptors [14]. We refer to this class of transforms as *densely sampled sparse features*.

Natural images are known to stem from a $\frac{1}{f}$ frequency spectrum [20]. This means that most of the energy in the image is concentrated in the lower frequencies—the image function is naturally smooth. Sparse multichannel features follow no such statistics. In fact, they often exhibit highly nonlinear properties: small changes in the input can sometimes produce large changes in the output (e.g., gradient orientations close to a quantization boundary in HOG/SIFT can cause the output feature to jump channels, pixel differences close to zero in binary features can cause the output feature to swap signs), and other times produce no change in the output (e.g., orientations in the center of a bin, pixel differences far from zero).

To evaluate the generative capacity of different representations (i.e., how well the tangent approximation predicts the true image function at increasing displacements) we performed a simple experiment. We evaluated the signal-to-noise (SNR) ratio of the linearization function $\nabla \mathcal{R}(\mathbf{x})$ for increasing displacements across a number of images,

$$\text{SNR}(\mathbf{x}) = 10 \log_{10} \left(\frac{\|\mathcal{R}(\mathbf{x} + \Delta \mathbf{x})\|}{\|\mathcal{R}(\mathbf{x}) + \nabla \mathcal{R}(\mathbf{x}) \Delta \mathbf{x} - \mathcal{R}(\mathbf{x} + \Delta \mathbf{x})\|} \right)^2. \quad (11)$$

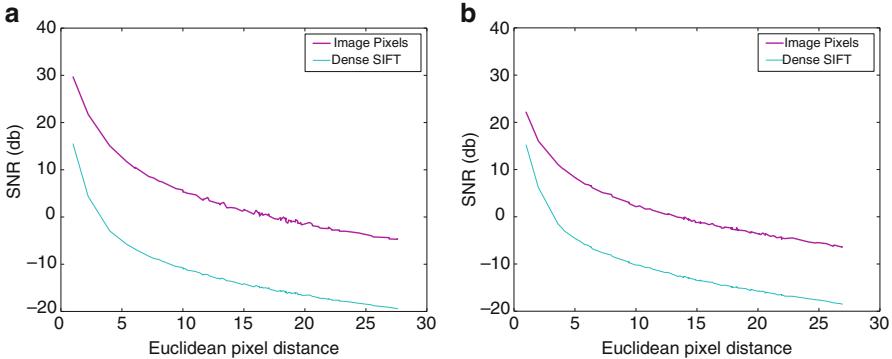


Fig. 1 An experiment to illustrate the generative ability of pixel and densely sampled sparse features (in this case dense SIFT). We compute the linearization error $\mathcal{R}(\mathbf{x}) + \nabla \mathcal{R}(\mathbf{x}) \Delta \mathbf{x} - \mathcal{R}(\mathbf{x} + \Delta \mathbf{x})$ for a range of $\Delta \mathbf{x}$ (x-axis), and look at the resulting signal-to-noise ratio (SNR) on the y-axis. The results are averaged over 10,000 random trials across 100 images drawn from a set of (a) faces and (b) animals. As expected, the generative accuracy of pixels is consistently higher than densely sampled sparse features, and better for face imagery than animal+background imagery (though the sparse representation is largely unchanged)

For simplicity, we restricted the deformation vectors $\Delta \mathbf{x}$ to global translation. Figure 1 illustrates the SNR versus Euclidean distance (i.e., $\|\Delta \mathbf{x}\|_2$) for images of (a) faces and (b) animals.

The tangent to the pixel image is a consistently better predictor of image appearance than the same applied to sparse features (in this case, dense SIFT). This confirms the intuition that pixel images are smoothly varying, whereas nonlinear multichannel features are not. The experiment of Fig. 1 indirectly suggests sparse features would not be appropriate for gradient-based alignment search strategies. Unsurprisingly, graph-based optimization has become the strategy of choice for alignment when using sparse features, with some notable exceptions [16, 23, 25]. As a result, the wealth of research into continuous alignment methods [5, 12, 15, 18, 28] has largely been overlooked by the broader vision community.

5 Experiments

So far we have talked in generalities about gradient-based alignment methods and the properties they enjoy. In this section, we instantiate a particular model of gradient-based alignment based on the Lucas & Kanade (LK) algorithm [15] in order to illustrate these properties in a number of synthesized and real-world settings.

We perform two tasks: (1) pairwise image alignment from a template image to a query image, and (2) ensemble alignment where the alignment error of a group of images stemming from the same object class is minimized. In both tasks, existing

gradient-based alignment approaches have typically only used pixel intensities, and as a result have only been evaluated on constrained domains such as faces, handwritten digits, and building façades. Understandably, this has failed to capture the attention of the broader vision community working on challenging object classes with high intra-class variation.

We seek to show that gradient-based methods *can* be applied to object categories for which densely sampled sparse features are requisite to attaining meaningful similarities between images, and that a vanilla implementation of LK can go a long way to achieving interesting results on a number of challenging tasks.

5.1 The Lucas & Kanade Algorithm

Recollect our formulation of the alignment problem in Eq.(1), this time using the representation function \mathcal{R} that is agnostic to the choice of image function,

$$\Delta\mathbf{x}^* = \arg \min_{\Delta\mathbf{x}} \mathcal{D}\{\mathcal{R}(\mathbf{x} + \Delta\mathbf{x})\} + \mathcal{A}\{\Delta\mathbf{x}\}. \quad (12)$$

A common substitution within the LK algorithm is

$$\mathbf{p}^* = \arg \min_{\mathbf{p}} \|\mathcal{R}(\mathbf{p}) - \mathcal{T}(\mathbf{0})\|_2^2, \quad (13)$$

where \mathbf{p} is a set of warp parameters that model the deformation vector $\Delta\mathbf{x}$ by proxy of a warp function,

$$\mathcal{R}(\mathbf{p}) = \begin{bmatrix} \mathcal{R}\{\mathcal{W}(\mathbf{x}_1; \mathbf{p})\} \\ \vdots \\ \mathcal{R}\{\mathcal{W}(\mathbf{x}_D; \mathbf{p})\} \end{bmatrix} \quad (14)$$

and $\mathcal{W}(\mathbf{x}; \mathbf{p}) : \mathbb{R}^{2D} \rightarrow \mathbb{R}^P$. The warp function conditions the deformation vector on the warp parameters such that $\mathbf{x} + \Delta\mathbf{x} = \mathcal{W}(\mathbf{x}; \mathbf{p})$. In most instances the dimensionality of $\mathbf{p} \in \mathbb{R}^P$ is substantially less than the canonical deformation vector $\Delta\mathbf{x} \in \mathbb{R}^{2D}$ (e.g., for a 2D affine warp $P = 6$). This is equivalent to setting \mathcal{A} to be an indicator function, which has zero cost when the parameters fall within the feasible set of warps, and infinity otherwise. As in Eq.(3), the LK algorithm takes successive first-order Taylor expansions about the current estimate of the warp parameters, and solves for the local update,

$$\Delta\mathbf{p}^* = \arg \min_{\Delta\mathbf{p}} \|\mathcal{R}(\mathbf{p}) + \nabla \mathcal{R}(\mathbf{p}) \frac{\partial \mathcal{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - \mathcal{T}(\mathbf{0})\|_2^2, \quad (15)$$

where $\nabla\mathcal{R}(\mathbf{p})$ is the gradient estimator, and $\frac{\partial\mathcal{W}}{\partial\mathbf{p}}$ is the Jacobian of the warp function which can be found deterministically or learned offline. Here we have presented the LK algorithm using the canonical L_2 loss function and the linearization function estimated from the input image representation \mathcal{R} as opposed to the template \mathcal{T} . In reality there are a slew of possible variations on this classical LK form. Baker and Matthews [2, 3] provide a thorough reference for choosing an appropriate update strategy and loss function. We present LK in this manner to avoid introducing unnecessary and distracting detail for the unfamiliar reader.¹ Regardless of these details, the choice of image representation and method of gradient calculation remain integral to the performance observed.

5.2 Pairwise Image Alignment

Earlier in Fig. 1 we performed a synthetic experiment showing the linearization error as a function of displacement for different image representations. Here we perform the sequel to that experiment, showing the frequency of convergence of the LK algorithm as a function of initial misalignment.

We initialize a bounding box template within an image, then perturb its location by a given RMS point error (measured from the vertices of the bounding box) and measure the frequency with which the perturbed patch converges back to the initialization after running LK. The results are shown in Fig. 2. We perform two variants of the experiment, (a) *intra-image* alignment, where the template and perturbation are sampled from the same image, and (b) *inter-image* alignment, where the perturbation is sampled from a different image of the same object class, with known ground-truth alignment. The task of inter-image alignment is markedly more difficult, since the objects within the template and the perturbation may have different nonrigid geometry, scene lighting, and background clutter.

Even in the intra-image scenario, dense SIFT consistently converges more frequently than pixel intensities. In the inter-image scenario, the difference is even more pronounced. Figure 3 shows a more comprehensive view of the inter-image scenario, with a comparison of the different gradient estimation techniques we have discussed. In general, there is a gradual degradation in performance from SVR to least squares regression to central differences. The *domain* in the legend specifies the blur kernel size in the case of central differences, or the support region over which training examples are gathered for regression. Figure 4 illustrates the type of

¹In our experiments we actually estimate our linearization function from the template image $\mathcal{T}(\mathbf{0}) \rightarrow \nabla\mathcal{T}(\mathbf{0})$ using a technique commonly known within LK literature as the *inverse compositional* approach. This was done due to the substantial computational benefit enjoyed by the inverse compositional approach, since one can estimate $\mathcal{T}(\mathbf{0}) \rightarrow \nabla\mathcal{T}(\mathbf{0})$ once, as opposed to the classical approach of estimating $\mathcal{R}(\mathbf{p}) \rightarrow \nabla\mathcal{R}(\mathbf{p})$ at each iteration. See [2, 3] for more details.

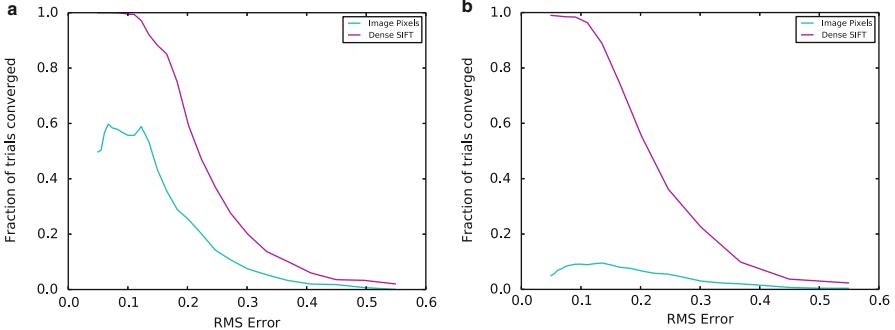


Fig. 2 An experiment to illustrate the alignment performance of pixel intensities versus densely sampled sparse features (in this case densely extracted SIFT descriptors). In both scenarios, we initialize a bounding box within an image, then perturb its location by a given RMS point error (x -axis) and measure the frequency with which the perturbed patch converges back to the initialization (y -axis). In (a) we perform *intra*-image alignment, where the template and perturbation are sampled from the same image. In (b) we perform *inter*-image alignment, where the perturbation is sampled from a different image of the same object class with known ground-truth alignment. The task of inter-image alignment is markedly more difficult, since the two objects being aligned may be experiencing different lightning and pose conditions. The drop in pixel performance is more pronounced than dense SIFT when moving to the harder task

imagery on which we evaluated the different methods—animal classes drawn from the ImageNet dataset, often exhibiting large variations in pose, rotation, scale, and translation.

5.3 Ensemble Alignment

We finish the piece with the challenging real-world application of ensemble alignment. The task of ensemble alignment is to discover the appearance of an object of interest in a corpus of images in an unsupervised manner. Discrete approaches are typically unsuitable to this problem because searching over translation and scale alone is insufficient for good alignment, and exploring higher-dimensional warps using discrete methods is either infeasible or computationally challenging.

We present results using a gradient-based approach called least squares congealing [6]. The details of the algorithm are not essential to our discussion, however it features the same linearization as the LK algorithm, and as such is subject to the same properties we have discussed throughout this chapter.

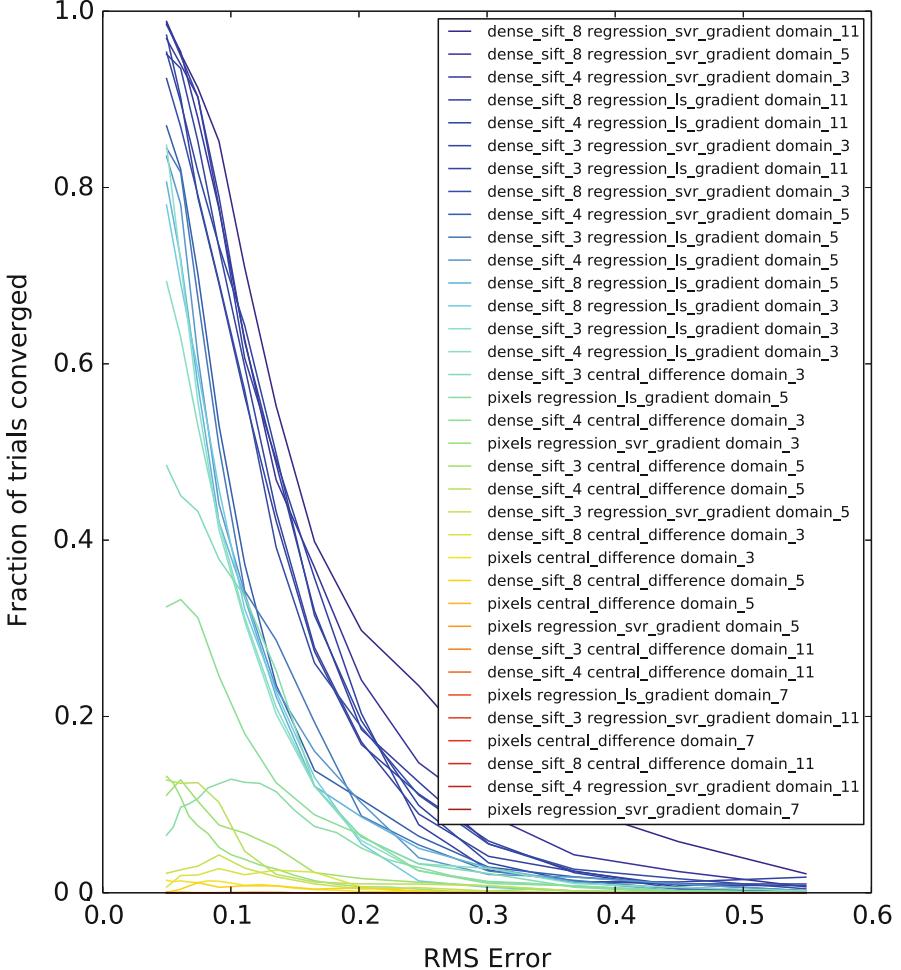


Fig. 3 Inter-image alignment performance. We initialize a bounding box within the image, then perturb its location by a given RMS point error (x axis), run Lucas Kanade on the resulting patch, and measure the frequency with which the patch converges back to the initialization (y axis). The *domain* specifies the Gaussian standard deviation in the case of central differences, or the maximum displacement from which training examples are gathered for regression. On dense SIFT, there is a progressive degradation in performance from SVR to least-squares regression to central differences. Pixel intensities (using any gradient representation) perform significantly worse than the top dense SIFT-based approaches

Figure 5 shows the results of aligning a subset of 170 elephants drawn from the ImageNet dataset,² using dense SIFT features and least squares regression,

²We removed those elephants whose out-of-plane rotation from the mean image could not be reasonably captured by an affine warp. The requirement of a single basis is a known limitation of the congealing algorithm.

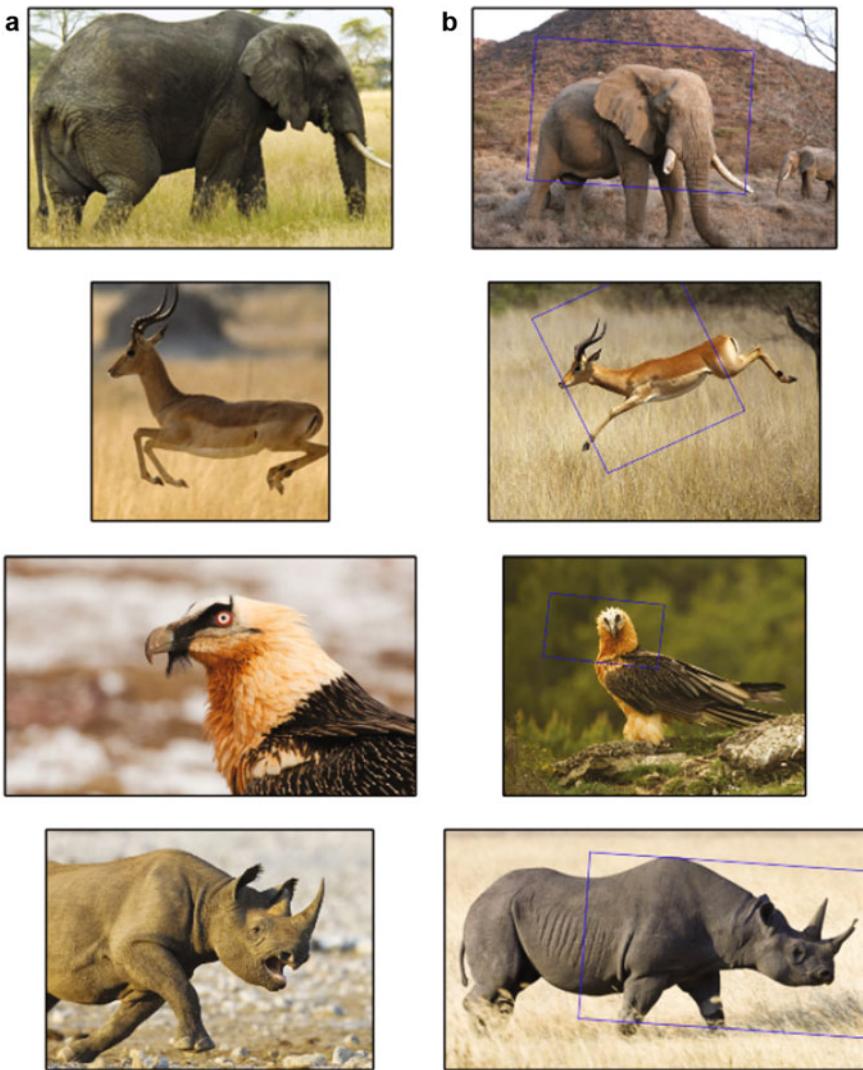


Fig. 4 Representative pairwise alignments. (a) is the template region of interest, and (b) is the predicted region that best aligns the image to the template. The exemplars shown here all used dense SIFT features and least squares regression to learn the descent directions. The four examples exhibit robustness to changes in pose, rotation, scale, and translation, respectively

parametrized on a similarity warp. The same setup using pixel intensities failed to produce any meaningful alignment. Figure 6 shows the mean of the image stack before and after congealing. Even though individual elephants appear in different poses, the aligned mean clearly elicits an elephant silhouette.

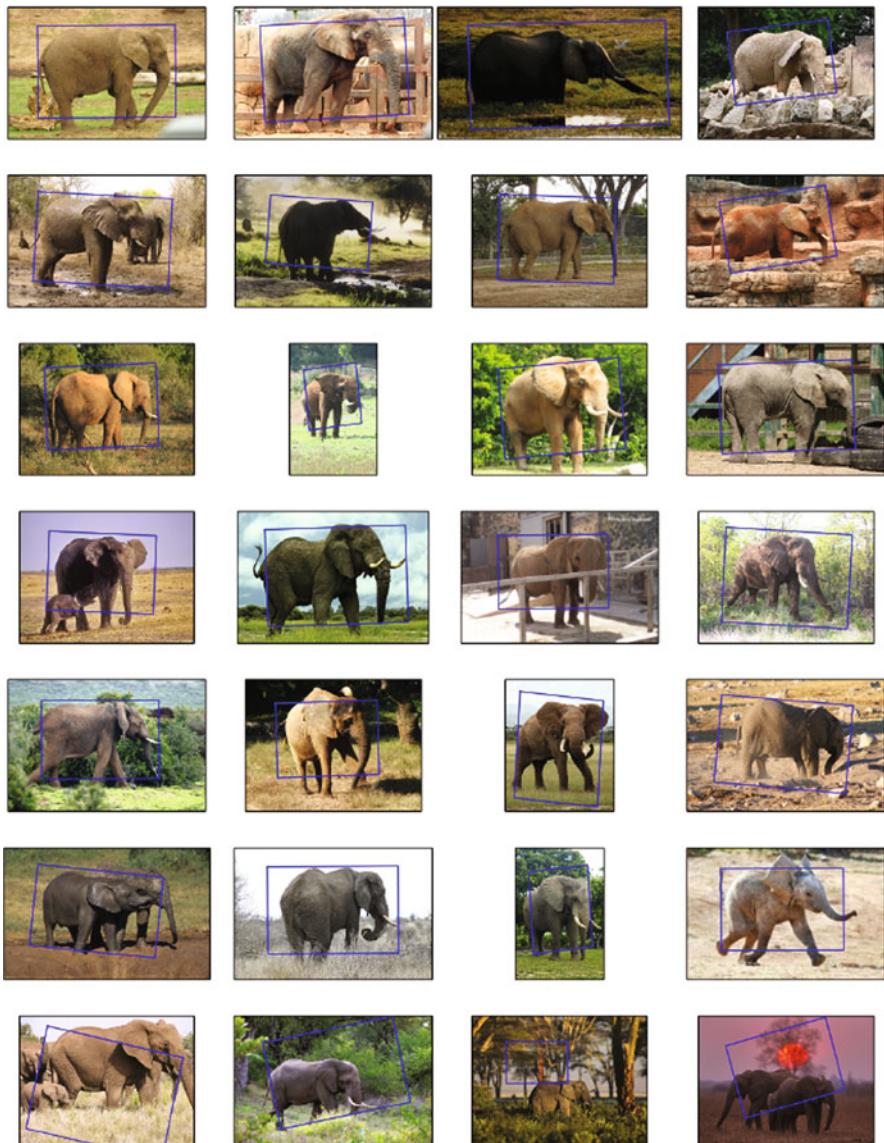


Fig. 5 Unsupervised ensemble alignment (congealing) on a set of 170 elephants taken from ImageNet. The objective is to jointly minimize the appearance difference between all of the images in a least-squares sense—no prior appearance or geometric information is used. The *first six rows* present exemplar images from the set that converged. The *final row* presents a number of failure cases

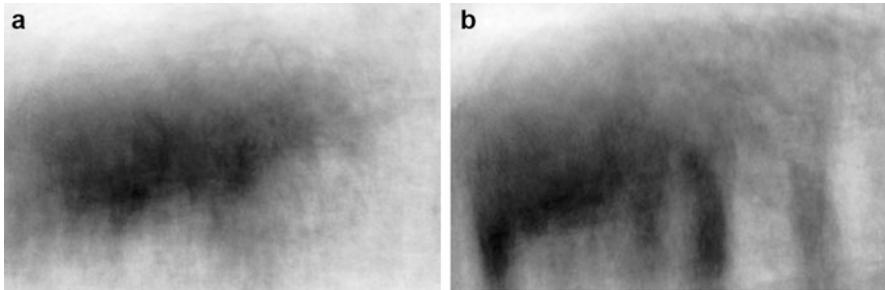


Fig. 6 The mean image of Fig. 5 (a) before alignment, and (b) after alignment with respect to a similarity warp. Although individual elephants undergo different nonrigid deformations, one can make out an elephant silhouette in the aligned mean

6 Discussion

So far in this chapter we have presented the somewhat paradoxical result that densely sampled sparse features perform well in real-world alignment applications (Figs. 2 and 3) while sporting poor tangent approximations (Fig. 1). Here we try to offer some insight into why this might be the case.

Consider first the effect of convolving a sparse signal with a low-pass filter. We know from compressive-sensing that observed blurred signals can be recovered almost exactly if the underlying signal is sparse [21]. Unlike traditional dense pixel representations whose high-frequency information is attenuated when convolved with a low-pass filter, sparse signals can be blurred to a much larger extent without any information loss before reaching the limits of sampling theory. Figure 7 illustrates the effect of comparing dense and sparse signals as the degree of misalignment and blur increases.

The immediate implication of this for image alignment is that a sparse multi-channel representation can be blurred to dilate the convergent region whilst preserving information content. The encoding of local pixel interactions ensures this information content contains high-frequency detail required for good alignment.

7 Conclusion

Image alignment is a fundamental problem for many computer vision tasks. In general, there are two approaches to solving for the optimal displacement parameters: (1) to iteratively linearize the image function and take gradient steps over the parameters directly, or (2) to exploit redundancies in the set of allowable deformations and enumerate the set using graphical models. While a large body of research has focussed on gradient-based alignment strategies in the facial domain, they have rarely been applied to broader object categories. For general objects,

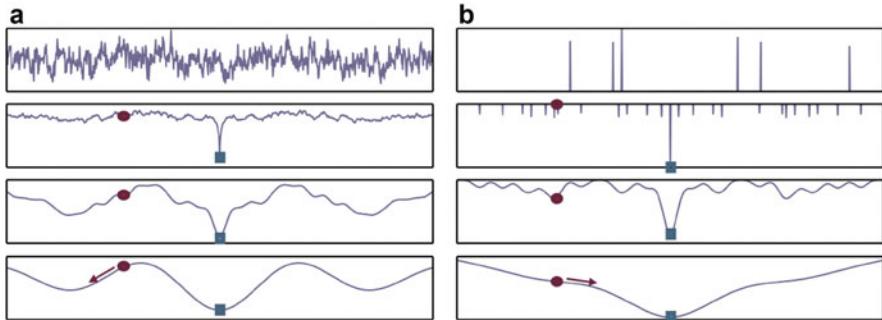


Fig. 7 A 1D alignment thought experiment. The *first row* shows two signals: a dense signal with a $\frac{1}{f}$ frequency spectrum, and a sparse positive signal. The *second, third, and fourth rows* show the negative auto-correlation of the signals to simulate the expected loss for varying degrees of misalignment (x -axis) with increasing amounts of Gaussian blur applied to the original signals (*row-wise*). The *red circles* represent a hypothetical initialization of the parameters (in this case x -translation), the *green squares* represent the global optima, and the *arrows* indicate the direction of steepest descent. For the given initialization, gradient-based alignment on the dense signal will never converge to the true optima. Even with a large amount of blur applied, the solution is divergent (the gradient of the cross-correlation is moving away from the optima). The sparse signal, on the other hand, can tolerate a larger amount of blur and still maintain the location of the optima, in this case converging with the greatest amount of blur applied. This illustrates the importance of sparse, positive representations when matching misaligned signals. In order to retain discriminative appearance information, modern features use *multi-channel*, sparse, positive representations—but the basic concept remains

alignment in pixel space performs poorly because low frequency information in the signal is dominated by lighting variation. Densely sampled sparse features provide tolerance to local image contrast variation, at the expense of reducing the range over which tangent approximations to the image function are accurate. As a result, graphical models have become the preferred approach to alignment when using densely sampled sparse features.

We motivated this chapter with the surprising result that although the tangent approximation is poor, the real-world results when using image features are impressive. We offered some insights into why this may be the case, along with a number of approaches for estimating the descent directions. We ended the piece with an unsupervised ensemble alignment experiment to illustrate how gradient-based methods can operate on challenging imagery with high-dimensional warp functions.

In summary, we showed that the application of gradient-based methods to general object alignment problems is possible when using densely sampled sparse features, and their capacity to handle complex warp/regularization schemes may facilitate some interesting new approaches to existing challenges in image and object alignment.

References

1. Avidan, S.: Support vector tracking. *Pattern Anal. Mach. Intell. (PAMI)* **26**(8), 1064–72 (2004)
2. Baker, S., Matthews, I.: Lucas-Kanade 20 years on: a unifying framework: part 1. *Int. J. Comput. Vis.* **56**(3), 221–255 (2004)
3. Baker, S., Matthews, I.: Lucas-Kanade 20 years on: a unifying framework: part 2. *Int. J. Comput. Vis. (IJCV)* **56**(3), 221–255 (2004)
4. Black, M.J., Jepson, A.: Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *Int. J. Comput. Vis. (IJCV)* **26**(1), 63–84 (1998)
5. Cootes, T.F., Edwards, G.J., Taylor, C.J.: Active appearance models. *IEEE Trans. Pattern Anal. Mach. Intell. (PAMI)* **6**, 681–685 (2001)
6. Cox, M., Sridharan, S., Lucey, S.: Least-squares congealing for large numbers of images. In: *International Conference on Computer Vision (ICCV)* (2009)
7. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 886–893 (2005)
8. Dowson, N., Bowden, R.: Mutual information for Lucas-Kanade Tracking (MILK): an inverse compositional formulation. *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(1), 180–5 (2008)
9. Drucker, H., Burges, C., Kaufman, L., Smola, A., Vapnik, V.: Support vector regression machines. In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 155–161 (1997)
10. Felzenszwalb, P.F.: Representation and detection of deformable shapes. *Pattern Anal. Mach. Intell. (PAMI)* **27**(2), 208–20 (2005)
11. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. *Pattern Anal. Mach. Intell. (PAMI)* **32**(9), 1627–45 (2010)
12. Horn, B.K., Schunck, B.G.: Determining optical flow. *Artif. Intell.* **17**(1–3), 185–203 (1981)
13. Liu, C., Yuen, J., Torralba, A.: SIFT flow: dense correspondence across scenes and its applications. *Pattern Anal. Mach. Intell. (PAMI)* **33**(5), 978–94 (2011)
14. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis. (IJCV)* **60**(2), 91–110 (2004)
15. Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: *International Joint Conference on Artificial Intelligence (IJCAI)* (1981)
16. Rakêt, L., Roholm, L., Nielsen, M., Lauze, F.: TV-L 1 optical flow for vector valued images. In: *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 1–14 (2011)
17. Ramakrishna, V., Munoz, D., Hebert, M., Bagnell, J.A., Sheikh, Y.: Pose machines: articulated pose estimation via inference machines. In: *European Conference on Computer Vision (ECCV)*, pp. 1–15 (2014)
18. Saragih, J., Lucey, S., Cohn, J.: Face alignment through subspace constrained mean-shifts. In: *International Conference on Computer Vision (ICCV)* (2009)
19. Saragih, J.M., Lucey, S., Cohn, J.F.: Deformable model fitting by regularized landmark mean-shift. *Int. J. Comput. Vis. (IJCV)* **91**(2), 200–215 (2011)
20. Simoncelli, E.P., Olshausen, B.A.: Natural image statistics and neural representation. *Annu. Rev. Neurosci.* **24**(1), 1193–1216 (2001)
21. Tsagkatakis, G., Tsakalides, P., Woiselle, A.: Compressed sensing reconstruction of convolved sparse signals. In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2014)
22. Viola, P., Jones, M.: Robust real-time object detection. *Int. J. Comput. Vis.* **4**, 51–52 (2001)
23. Weber, J., Malik, J.: Robust computation of optical flow in a multi-scale differential framework. *Int. J. Comput. Vis. (IJCV)* **81**, 67–81 (1995)
24. Weinzaepfel, P., Revaud, J., Harchaoui, Z., Schmid, C.: DeepFlow: large displacement optical flow with deep matching. In: *International Conference on Computer Vision (ICCV)*, pp. 1385–1392 (2013)

25. Xiong, X., De la Torre, F.: Supervised descent method and its applications to face alignment. In: International Conference of Computer Vision and Pattern Recognition (CVPR), pp. 532–539 (2013)
26. Yuille, A.: Deformable templates for face recognition. *J. Cogn. Neurosci.* **3**(1), 59–70 (1991)
27. Zhu, X., Ramanan, D.: Face detection, pose estimation, and landmark localization in the wild. In: International Conference of Computer Vision and Pattern Recognition (CVPR), pp. 2879–2886 (2012)
28. Zimmermann, K., Matas, J., Svoboda, T.: Tracking by an optimal sequence of linear predictors. *Pattern Anal. Mach. Intell. (PAMI)* **31**(4), 677–92 (2009)

Part II

Dense Correspondences and Their

Applications

From Images to Depths and Back

Tal Hassner and Ronen Basri

Abstract This chapter describes what is possibly the earliest use of dense correspondence estimation for transferring semantic information between images of different scenes. The method described in this chapter was designed for non-parametric, “example-based” depth estimation of objects appearing in single photos. It consults a database of example 3D geometries and associated appearances, searching for those which look similar to the object in the photo. This is performed at the pixel level, in similar spirit to the more recent methods described in the following chapters. Those newer methods, however, use robust, generic dense correspondence estimation engines. By contrast, the method described here uses a hard-EM optimization to optimize a well-defined target function over the similarity of appearance/depth pairs in the database to appearance/estimated-depth pairs of a query photo. Results are presented demonstrating how depths associated with diverse reference objects may be assigned to different objects appearing in query photos. Going beyond visible shape, we show that the method can be employed for the surprising task of estimating shapes of occluded objects’ backsides. This, so long as the reference database contains examples of mappings from appearances to backside shapes. Finally, we show how the duality of appearance and shape may be exploited in order to “paint colors” on query shapes (“colorize” them) by simply reversing the matching from appearances to depths.

1 Introduction

The human visual system is remarkably adept at estimating the shapes of objects from their appearance in just a single view. This, despite this being an ill-posed

T. Hassner (✉)

Department of Mathematics and Computer Science, The Open University of Israel, Raanana,
Israel

e-mail: hassner@openu.ac.il

R. Basri

The Weizmann Institute, Rehovot, Israel

e-mail: ronen.basri@weizmann.ac.il

problem: different shapes may appear the same in a photo, any one of them as plausible as the next. In order to overcome this challenge, existing computational methods often make a-priori assumptions on various aspects and properties of the objects and scenes, the lighting and viewing conditions, and the camera setup.

The method described here makes the following, alternative assumption: We assume that the object viewed is roughly similar in appearance and shape (but by no means identical) to those of known, related objects. The obvious example here is of course faces, a fact which has recently been exploited by methods including [17, 24, 31]. As we will later show, examples of typical face shapes can be used to estimate even highly unusual faces. We claim that the same is true for other object classes and indeed demonstrate results for images of challenging objects, including highly deformable hands and full body figures, and heavily textured fish.

Our method assumes that a database of relevant example 3D geometries and their associated appearances is available. Given textured 3D models, their appearances and associated depths can easily be obtained using standard, computer graphics, rendering techniques [23]. To estimate the shape of a novel object viewed in a single photo, we search through the appearances of these objects, looking for those similar in appearance to the object in the query photo. The known depths of the selected objects serve as priors for the shape of the query object. This matching is performed per pixel, at the patch level, by forming dense correspondences between query and reference images. This process therefore produces depth estimates very different from the ones of the reference objects in the database. The entire process is cast as a Hard-EM procedure, optimizing a well-defined target likelihood function which reflects the estimated depth likelihood, given the input photo and reference database.

This approach of transferring depths by dense correspondences has several advantages, recognized also by others (see, e.g., [28] and subsequent chapters in this book): (1) It is non-parametric, and so requires no a-priori model selection or design. (2) It is therefore *versatile* and as we will show can be used to estimate the shapes of objects belonging to very different object classes or solve other appearance/depth related tasks. Finally, (3) data driven methods require making no assumptions on the properties of the object, scenes, viewing conditions or lighting conditions in the input photos. Their main requirement is the availability of a suitable set of reference 3D examples. Today, such examples can easily be obtained from standard depth sensing devices (e.g., the Kinect), or from any of the many databases of hand-crafted 3D computer graphics models.

The rest of this chapter is organized as follows. In the next section we review related work. Our depth estimation framework is described in Sect. 3. Other applications of our approach, including backside shape estimation and “depth painting,” are described in Sect. 4. Implementation and results are presented in Sect. 5. Finally, we conclude in Sect. 6.

This chapter is based on work which has previously appeared in [18–20].

2 Related Work

This section surveys related methods published before and after the one described in this chapter. Subsequent chapters of this book discuss related applications of dense correspondences and share some of the same background. Additional information on related methods is therefore available there as well. Here we focus on previous methods for the specific tasks of single view depth reconstruction and shape decoration.

Depth Estimation The literature on computerized estimation of shape from single views is immense. Indeed, this problem is often considered to be one of the classical challenges of computer vision. Monocular reconstruction methods often rely on different cues such as shading, silhouette shapes, texture, and vanishing points (e.g., [3, 8, 10, 12, 27, 38, 44, 53]). These methods all restrict possible reconstructions by placing constraints on the properties of the objects in the image (e.g., reflectance properties, viewing conditions, and symmetry).

Quite a few methods focus specifically on single view, scene reconstruction problems, specifically outdoor or indoor (e.g., office) scenes. The successful early method of Hoiem et al. [25, 26] reconstructs outdoor scenes assuming a “ground,” “sky,” and “vertical” billboard partitioning of the scene. Other related methods include the diorama construction [1] and the Make3D system of Saxena et al. [43]. More recently, [15] reconstructs and labels the components of indoor office scenes. Finally, two recent methods use deep-learning to predict scene depth for indoor scenes [14, 36].

A growing number of methods propose using examples to guide the reconstruction process. One notable approach makes the assumption that all 3D objects in an object class lie close to a linear subspace spanned using a few basis objects (e.g., [2, 4, 13, 42]). This approach was mostly applied to faces, and less so to other object classes where its underlying assumption is less appropriate. Another approach [30] uses a single reference 3D surface to produce accurate, shape-from-shading estimates for faces. This approach is also designed for the specific problem of face shape estimation. More recently, others [6, 7] attempt to model allowable deformations of known object classes and then using these models to estimate object shape and pose from single views.

A fully data driven method, employing dense pixel correspondences, was originally proposed in [19], inspired by methods for constructing 3D models by combining parts [21]. This chapter elaborates on this method. Other methods have since proposed different ways of directly using shape/appearance examples of known references, including the shape from recognition approach of Thomas et al. [46], the Shape Collages of Cole et al. [9] and the depth transfer method of Karsch et al. [28] (described in more detail later in this book). Finally, several have recently proposed employing shape-from-shading assumptions in a local, example-based approach (e.g., [39, 54]).

Shape Decoration The relation of appearance and depth is exploited in this chapter not only to estimate depths from appearances, but also to solve the reverse problem of estimating *appearances from depths*. Though this problem is less studied than the first, it has been considered in the past, particularly in computer graphics related forums. Early automatic methods for decorating, or “colorizing,” 3D models have mostly focused on covering surfaces of 3D models with texture examples (e.g., [48, 52, 55]). An optimization similar to the one used here was described by Han et al. [16] for the purpose of covering 3D surfaces with 2D texture examples. Others provide the human operator with semi-automatic control over the application of non-textural image-maps for 3D models (e.g., [56]).

Some have proposed forming correspondences between textured 3D models, thereby allowing for the texture of one model to be transferred to the other (e.g., [32, 40]). These methods often require human operators to seed a set of correspondences between the two models, or else assume that the models have very similar 3D structure.

Relevant to the work described here is the recent mesh-colorization method of Leifman and Tal [33]. Though their goal is similar to our own, unlike us, their method relies on color propagation on the 3D surface structure, rather than dense transfer of appearances from one 3D shape to another.

3 Estimating Depth from Correspondences

Given a query photo I of an object belonging to a certain class, we wish to estimate the object’s depth-map D . To this end, we use a collection of reference, *feasible mappings* from appearances (intensities) to depths of objects of the same class. This example set is stored in a database $S = \{M_i\}_{i=1}^n = \{(I_i, D_i)\}_{i=1}^n$, where I_i and D_i , respectively, are a photo and depth-map of one of the database objects. In practice, we produce such image/depth pairs by using standard computer graphics rendering methods, applying them to textured, 3D geometries.

We wish to estimate a depth D such that every $k \times k$ patch of mappings in $M = (I, D)$ is feasible. Feasibility is defined here with respect to the database S : we wish to produce mappings which have similar enough counterparts S . That is, our goal is to produce a depth D which satisfies the following criteria:

1. For every $k \times k$ patch of mappings in M , a similar patch of mappings exists in the database S and
2. If two patches of mappings overlap at a pixel p , then the two selected matching database patches agree on the depth at p .

The optimization scheme described next is designed to produce depth estimates satisfying these criteria.

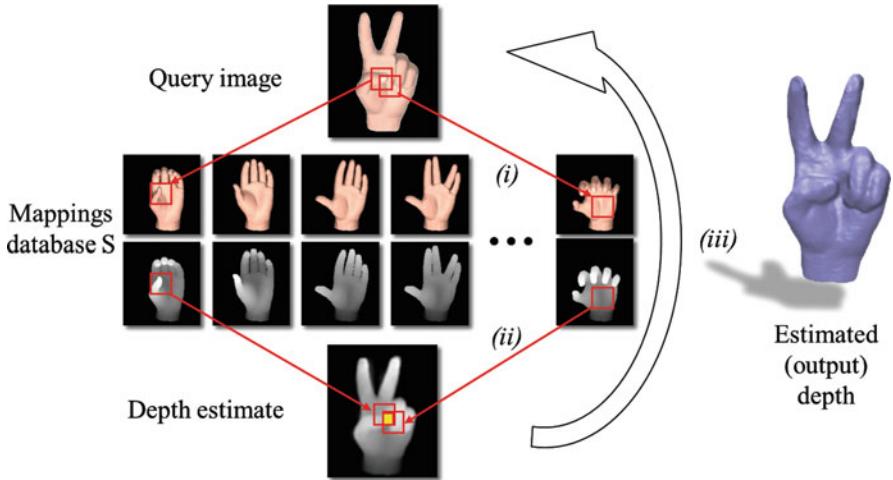


Fig. 1 *Left:* optimization illustration. For every query patch Step (i) finds a similar database patch. Database patches provide depth estimates for their matched pixels. Overlapping patches provide multiple estimates for each pixel. These are combined at each pixel to produce a new depth estimate for that pixel in Step (ii). This process is repeated until convergence (iii) by returning to Step (i), now matching patches using intensities *and* depths. *Right:* output depth estimate, rendered in 3D

3.1 Optimization Scheme

Given a query photo I , the following simple, two-step procedure is used to produce a depth estimate D meeting our two criteria (see also Fig. 1): (*Step 1*) At every pixel p in I we consider a $k \times k$ window around p . We search the database for intensity patches similar to the query patch in the least squares sense (Fig. 1(i)). (*Step 2*) The $k \times k$ depth values associated with each selected database patch are extracted and assigned as depth estimates for the $k \times k$ neighborhood around p . Since this is performed at each pixel location, image pixels will be assigned k^2 depth estimates: one estimate for each $k \times k$ patch overlapping the pixel. A single depth estimate for p is then obtained by taking the weighted mean of its k^2 depth estimates (Fig. 1(ii)). Here, Gaussian weights are used to bias depth value estimates from patches centered closer to p .

The result of this process is only taken as an initial depth estimate. This holds because the process above does not guarantee that the resulting depth-map would meet the criteria defined above. We therefore refine our estimate using an iterative procedure, repeating the following process until convergence (see also Fig. 2): Similarly to the initial step, we again seek $k \times k$ database patches similar to those in the query. Now, however, patches are matched using not only intensities, but also depths; that is, matching *mappings* from appearances to depths, rather than just appearances. Once database mappings are selected, a new depth estimate is computed for each pixel as before, by taking the Gaussian weighted mean of its k^2

Fig. 2 Pseudo-code summarizing the basic steps of our method

```

D = estimateDepth(I, S)
M = (I, ?)
repeat until no change in M
  (i)   V = getSimilarPatches(M, S)
  (ii)  D = updateDepths(M, V)
        M = (I, D)

```

estimates. Later, in Sect. 3.2 we prove that this two-step procedure is a hard-EM optimization of a well-defined target function reflecting our criteria, and is therefore guaranteed to converge at a local optimum of the target function.

Figure 2 provides pseudo-code for this process. The function *getSimilarPatches* searches the database S for patches of mappings matching those in M . We denote by \mathcal{V} the set of all such matching patches for all pixels p . Function *updateDepths* updates every pixel p with a new depth estimate computed by taking the weighted mean of the depth values assigned to p in \mathcal{V} .

3.2 Depth Plausibility as a Likelihood Function

The iterative process described above can be shown to be a hard-EM optimization [29] of the following target function, satisfying our criteria of Sect. 3. Denote by W_p a $k \times k$ patch from the query M , centered at p , containing both intensity values and (unknown) depths, and denote by V a similar reference patch in some $M_i \in S$. We can now define the target function as

$$\text{Plaus}(D|I, S) = \prod_{p \in I} \max_{V \in S} \text{Sim}(W_p, V), \quad (1)$$

with the similarity measure $\text{Sim}(W_p, V)$ being:

$$\text{Sim}(W_p, V) = \exp\left(-\frac{1}{2}(W_p - V)^T \Sigma^{-1}(W_p - V)\right), \quad (2)$$

Here, Σ is a constant diagonal matrix, its components representing the individual variances of the intensity and depth components of patches in the class. These can be provided by the user as weights, and in fact, are the only user-controlled parameter employed by our method (see also Sect. 5). Robustness to illumination is obtained by intensity normalization, setting intensities to have zero mean and unit variance in each patch. We note that more recent methods have introduced robustness to illumination and other challenging viewing conditions, by employing feature descriptors designed for these purposes. These include the SIFT [37] used by the SIFT-flow method [34, 35], or more elaborate representations (e.g., [22]).

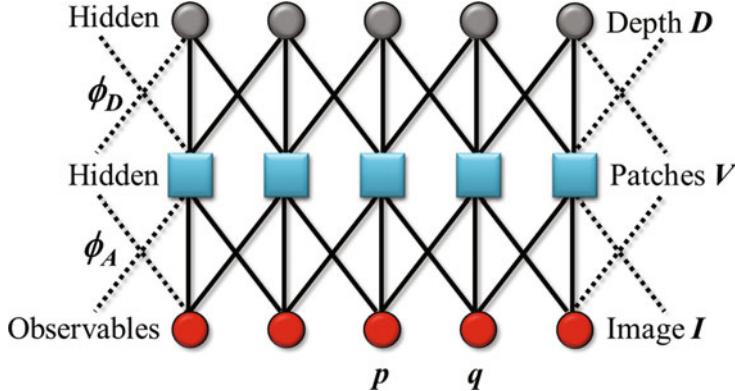


Fig. 3 Graphical model representation. Please see text for more details

Some of these methods have been surveyed in the first part of this book. These can be used as surrogates to the patches used here, as was recently demonstrated in [17].

Figure 3 presents query image intensities as observable values of a graphical model. Hidden values represent the depth values D which we seek to estimate. The joint probability of hidden and observed variables can be formulated through the edge potentials by

$$f(I, \mathcal{V}; D) = \prod_{p \in I} \prod_{q \in W_p} \phi_I(V_p(q), I(q)) \cdot \phi_D(V_p(q), D(q))$$

where V_p is the database patch matched by the global assignment \mathcal{V} to an image patch W_p centered at p . Taking ϕ_I and ϕ_D to be Gaussians with different covariances over the appearance and depth, respectively, implies

$$f(I, \mathcal{V}; D) = \prod_{p \in I} \text{Sim}(W_p, V_p)$$

where Sim is defined in Eq. (2). Integrating over all possible assignments of \mathcal{V} we obtain the following likelihood function

$$L = f(I; D) = \sum_{\mathcal{V}} f(I, \mathcal{V}; D) = \sum_{\mathcal{V}} \prod_{p \in I} \text{Sim}(W_p, V_p).$$

This sum is approximated by a maximum operator, a common practice for EM algorithms often referred to as hard-EM (e.g., [29]). Since similarities at different pixels may be computed independently, we interchange the product and maximum operators, resulting in the following maximum likelihood:

$$\max L \approx \prod_{p \in I} \max_{V \in S} \text{Sim}(W_p, V) = \text{Plaus}(D|I, S),$$

which is our cost function (1).

The function *estimateDepth* of Fig. 2 maximizes this measure by implementing a hard-EM optimization: The function *getSimilarPatches* performs a hard E-step by selecting the set of assignments \mathcal{V}^{t+1} for time $t + 1$ which maximizes the posterior:

$$f(\mathcal{V}^{t+1}|I; D^t) \propto \prod_{p \in I} \text{Sim}(W_p, V_p).$$

Here, D^t is the depth estimated at time t . Because patch similarities are independent, this can be maximized by matching each patch in M with its most similar database patch, in the least squares sense.

The function *updateDepths* approximates the M-step by finding the most likely depth assignment at each pixel:

$$D^{t+1}(p) = \arg \max_{D(p)} \left(- \sum_{q \in W_p} (D(p) - \text{depth}(V_q^{t+1}(p)))^2 \right).$$

This expression is maximized by the mean depth value over all k^2 estimates $\text{depth}(V_q^{t+1}(p))$, for all neighboring pixels q . Finally, we note that Hard-EM optimization is well known to converge to a local optimum of the target function [29].

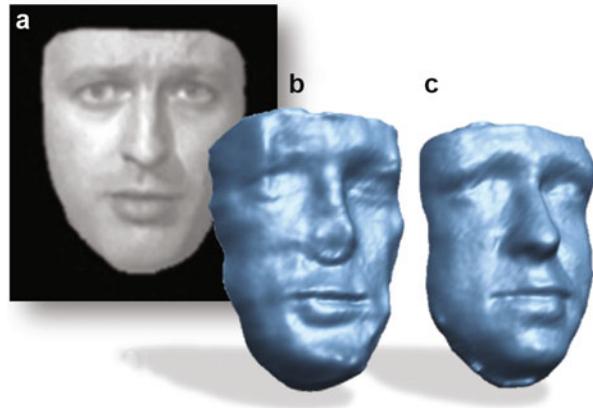
3.3 Preserving Global Structure

Unlike methods such as SIFT-Flow, which seek smooth flow fields with small displacements [34, 35], the optimization above is not designed to preserve spatial continuity or smoothness of correspondences. It therefore implicitly assumes the output is stationary [51]: That is, that the probability for the depth at any pixel, given those of its neighbors, is the same, regardless of the location of that pixel. This is generally untrue for structured objects, where shapes often depend on position. To illustrate, the probability of a pixel's depth changes if that pixel displays the tip of the nose of the corner of the eyes.

To address this issue, we enforce *non-stationarity* by adding constraints to the patch matching process. We encourage selecting database patches from similar semantic parts. This is by favoring patches which match not only in their appearance/depth mapping values, but in their positions relative to the centroid of the (estimated) query depth-map. In practice, we add the relative positions as additional values in each patch of mappings, both for database and query image patches.

Let $p = (x, y)$ be the coordinates of a pixel in I (the center of the patch W_p) and (x_c, y_c) be the coordinates of the center of mass of the area occupied by non-background depths in the depth estimate D at time t . The values $(\delta x, \delta y) = (x - x_c, y - y_c)$, are appended to each patch W_p and similarly to all database patches (i.e., by using the center of depth image D_i for (x_c, y_c)). By doing so, the matching

Fig. 4 Preserving relative position. (a) Query photo. (b) Depths estimated without preserving global structure. (c) With structure preservation



process now must match appearance, depth, and relative location values. We provide an example of a depth reconstruction with and without these constraints in Fig. 4.

In practice, if the query object is segmented from its background, an initial estimate for the query's centroid can be obtained from the foreground pixels. Alternatively, this constraint can be applied only after an initial depth estimate has been computed (i.e., Sect. 3).

3.4 Coarse to Fine Optimization

We have found that attempting to generate depth estimates for the query photo directly often results in low frequency noise, appearing as “lumpy” depth surfaces. To avoid this, as well as speed convergence of the process, we perform our optimization in a coarse-to-fine manner. Specifically, we represent the appearance component of each pixel by both its intensity and high frequency values, as encoded in the Gaussian and Laplacian pyramids of I [5]. Hence, in practice, our method generates the layers of a depth Laplacian pyramid and the final depth is obtained by collapsing these depth high-frequencies estimates from all scales. In doing so, low frequency depth values are generated in the coarse scale of the pyramid and only sharpened at finer scales, leading to smoother depth surfaces containing high frequency discontinuities (see example in Fig. 5).

As we discuss in Sect. 3.2, the different components of our mappings are assumed to originate from Gaussian distributions with different covariances [i.e., Σ in the definition of Sim , Eq. (2)]. In practice, we treat these covariances as weights associated with each component and reflecting their relative contribution when measuring the similarity of query mappings to database mappings.

To illustrate this, consider the relative position component (Sect. 3.3). In some classes, where object shapes are more structured (less deformable), we expect that matching should encourage the use of mappings from similar relative positions,

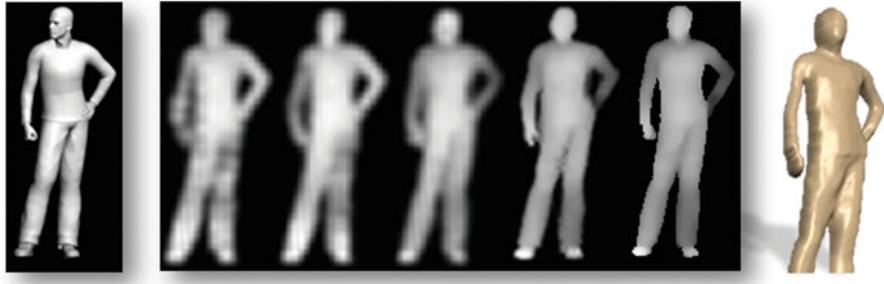


Fig. 5 Coarse to fine processing. *From left to right*, query photo, five intermediate depth-map estimates from different scales and a zoomed-in view of our output estimate

whereas classes which include deformable objects should be less strict with these values. We therefore amplify different components of each patch W_p for different classes, by weighting them differently. In Sect. 5 we provide the values of these weights, automatically computed, for the object classes used in our experiments.

4 Beyond Appearance to Depth Mappings

The method described so far produces depth estimates by matching query appearance, or appearance/depth pairs to a set of appearance/depth examples. We observe that the choice of information being matched and generated is arbitrary: rather than matching query appearances and generating depths, one may just as well match query depths, generating appearances. Another alternative is generating a second, occluded (backside) depth layer, given examples of mappings from appearances to backside depths. We next describe these two applications, using the method described above, with the only difference being the nature of the information given in the query and the mappings of the example database.

4.1 Automatic Depth-Map Colorization

We are motivated by computer graphics applications where 3D shapes are typically textured with colors in order to appear realistic. Different ways of applying colors (“colorizing”) 3D models were described in Sect. 2. When a shape is available as a depth-map, the same process described above can be readily applied as an alternative means of fitting it with a realistic appearance; this, by simply reversing the mappings defined in Sect. 3, producing a database of depth to appearance mappings, rather than the reverse used so far.

Specifically, we define our database as $S = \{M'_i\}_{i=1}^n = \{(D_i, I_i)\}_{i=1}^n$. For a query depth-map D , we now generate an image-map I such that $M' = (D, I)$ consists of plausible mappings from *shape to intensities*. Section 5.1 provides examples of depths from several object classes, and the appearances automatically generated for them using this process.

4.2 Backside Reconstruction

We note that behind the method described in this chapter is the following implicit assumption: that the relationship between the appearances and shapes of known object classes is predictable; given enough example mappings from appearance to shape, one can estimate a plausible shape given a single image of an object from the class. We claim that the same assumption often holds when considering the relationship between the depth-maps of objects in a known class and their *occluded backside* depths.

Rather than a database of mappings of appearances and (visible) depth-maps, here we produce a database of mappings from depth-maps to their backside depths. We refer to the surface of the object, furthest from the camera plane. A “backside depth-map” can then be defined similarly to the standard depth-map definition: it is the distance along the ray originating in the camera center, through each pixel, to its point of intersection with the surface at the object’s back.

A database of mappings $S' = \{M'_i\}_{i=1}^n = \{(D_i, D'_i)\}_{i=1}^n$ is produced using the standard computer graphics rendering utilities. Here, D'_i is a second depth layer, specifically, the backside depth-map. Once a plausible depth-map D is estimated from the query photo (Sect. 3), we use it to estimate a plausible backside depth-map D' by replacing our mappings from appearance to depth, to mappings $M'(p) = (D(p), D'(p))$ from depth-map D to backside depth-map D' . We provide qualitative examples of backside depth-map generation results in Sect. 5.1.

5 Implementation and Experiments

Our experiments include datasets representing articulated hands (52 appearance/depth-map pairs), whole bodies (45 pairs), faces (77 pairs), and fish (41 pairs). Hands and body examples were all produced by exporting the built-in models available by the Poser software. Faces were produced using the USF head database [49]. Finally, fish models were obtained from [47]. Our colorization experiments additionally used five, textured whole body scans courtesy of Cyberware [11].

The triangulated, textured models in these datasets were processed using standard rendering software (e.g., [23]) in order to generate appearances (rendered images) and corresponding depth-maps. Some example appearance/depth mappings of fish



Fig. 6 Example database mappings for depth estimation and colorization. *Left:* Two appearance/depth mappings from the fish object class. *Right:* Two mappings from the whole body class

Table 1 Relative weights of mapping components

DB name	Weights		
	Appearance	Depth	Position
Human-posture	0.2140	0.1116	0.0092
Hands	0.1294	0.1120	0.0103
Fish	0.1361	0.1063	0.0116

Weights used for intensities, depth, and relative position components for different object classes (databases)

and whole bodies are provided in Fig. 6. We note that, as with shape-from-shading based methods, we assume query objects were pre-segmented from their background and then aligned with a single preselected database image to solve for scale and image-plane rotation differences (see, e.g., [30]).

System Parameters The relative weights of the three components of our mappings, namely, appearance, depth, and relative positions, were estimated empirically for depth reconstruction. Five randomly selected objects from each object class were selected to automatically search for optimal weights. A direct simplex search method was used to search for these three parameters, separately, for each object class, minimizing the L2 error between our depth estimate and the known ground truth. We provide the parameter values Table 1. Parameter search was performed once for each class; the selected parameters were used with all query images. We note that these parameter values reflect the covariances over the appearance and depths and relative locations mentioned in Sect. 3.2.

Finally, for 200×150 pixel images, we used three image scales (Sect. 3.4). Patch sizes were 5×5 at the coarsest scale, 7×7 at the second level, and 9×9 for the finest scale.

5.1 Results

This chapter provides qualitative examples of structured objects such as faces (Figs. 4 and 7) and non-rigid object classes such as hands (Figs. 1 and 8) and whole bodies (Figs. 5 and 8). Higher than zero genus objects are provided in, e.g., Fig. 8.



Fig. 7 Two face depth estimation results. For both results, *left to right* are the query photo, its estimated depth-map and the depth-map textured using the input photo. Depth-map results are shown rendered in 3D

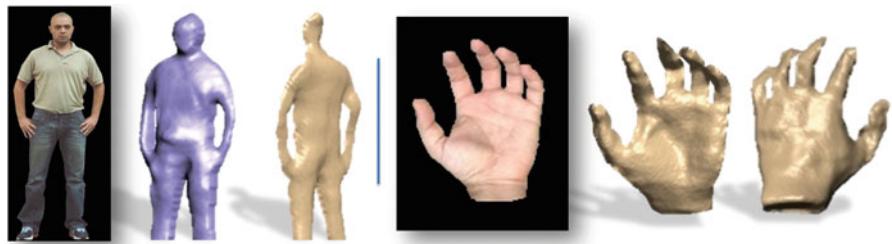


Fig. 8 Whole body (*left*) and hand (*right*) depth estimation results. For both results, *left to right* are the query photo, its estimated depth-map and its estimated backside depth-map. Depth-map results are shown rendered in 3D

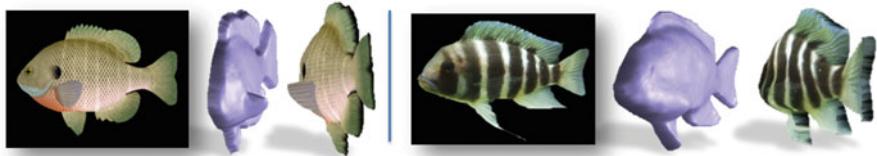


Fig. 9 Two fish object depth estimation results. For both results, *left to right* are the query photo, its estimated depth-map and the depth-map textured using the input photo. Depth-map results are shown rendered in 3D

Handling of depth discontinuities is demonstrated in the hand results of Fig. 1 and the fish fin of Fig. 9. Results with highly textured query photos are provided for the fish class in Fig. 9.

The flexibility of a per pixel, example based approach is demonstrated in Fig. 10. It compares our reconstruction result to the shape-from-shading method of Kemelmacher and Basri [30] on a non-standard (Cyclops) face. Although [30] generate finer details than our method, their strong, *global* face shape prior results in an estimate erroneously containing three separate eyes.

Occluded backside depth-map result (Sect. 4.2) are provided in Fig. 8. Two, non-structured object classes with non-trivial back shapes were selected for these examples. Finally, some generated colorizations are presented in Figs. 11, 12, 13 and 14.

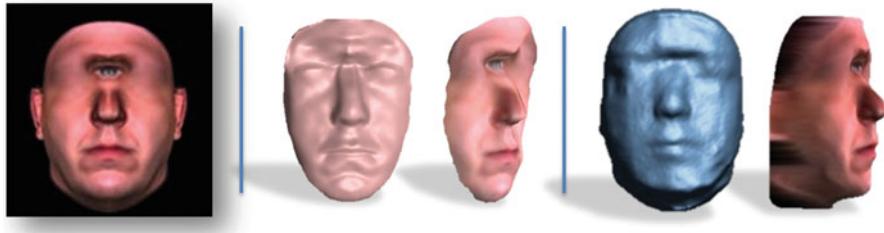


Fig. 10 Cyclops face reconstructions. *Left:* input image; *middle:* reconstruction result obtained by the face shape molding method of Kemelmacher and Basri [30]; *right:* our depth estimate. Each of the two results presents, *from left to right*, the estimated depth and the depth rendered using the query photo as texture. Depth-map results are shown rendered in 3D. Both methods use example of binocular faces. Although the results from [30] are more detailed estimates, their strong face-shape prior results in a depth-map with three separate eyes

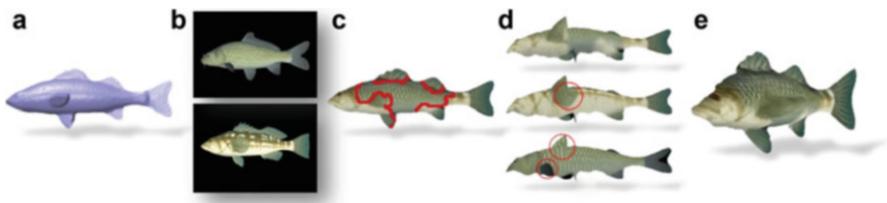


Fig. 11 Colorizing a fish. **(a)** The query fish depth-map. **(b)** Fish appearances from the appearance/depth mapping database used for colorization. **(c)** Output colorization applied to the query shape. Red marks delineate appearance regions taken from each example in **(b)**. **(d)** Query depth-map rendered with, *from top to bottom*, our output colors and the two original database appearances. The two *lower images* show clearly visible mismatches between appearances of the database objects and the query depth-map. Rendered view of the query depth-map and our output colors



Fig. 12 Fish colorization examples. *Top row* are the query depth-maps; *bottom row* are our output colorizations

Note in particular Fig. 14 where the example database objects are presented alongside the each result. It demonstrates how our use of a per pixel method allows for pairs of database image-maps to seamlessly mesh into the output image-map.



Fig. 13 Whole body colorization examples. Colorization results for three whole-body depth-maps. Each result shows, *from left to right*: the query depth-map. The depth-map colorized with a database appearance (*colors*), unmodified. These show the severe misalignment when appearances are not modified to fit the depth-map (in red). Finally, output colorization rendered using the query depth. In all these examples, a single database object was used. The results all show its appearance effectively warped to fit the depth-map

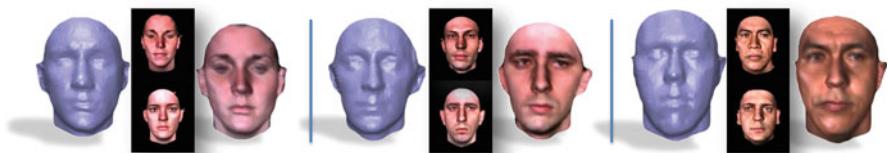


Fig. 14 Face colorization examples. Colorization results for three face depth-maps. Each result shows, *from left to right*: the query depth-map, appearances (*colors*) of two database examples used to colorize the query depth-map, and the output colorization rendered using the query depth

6 Conclusions

The method described in this chapter is one of the first to demonstrate how semantic information can be transferred from examples to a query by establishing dense correspondences between them. More robust alternatives to the normalized intensity patch representation used here have been proposed since this method was originally described, including the original Dense-SIFT (DSIFT) descriptor [50] and its extensions to multiscale (e.g., [22, 45]) some of which are discussed elsewhere in this book. Newer dense correspondence estimation methods have also been proposed over the years, in particular the SIFT flow [34, 35] and its own extensions [41]. All these could conceivably be used to improve the robustness and the method described in this chapter. In fact, one recent and successful approach to utilizing these modern tools for better depth estimation is presented in the next chapter. Yet the results presented here make clear the diverse potential of dense correspondence-base approaches to image understanding. In the next chapters, newer representations and correspondence estimation methods are described, harnessing ideas similar to the ones presented here, in order to extract semantic information from images.

References

1. Assa, J., Wolf, L.: Diorama construction from a single image. In: Eurographics, pp. 599–608 (2007)
2. Atick, J., Griffin, P., Redlich, A.: Statistical approach to shape from shading: reconstruction of three-dimensional face surfaces from single two-dimensional images. *Neural Comput.* **8**(6), 1321–1340 (1996)
3. Barron, J.T., Malik, J.: Shape, albedo, and illumination from a single image of an unknown object. In: Proceedings of Conference on Computer Vision Pattern Recognition, pp. 334–341. IEEE, Providence (2012)
4. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3D faces. In: Proceedings of ACM SIGGRAPH Conference on Computer Graphics, pp. 187–194. ACM/Addison-Wesley, New York (1999)
5. Burt, P., Adelson, E.: The laplacian pyramid as a compact image code. *IEEE Trans. Commun.* **30**, 532–540 (1983)
6. Chen, Y., Kim, T.K., Cipolla, R.: Inferring 3D shapes and deformations from single views. In: European Conference on Computer Vision, pp. 300–313. Springer, Heidelberg (2010)
7. Chen, X., Guo, Y., Zhou, B., Zhao, Q.: Deformable model for estimating clothed and naked human shapes from a single image. *Vis. Comput.* **29**(11), 1187–1196 (2013)
8. Cipolla, R., Fletcher, G., Giblin, P.: Surface geometry from cusps of apparent contours. In: Proceedings of International Conference on Computer Vision, pp. 858–863 (1995)
9. Cole, F., Isola, P., Freeman, W.T., Durand, F., Adelson, E.H.: Shapecollage: occlusion-aware, example-based shape interpretation. In: European Conference on Computer Vision, pp. 665–678. Springer, Heidelberg (2012)
10. Criminisi, A., Reid, I., Zisserman, A.: Single view metrology. *Int. J. Comput. Vis.* **40**(2), 123–148 (2000)
11. Cyberware: <http://www.cyberware.com/>
12. Delage, E., Lee, H., Ng, A.: Automatic single-image 3D reconstructions of indoor manhattan world scenes. In: Proceedings of the International Symposium of Robotics Research (ISRR), pp. 305–321 (2005)
13. Dovgord, R., Basri, R.: Statistical symmetric shape from shading for 3D structure recovery of faces. In: European Conference on Computer Vision, vol. 2, pp. 99–113. Springer, Berlin/Heidelberg (2004)
14. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. arXiv preprint (2014) [arXiv:1411.4734]
15. Gupta, A., Satkin, S., Efros, A.A., Hebert, M.: From 3D scene geometry to human workspace. In: Proceedings of Conference on Computer Vision Pattern Recognition, pp. 1961–1968. IEEE, Providence (2011)
16. Han, J., Zhou, K., Wei, L., Gong, M., Bao, H., Zhang, X., Guo, B.: Fast example-based surface texture synthesis via discrete optimization. *Vis. Comput.* **22**, 918–925 (2006)
17. Hassner, T.: Viewing real-world faces in 3D. In: Proceedings of International Conference on Computer Vision, pp. 3607–3614. IEEE, Sydney (2013). Available: www.openv.ac.il/home/hassner/projects/poses
18. Hassner, T., Basri, R.: Automatic depth-map colorization. In: Eurographics (short), pp. 73–76 (2006)
19. Hassner, T., Basri, R.: Example based 3D reconstruction from single 2D images. In: Beyond Patches Workshop at CVPR, p. 15 (2006)
20. Hassner, T., Basri, R.: Single view depth estimation from examples. arXiv preprint (2013) [arXiv:1304.3915]
21. Hassner, T., Zelnik-Manor, L., Leifman, G., Basri, R.: Minimal-cut model composition. In: International Conference on Shape Modeling and Applications (SMI' 05), pp. 72–81 (2005)
22. Hassner, T., Mayzels, V., Zelnik-Manor, L.: On sifts and their scales. In: Proceedings of Conference on Computer Vision Pattern Recognition, pp. 1522–1528. IEEE, Providence (2012)

23. Hassner, T., Assif, L., Wolf, L.: When standard RANSAC is not enough: cross-media visual matching with hypothesis relevancy. *Mach. Vis. Appl.* **25**(4), 971–983 (2014)
24. Hassner, T., Harel, S., Paz, E., Enbar, R.: Effective face frontalization in unconstrained images. In: Proceedings of Conference on Computer Vision Pattern Recognition (2015)
25. Hoiem, D., Efros, A., Hebert, M.: Automatic photo pop-up. *ACM Trans. Graph.* **24**(3), 577–584 (2005)
26. Hoiem, D., Efros, A., Hebert, M.: Geometric context from a single image. In: Proceedings of International Conference on Computer Vision, pp. 654–661. IEEE Computer Society, Beijing (2005)
27. Horn, B.: Obtaining Shape from Shading Information. *The Psychology of Computer Vision*. McGraw-Hill, New York (1975)
28. Karsch, K., Liu, C., Kang, S.B.: Depth extraction from video using non-parametric sampling. In: European Conference on Computer Vision, pp. 775–788. Springer, Heidelberg (2012)
29. Kearns, M., Mansour, Y., Ng, A.: An information-theoretic analysis of hard and soft assignment methods for clustering. In: Proceedings of the NATO Advanced Study Institute on Learning in Graphical Models, pp. 495–520. Kluwer Academic, Norwell (1998)
30. Kemelmacher, I., Basri, R.: Molding face shapes by example. In: European Conference on Computer Vision, p. 2006 (277–288)
31. Kemelmacher-Shlizerman, I., Seitz, S.: Face reconstruction in the wild. In: Proceedings of International Conference on Computer Vision, pp. 1746–1753. IEEE, Washington (2011)
32. Kraevoy, V., Sheffer, A.: Cross-parameterization and compatible remeshing of 3D models. *ACM Trans. Graph.* **23**(3), 861–869 (2004)
33. Leifman, G., Tal, A.: Mesh colorization. *Comput. Graph. Forum* **31**(2), 421–430 (2012)
34. Liu, C., Yuen, J., Torralba, A., Sivic, J., Freeman, W.: Sift flow: dense correspondence across different scenes. In: European Conference on Computer Vision, pp. 28–42 (2008). Available: people.csail.mit.edu/celiu/ECCV2008/
35. Liu, C., Yuen, J., Torralba, A.: Sift flow: dense correspondence across scenes and its applications. *Trans. Pattern Anal. Mach. Intell.* **33**(5), 978–994 (2011). Available: people.csail.mit.edu/celiu/SIFTflow/
36. Liu, F., Shen, C., Lin, G., Reid, I.: Learning depth from single monocular images using deep convolutional neural fields. arXiv preprint (2015) [arXiv:1502.07411]
37. Lowe, D.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **60**(2), 91–110 (2004)
38. Oswald, M.R., Toppe, E., Cremers, D.: Fast and globally optimal single view reconstruction of curved objects. In: Proceedings of Conference on Computer Vision Pattern Recognition, pp. 534–541. IEEE, Washington (2012)
39. Panagopoulos, A., Hadap, S., Samaras, D.: Reconstructing shape from dictionaries of shading primitives. In: Asian Conference on Computer Vision, pp. 80–94. Springer, Heidelberg (2013)
40. Praun, E., Sweldens, W., Schröder, P.: Consistent mesh parameterizations. In: Proceedings of ACM SIGGRAPH Conference on Computer Graphics, pp. 179–184. ACM, New York (2001)
41. Qiu, W., Wang, X., Bai, X., Yuille, A., Tu, Z.: Scale-space sift flow. In: Proceedings of Winter Conference on Applications of Computer Vision, pp. 1112–1119. IEEE, Steamboat Springs (2014)
42. Romdhani, S., Vetter, T.: Efficient, robust and accurate fitting of a 3D morphable model. In: Proceedings of International Conference on Computer Vision, p. 59 (2003)
43. Saxena, A., Sun, M., Ng, A.: Make3d: learning 3-D scene structure from a single still image. *Trans. Pattern Anal. Mach. Intell.* **31**(5), 824–840 (2009)
44. Schwing, A.G., Urtasun, R.: Efficient exact inference for 3D indoor scene understanding. In: European Conference on Computer Vision, pp. 299–313. Springer, Heidelberg (2012)
45. Tau, M., Hassner, T.: Dense correspondences across scenes and scales. *IEEE Trans. Pattern Anal. Mach. Intell.* **PP**(99), 1 (2015)
46. Thomas, A., Ferrari, V., Leibe, B., Tuytelaars, T., Gool, L.: Shape-from-recognition: recognition enables meta-data transfer. *Comput. Vis. Image Underst.* **113**(12), 1222–1234 (2009)
47. Toucan virtual museum. Available: <http://www.toucan.co.jp/indexE.html>

48. Turk, G.: Texture synthesis on surfaces. In: Proceedings of ACM SIGGRAPH Conference on Computer Graphics, pp. 347–354. ACM, New York (2001)
49. USF: DARPA Human-ID 3D Face Database: Courtesy of Prof. Sudeep Sarkar. University of South Florida, Tampa. <http://marthon.csee.usf.edu/HumanID/>
50. Vedaldi, A., Fulkerson, B.: VLFeat: an open and portable library of computer vision algorithms. In: Proceedings of International Conference on Multimedia, pp. 1469–1472 (2010). Available: www.vlfeat.org/
51. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: Proceedings of ACM SIGGRAPH Conference on Computer Graphics, pp. 479–488. ACM/Addison-Wesley, New York (2000)
52. Wei, L.Y., Levoy, M.: Texture synthesis over arbitrary manifold surfaces. In: Proceedings of ACM SIGGRAPH Conference on Computer Graphics, pp. 355–360. ACM, New York (2001)
53. Witkin, A.: Recovering surface shape and orientation from texture. *Artif. Intell.* **17**(1–3), 17–45 (1981)
54. Xiong, Y., Chakrabarti, A., Basri, R., Gortler, S.J., Jacobs, D.W., Zickler, T.: From shading to local shape. arXiv preprint (2014) [arXiv:1310.2916]
55. Ying, L., Hertzmann, A., Biermann, H., Zorin, D.: Texture and shape synthesis on surfaces. In: Proceedings of the 12th Eurographics Workshop on Rendering Techniques, pp. 301–312. Springer, Heidelberg (2001)
56. Zhou, K., Wang, X., Tong, Y., Desbrun, M., Guo, B., Shum, H.Y.: Texturemontage: seamless texturing of arbitrary surfaces from multiple images. *ACM Trans. Graph.* **24**(3), 1148–1155 (2005)

Depth Transfer: Depth Extraction from Videos Using Nonparametric Sampling

Kevin Karsch, Ce Liu, and Sing Bing Kang

Abstract In this chapter, a technique that automatically generates plausible depth maps from videos using nonparametric depth sampling is discussed. We demonstrate this method in cases where existing methods fail (nontranslating cameras and dynamic scenes). This technique is applicable to single images as well as videos. For videos, local motion cues are used to improve the inferred depth maps, while optical flow is used to ensure temporal depth consistency. For training and evaluation, a Microsoft Kinect-based system is developed to collect a large dataset containing stereoscopic videos with known depths, and this depth estimation technique outperforms the state-of-the-art on benchmark databases. This method can be used to automatically convert a monoscopic video into stereo for 3D visualization demonstrated through a variety of visually pleasing results for indoor and outdoor scenes, including results from the feature film *Charade*.

1 Introduction

Scene depth is useful for a variety of tasks, ranging from 3D modeling and visualization to robot navigation. It also facilitates spatial reasoning about objects in the scene in the context of scene understanding. In the growing 3D movie industry, knowing the scene depth greatly simplifies the process of converting 2D movies to their stereoscopic counterparts. The problem we are tackling in this paper is: given an arbitrary 2D video, how can we automatically extract plausible depth maps at every frame? At a deeper level, we investigate how we can reasonably extract depth maps in cases where conventional structure-from-motion and motion stereo fail.

K. Karsch (✉)

University of Illinois, Urbana, IL, USA

e-mail: kevin@kevinkarsch.com

C. Liu

Google Research, Cambridge, MA, USA

e-mail: celiu@microsoft.com

S.B. Kang

Microsoft Research, Redmond, WA, USA

e-mail: sbkang@microsoft.com

While many reconstruction techniques for extracting depth from video sequences exist, they typically assume moving cameras and static scenes. They do not work for dynamic scenes or for stationary, rotating, or variable focal length sequences. There are some exceptions, e.g., [41], which can handle some moving objects, but they still require camera motion to induce parallax and allow depth estimation.

In this chapter, we present a novel solution to generate depth maps from ordinary 2D videos; our solution also applies to single images. This technique is applicable to arbitrary videos, and works in cases where conventional depth recovery methods fail (static/rotating camera; change in focal length; dynamic scenes). Our primary contribution is the use of a nonparametric “depth transfer” approach for inferring temporally consistent depth maps without imposing requirements on the video (Sects. 3 and 4), including a method for improving the depth estimates of moving objects (Sect. 4.1). In addition, we introduce a new, ground truth stereo RGBD (RGB+depth) video dataset¹ (Sect. 5). We also describe how we synthesize stereo videos from ordinary 2D videos using the results of our technique (Sect. 7). Exemplar results are shown in Fig. 1.

2 Related Work

In this section, we briefly survey techniques related to our work, namely 2D-to-3D conversion techniques (single image and video, automatic and manual) and non-parametric learning.

2.1 Single Image Depth Estimation and 2D-to-3D

Early techniques for single image 2D-to-3D are semi-automatic; probably the most well known of them is the “tour-into-picture” work of Horry et al. [11]. Here, the user interactively adds planes to the single image for virtual view manipulation. Two other representative examples of interactive 2D-to-3D conversion systems are those of Oh et al. [26] (where a painting metaphor is used to assign depths and extract layers) and Zhang et al. [38] (where the user adds surface normals, silhouettes, and creases as constraints for depth reconstruction).

One of the earliest automatic methods for single image 2D-to-3D was proposed by Hoiem et al. [9]. They created convincing looking reconstructions of outdoor images by assuming an image could be broken into a few planar surfaces; similarly, Delage et al. developed a Bayesian framework for reconstructing indoor scenes [3]. Saxena et al. devised a supervised learning strategy for predicting depth from a single image [30], which was further improved to create realistic reconstructions for

¹Our dataset and code are publicly available at <http://kevinkarsch.com/depthtransfer>.

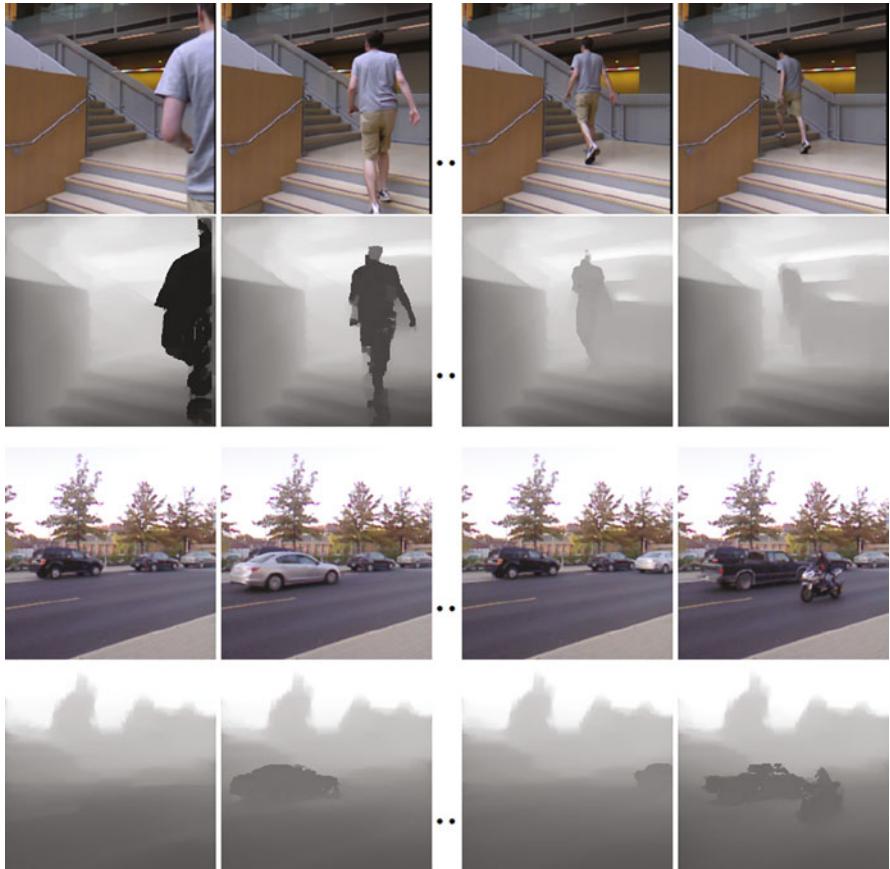


Fig. 1 Our technique takes a video sequence (*rows 1 and 3*) and automatically estimates per-pixel depth (*rows 2 and 4*). Our method does not require any cues from motion parallax or static scene elements; these videos were captured using a stationary camera with multiple moving objects

general scenes [31], and efficient learning strategies have since been proposed [1]. Better depth estimates have been achieved by incorporating semantic labels [20] or more sophisticated models [16].

Apart from learning-based techniques for extracting depths, more conventional techniques have been used based on image content. For example, repetitive structures have been used for stereo reconstruction from a single image [37]. The dark channel prior has also proven effective for estimating depth from images containing haze [7]. In addition, single image shape from shading is also possible for known (a priori) object classes [5, 6].

Compared to techniques here, we not only focus on depth from a *single image*, but also present a framework for using temporal information for enhanced and time-coherent depth when multiple frames are available.

The approach closest to ours is the contemporaneous work of Konrad et al. [13, 14], which also uses some form of nonparametric depth sampling to automatically convert monocular images into stereoscopic images. They make similar assumptions to ours (e.g., appearance and depth are correlated), but use a simpler optimization scheme, and argue that the use of SIFT flow only provides marginal improvements (opposed to not warping candidates via SIFT flow). Our improvements are twofold: their depth maps are computed using the median of the candidate disparity fields and smoothed with a cross bilateral filter, while we consider the candidate depths (and depth gradients) on a per-pixel basis. Furthermore, we propose novel solutions to incorporate temporal information from videos, whereas the method of Konrad et al. works on single images. We also show a favorable comparison in Table 2.

2.2 Video Depth Estimation and 2D-to-3D

A number of video 2D-to-3D techniques exist, but many of them are interactive. Examples of interactive systems include those of Guttman et al. [4] (scribbles with depth properties are added to frames for video cube propagation), Ward et al. [36] (“depth templates” for primitive shapes are specified by the user, which the system propagates over the video), and Liao et al. [17] (user interaction to propagate structure-from-motion information with aid of optical flow).

There are a few commercial solutions that are automatic, e.g., Tri-Def DDD, but anecdotal tests using the demo version revealed room for improvement. There is even hardware available for real-time 2D-to-3D video conversion, such as the DA8223 chip by Dialog Semiconductor. It is not clear, however, how well the conversion works, since typically simple assumptions are made on what constitute foreground and background areas based on motion estimation. There are also a number of production houses specializing in 2D-to-3D conversions (e.g., In-Three [34] and Identity FX, Inc.), but their solutions are customized and likely to be manual-intensive. Furthermore, their tools are not publicly available.

If the video is mostly amenable to structure-from-motion and motion stereo, such technologies can be used to compute dense depth maps at every frame of the video. One system that does this is that of Zhang et al. [39], which also showed how depth-dependent effects such as synthetic fog and depth-of-focus can be achieved. While this system (and others [40, 41]) can handle some moving objects, there is significant reliance on camera motion that induces parallax.

2.3 Nonparametric Learning

As the conventional notion of image correspondences was extended from different views of the same 3D scene to semantically similar, but different, 3D scenes [22], the information such as labels and motion can be transferred from a large database

to parse an input image [21]. Given an unlabeled input image and a database with known per-pixel labels (e.g., sky, car, tree, window), their method works by transferring the labels from the database to the input image based on SIFT flow, which is estimated by matching pixel-wise, dense SIFT features. This simple methodology has been widely used in many computer vision applications such as image super resolution [33], image tagging [28], and object discovery [29].

We build on this work by transferring depth instead of semantic labels. Furthermore, we show that this “transfer” approach can be applied in a continuous optimization framework (Sect. 3), whereas their method used discrete MRFs.

3 Nonparametric Depth Estimation by Continuous Label Transfer

We leverage recent work on nonparametric learning [19], which avoids explicitly defining a parametric model and requires fewer assumptions as in past methods (e.g., [20, 30, 31]). This approach also scales better with respect to the training data size, requiring virtually no training time. Our technique imposes no requirements on the video, such as motion parallax or sequence length, and can even be applied to a single image. We first describe our depth estimation technique as it applies to the single images below, and in Sect. 4 we discuss novel additions that allow for improved depth estimation in videos.

Our depth transfer approach, outlined in Fig. 2, has three stages. First, given the database RGBD images, we find candidate images in the database that are “similar” to the input image in RGB space. Then, a warping procedure (SIFT Flow [22]) is applied to the candidate images and depths to align them with the input. Finally, an optimization procedure is used to interpolate and smooth the warped candidate depth values; this results in the inferred depth.

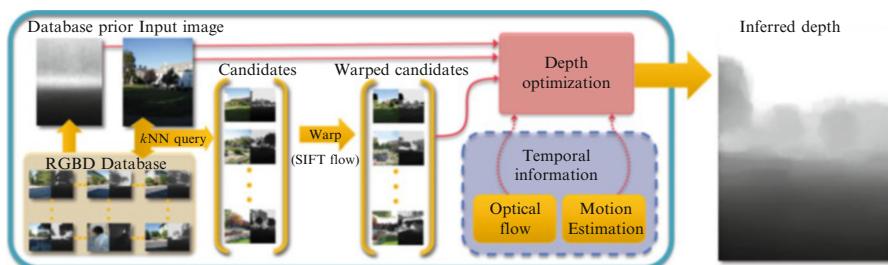


Fig. 2 Our pipeline for estimating depth. Given an input image, we find matching candidates in our database and warp the candidates to match the structure of the input image. We then use a global optimization procedure to interpolate the warped candidates [Eq. (2)], producing per-pixel depth estimates for the input image. With temporal information (e.g., extracted from a video), our algorithm can achieve more accurate, temporally coherent depth

Our core idea is that scenes with similar semantics should have roughly similar depth distributions when densely aligned. In other words, images of semantically alike scenes are expected to have similar depth values in regions with similar appearance. Of course, not all of these estimates will be correct, which is why we find several candidate images and refine and interpolate these estimates using a global optimization technique that considers factors other than just absolute depth values.

3.1 RGBD Database

Our system requires a database of RGBD images and/or videos. We have collected our own RGBD video dataset, as described in Sect. 5; a few already exist online, though they are for single images only.²

3.2 Candidate Matching and Warping

Given a database and an input image, we compute high level image features (we use GIST [27] and optical flow features) for each image or frame of video in the database as well as the input image. We then select the top K ($= 7$ in our work, unless explicitly stated) matching frames from the database, but ensure that each video in the database contributes no more than one matching frame. This forces matching images to be from differing viewpoints, allowing for greater variety among matches. We call these matching images *candidate images*, and their corresponding depths *candidate depths*.

Because the candidate images match the input closely in feature space, it is expected that the overall semantics of the scene are roughly similar. We also make the critical assumption that the distribution of depth is comparable among the input and candidates. However, we want pixel-to-pixel correspondences between the input and all candidates, as to limit the search space when inferring depth from the candidates.

We achieve this pixel-to-pixel correspondence through SIFT flow [22]. Using SIFT flow, we estimate warping functions $\psi_i, i \in \{1, \dots, K\}$ for each candidate image. These warping functions map pixel locations from a given candidate's domain to pixel locations in the input's domain. The warping functions can be one-to-many.

²Examples: Make3D range image dataset (<http://make3d.cs.cornell.edu/data.html>), B3DO dataset (<http://kinectdata.com/>), NYU depth datasets (<http://cs.nyu.edu/~silberman/datasets/>), RGB-D dataset (<http://www.cs.washington.edu/rbgd-dataset/>), and our own (<http://kevinkarsch.com/depthtransfer>).

3.3 Features for Candidate Image Matching

In order to find candidate images which match the input image/sequence semantically and in terms of depth distribution, we use a combination of GIST features [27] and features derived from optical flow (used only in the case of videos, as in Sect. 4). To create flow features for a video, we first compute optical flow (using Liu’s implementation [18]) for each pair of consecutive frames, which defines a warping from frame i to frame $i + 1$. If the input is a single image or the last image in a video sequence, we consider this warping to be the identity warp. Then, we segment the image into $b \times b$ uniform blocks and compute the mean and standard deviation over the flow field in each block for both components of the flow (horizontal and vertical warpings), and for the second moments as well (each component squared). This leads to eight features per block, for a total of $8b^2$ features per image. We use $b = 4$ for our results.

To determine the matching score between two images, we take a linear combination of the difference in GIST and optical flow features described above. Denoting G_1, G_2 and F_1, F_2 as the GIST and flow feature vectors for two images respectively, we define the matching score as:

$$(1 - \omega)||G_1 - G_2|| + \omega||F_1 - F_2||, \quad (1)$$

where $\omega = 0.5$ in our implementation.

3.4 Depth Optimization

Each warped candidate depth is deemed to be a rough approximation of the input’s depth map. Unfortunately, such candidate depths may still contain inaccuracies and are often not spatially smooth. Instead, we generate the most likely depth map by considering all of the warped candidates, optimizing with spatial regularization in mind.

Let \mathbf{L} be the input image and \mathbf{D} the depth map we wish to infer. We minimize:

$$-\log(P(\mathbf{D}|\mathbf{L})) = E(\mathbf{D}) = \sum_{i \in \text{pixels}} E_t(\mathbf{D}_i) + \alpha E_s(\mathbf{D}_i) + \beta E_p(\mathbf{D}_i) + \log(Z), \quad (2)$$

where Z is the normalization constant of the probability, and α and β are parameters ($\alpha = 10, \beta = 0.5$). For a single image, our objective contains three terms: data (E_t), spatial smoothness (E_s), and database prior (E_p).

Data Cost The data term measures how close the inferred depth map \mathbf{D} is to each of the warped candidate depths, $\psi_j(C_i^{(j)})$. This distance measure is defined by ϕ , a robust error norm (we use an approximation to the $L1$ norm, $\phi(x) = \sqrt{x^2 + \epsilon}$, with $\epsilon = 10^{-4}$). We define the data term as:

$$E_t(\mathbf{D}_i) = \sum_{j=1}^K w_i^{(j)} \left[\phi(\mathbf{D}_i - \psi_j(C_i^{(j)})) + \gamma [\phi(\nabla_x \mathbf{D}_i - \psi_j(\nabla_x C_i^{(j)})) + \phi(\nabla_y \mathbf{D}_i - \psi_j(\nabla_y C_i^{(j)}))] \right], \quad (3)$$

where $w_i^{(j)}$ is a confidence measure of the accuracy of the j th candidate’s warped depth at pixel i , and $K (= 7)$ is the total number of candidates. We measure not only absolute differences, but also relative depth changes, i.e., depth gradients (∇_x, ∇_y are spatial derivatives). The latter terms of Eq. (3) enforce similarity among candidate depth gradients and inferred depth gradients, weighted by $\gamma (= 10)$.

Note that we warp the candidate’s gradients in depth, rather than warping depth and then differentiating. Warping depth induces many new discontinuities, which would result in large gradients solely due to the warping, rather than actual depth discontinuities in the data. The downside of this is that the warped gradients are not integrable, but this does not signify as we optimize over depth anyhow (ensuring the resulting depth map is integrable).

Some of the candidate depth values will be more reliable than others, and we model this reliability with a confidence weighting for each pixel in each candidate image (e.g., $w_i^{(j)}$ is the weight of the i th pixel from the j th candidate image). We compute these weights by comparing per-pixel SIFT descriptors, obtained during the SIFT flow computation of both the input image and the candidate images:

$$w_i^{(j)} = (1 + e^{(||\mathbf{S}_i - \psi_j(\mathcal{S}_i^{(j)})|| - \mu_s)/\sigma_s})^{-1}, \quad (4)$$

where \mathbf{S}_i and $\mathcal{S}_i^{(j)}$ are the SIFT feature vectors at pixel i in candidate image j . We set $\mu_s = 0.5$ and $\sigma_s = 0.01$. Notice that the candidate image’s SIFT features are computed first, and then warped using the warping function (ψ_j) calculated with SIFT flow.

Spatial Smoothness While we encourage spatial smoothness, we do not want the smoothness applied uniformly to the inferred depth, since there is typically some correlation between image appearance and depth. Therefore, we assume that regions in the image with similar texture are likely to have similar, smooth depth transitions, and that discontinuities in the image are likely to correspond to discontinuities in depth.

We enforce appearance-dependent smoothness with a per-pixel weighting of the spatial regularization term such that this weight is large where the image gradients are small and vice versa. We determine this weighting by applying a sigmoidal function to the gradients, which we found to produce more pleasing inferred depth maps than using other boundary detecting schemes such as [10, 24].

The smoothness term is specified as:

$$E_s(\mathbf{D}_i) = s_{x,i}\phi(\nabla_x \mathbf{D}_i) + s_{y,i}\phi(\nabla_y \mathbf{D}_i). \quad (5)$$

The depth gradients along x and y ($\nabla_x \mathbf{D}, \nabla_y \mathbf{D}$) are modulated by soft thresholds (sigmoidal functions) of image gradients in the same directions ($\nabla_x \mathbf{L}, \nabla_y \mathbf{L}$), namely $s_{x,i} = (1 + e^{(|\nabla_x \mathbf{L}_i| - \mu_L)/\sigma_L})^{-1}$ and $s_{y,i} = (1 + e^{(|\nabla_y \mathbf{L}_i| - \mu_L)/\sigma_L})^{-1}$. We set $\mu_L = 0.05$ and $\sigma_L = 0.01$.

Prior We also incorporate assumptions from our database that will guide the inference when pixels have little or no influence from other terms (due to weights w and s):

$$E_p(\mathbf{D}_i) = \phi(\mathbf{D}_i - \mathcal{P}_i). \quad (6)$$

We compute the prior \mathcal{P} by averaging all depth images in our database.

3.5 Numerical Optimization Details

Equation (2) requires an unconstrained, nonlinear optimization, and we use iteratively reweighted least squares to minimize our objective function. We choose IRLS because it is a fast alternative for solving unconstrained, nonlinear minimization problems such as ours. IRLS works by approximating the objective by a linear function of the parameters and solving the system by minimizing the squared residual (e.g., with least squares); it is repeated until convergence.

As an example, consider a sub-portion of our objective, the second term in Eq. (3) for candidate #1: $\sum_{i \in \text{pixels}} \phi(\nabla_x \mathbf{D}_i - \psi_1(\nabla_x C_i^{(1)}))$. To minimize, we differentiate with respect to depth and set equal to zero (letting $b = [\psi_1(\nabla_x C_1^{(1)}), \dots, \psi_1(\nabla_x C_N^{(1)})]^T$, keeping in mind that $\phi(x) = \sqrt{x^2 + \epsilon}$, and ∇_x is the horizontal derivative):

$$\sum_{i \in \text{pixels}} \frac{d}{d\mathbf{D}} \phi(\nabla_x \mathbf{D}_i - b) \nabla_x (\nabla_x \mathbf{D}_i - b) = 0. \quad (7)$$

We rewrite this using matrix notation as $G_x^T W (G_x \mathbf{D} - b) = 0$, where G_x is the $N \times N$ linear operator corresponding to a horizontal gradient (e.g., $[G_x \mathbf{D}]_i = \nabla_x \mathbf{D}_i$), and W is a diagonal matrix of “weights” computed from the nonlinear portion of the derivative of ϕ . By fixing W (computed for a given \mathbf{D}), we arrive at the following IRLS solution:

$$W = \text{diag} \left(\frac{d}{d\mathbf{D}} \phi(\nabla_x \mathbf{D}^{(t)} - b) \right)$$

$$\mathbf{D}^{(t+1)} = (G_x^T W G_x)^+ (G_x^T W b), \quad (8)$$

where $(\cdot)^+$ is the pseudoinverse and $D^{(t)}$ is the inferred depth at iteration t . We have found that thirty iterations typically approximates convergence. Extending IRLS to our full objective follows the same logic.³

In the general case of videos, the size of this system can be very large (number of pixels \times number of frames squared), although it will be sparse because of the limited number of pairwise interactions in the optimization. Still, given modern hardware limitations, we cannot solve this system directly, so we also must use an iterative method to solve the least squares system at each iteration of our IRLS procedure; we use preconditioned conjugate gradient and construct a preconditioner using incomplete Cholesky factorization.

Because we use iterative optimization, starting from a good initial estimate is helpful for quick convergence with fewer iterations. We have found that initializing with some function of the warped candidate depths provide a decent starting point, and we use the median value (per-pixel) of all candidate depths in our implementation (i.e., $\mathbf{D}_i^{(0)} = \text{median}\{\phi_1(C_i^{(1)}), \dots, \phi_K(C_i^{(K)})\}$). The method of Konrad et al. [13, 14] also uses the median of retrieved depth maps, but this becomes their final depth estimate. Our approach uses the median only for initialization and allows any of the warped candidate depths to contribute and influence the final estimate (dictated by the objective function). Figure 3 shows the optimization process for different initializations.

One issue is that this optimization can require a great deal of data to be stored concurrently in memory (several GBs for a standard definition clip of a few seconds). Solving this optimization efficiently, both in terms of time and space, is beyond the scope of this chapter.

4 Improved Depth Estimation for Video

Generating depth maps frame-by-frame without incorporating temporal information often leads to temporal discontinuities; past methods that ensure temporal coherence rely on a translating camera and static scene objects. Here, we present a framework that improves depth estimates and enforces temporal coherence for *arbitrary video sequences*. That is, our algorithm is suitable for videos with moving scene objects and rotating/zooming views where conventional SfM and stereo techniques fail. (Here, we assume that zooming induces little or no parallax.)

Our idea is to incorporate temporal information through additional terms in the optimization that ensure (a) depth estimates are consistent over time and (b) that moving objects have a depth similar to their contact point with the ground. Each frame is processed the same as in the single image case (candidate matching and warping), except that now we employ a global optimization (described below) that

³For further details and discussion of IRLS, see the appendix of Liu’s thesis [18].

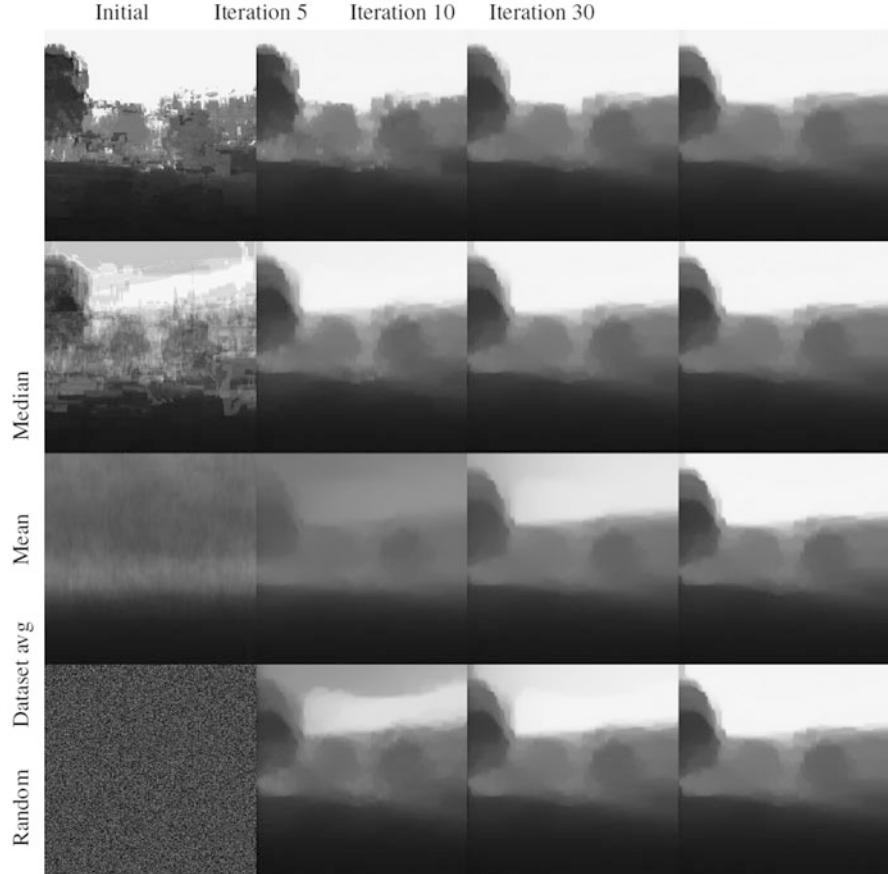


Fig. 3 Result of our optimization from different starting points (initialization methods on *left*). Our method typically converges to the same point given any initialization, but the median method (see text) is usually the most efficient. All depths are displayed at the same scale. The input image and depth can be seen in Fig. 2

infers depth for the entire sequence at once, incorporating temporal information from all frames in the video. Figure 4 illustrates the importance of these additional terms in our optimization.

We formulate the objective for handling video by adding two terms to the single image objective function:

$$E_{\text{video}}(\mathbf{D}) = E(\mathbf{D}) + \sum_{i \in \text{pixels}} v E_c(\mathbf{D}_i) + \eta E_m(\mathbf{D}_i), \quad (9)$$



Fig. 4 Importance of temporal information. *Left*: input frames. *Mid-left*: predicted depth without temporal information. Note that the car is practically ignored here. *Mid-right*: predicted depth with temporal information, with the depth of the moving car recovered. *Right*: detected moving object

where E_c encourages temporal coherence while E_m uses motion cues to improve the depth of moving objects. The weights ν and η balance the relative influence of each term ($\nu = 100$, $\eta = 5$).

We model temporal coherence first by computing per-pixel optical flow for each pair of consecutive frames in the video (using Liu's publicly available code [18]). We define the *flow difference*, ∇_{flow} , as a linear operator which returns the change in the flow across two corresponding pixels, and model the coherence term as:

$$E_c(\mathbf{D}_i) = s_{t,i} \phi(\nabla_{\text{flow}} \mathbf{D}_i). \quad (10)$$

We weight each term by a measure of flow confidence:

$$s_{t,i} = (1 + e^{-(||\nabla_{\text{flow}} \mathbf{L}_i|| - \mu_L)/\sigma_L})^{-1},$$

which intuitively is a soft threshold on the reprojection error ($\mu_L = 0.05$, $\sigma_L = 0.01$). Minimizing the weighted flow differences has the effect of temporally smoothing inferred depth in regions where optical flow estimates are accurate.

To handle motion, we detect moving objects in the video (Sect. 4.1) and constrain their depth such that these objects touch the floor. Let m be the binary motion

segmentation mask and \mathcal{M} the depth in which connected components in the segmentation mask contact the floor. We define the motion term as:

$$E_m(\mathbf{D}_i) = m_i \phi(\mathbf{D}_i - \mathcal{M}_i). \quad (11)$$

4.1 Detecting Moving Objects

Differentiating moving and stationary objects in the scene can be a useful cue when estimating depth. Here we describe our algorithm for detecting objects in motion in non-translating movies (i.e., static, rotational, and variable focal length videos).⁴ Note that there are many existing techniques for detecting moving objects (e.g., [8, 32]); we use what we consider to be easy to implement and effective for our purpose of depth extraction.

First, to account for dynamic exposure changes throughout the video, we find the image with the lowest overall intensity in the sequence and perform histogram equalization on all other frames in the video. We use this image so as to not propagate spurious noise found in brighter images. Next, we use RANSAC on point correspondences to compute the dominant camera motion (modeled using homography) to align neighboring frames in the video. Median filtering is then used on the stabilized images to extract the background B (ideally, without all the moving objects).

In our method, the likelihood of a pixel being in motion depends on how different it is from the background, weighted by the optical flow magnitude which is computed between stabilized frames (rather than between the original frames). We use relative differencing (relative to background pixels) to reduce reliance on absolute intensity values, and then threshold to produce a mask:

$$m_{i,k} = \begin{cases} 1 & \text{if } ||\text{flow}_{i,k}|| \frac{||W_{i,k} - B_i||^2}{B_i} > \tau \\ 0 & \text{otherwise,} \end{cases} \quad (12)$$

where $\tau = 0.01$ is the threshold, and $W_{i,k}$ is the k th pixel of the i th stabilized frame (i.e., warped according to the homography that aligns W with B). This produces a motion estimate in the background's coordinate system, so we apply the corresponding inverse homography to each warped frame to find the motion relative to each frame of the video. This segmentation mask is used [as in Eq. (11)] to improve depth estimates for moving objects in our optimization. Figure 5 illustrates this technique.

⁴In all other types of videos (e.g., those with parallax or fast moving objects/pose), we do not employ this algorithm; equivalently we set the motion segmentation weight to zero ($\eta = 0$).

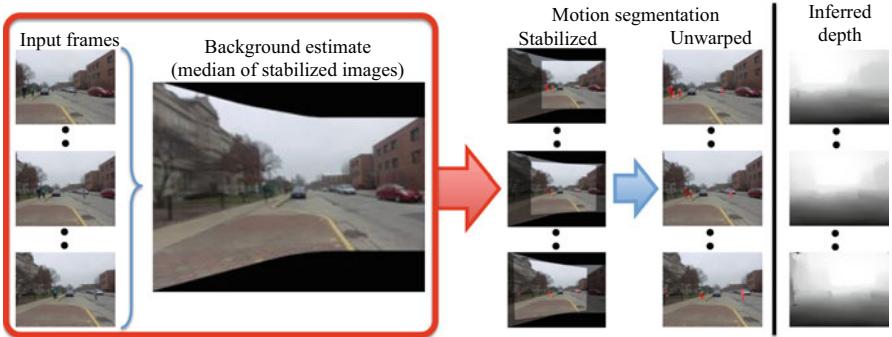


Fig. 5 Example of our motion segmentation applied to a rotating sequence. We first estimate homographies to stabilize the video frames and create a clean background image using a temporal median filter. We then evaluate our estimate of motion thresholding metric on the stabilized sequences, and un warp the result (via the corresponding inverse homography) to segment the motion in the original sequence. We can then improve our inferred depth using this segmentation. Note that this technique is applicable to all video sequences that do not contain parallax induced from camera motion

5 MSR-V3D Dataset

In order to train and test our technique on image/video input, we collected a dataset containing stereo video clips for a variety of scenes and viewpoints (116 indoor, 61 outdoor). The dataset primarily contains videos recorded from a static viewpoint with moving objects in the scene (people, cars, etc.). There are also 100 indoor frames (single images) in addition to the 116 indoor video clips within the database (stereo RGB, depth for the left frame). These sequences come from four different buildings in two cities and contain substantial scene variation (e.g., hallways, rooms, foyers). Each clip is filmed with camera viewpoints that are either static or slowly rotated. We have entitled our dataset the Microsoft Research Stereo Video + Depth (MSR-V3D), and it is available online at <http://kevinkarsch.com/depthtransfer>.

We captured the MSR-V3D dataset with two side-by-side, vertically mounted Microsoft Kinects shown in Fig. 6 (positioned about 5 cm apart). We collected the color images from both Kinects and only the depth map from the left Kinect. For each indoor clip, the left stereo view also contains view-aligned depth from the Kinect. Due to IR interference, depth for the right view was not captured indoors, and depth was totally disregarded outdoors due to limitations of the Kinect.

We also collected outdoor data with our stereo device. However, because the Kinect cannot produce depth maps outdoors due to IR interference from the sunlight, we could not use these sequences for training. We attempted to extract ground truth disparity between stereo pairs, but the quality/resolution of Kinect images was too low to get decent estimates. We did, however, use this data for testing and evaluation purposes.

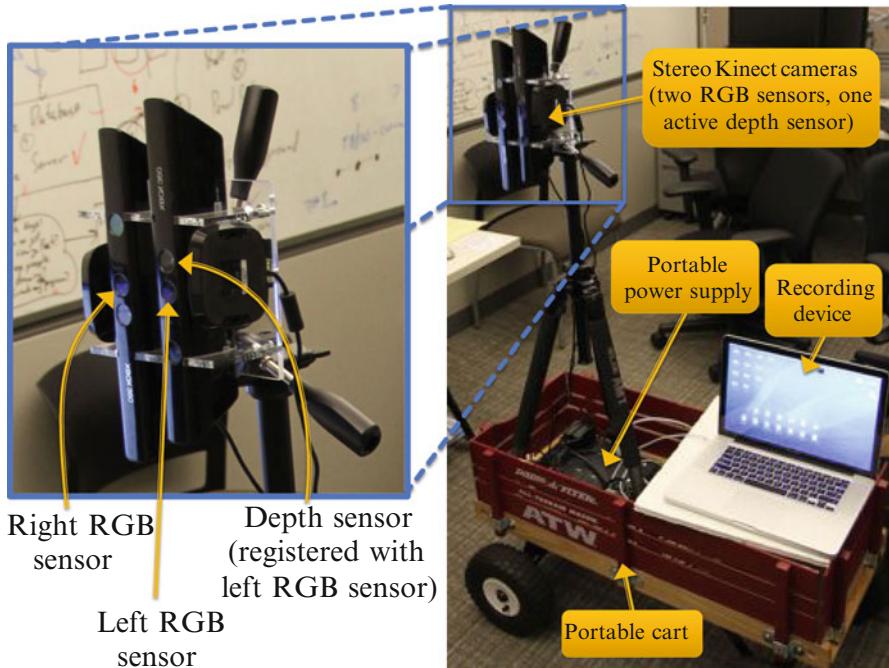


Fig. 6 Our stereo-RGBD collection rig consists of two side-by-side Microsoft Kinects. The rig is mobile through the use of an uninterrupted power supply, laptop, and rolling mount

Because the Kinect estimates depth by triangulating a pattern of projected infrared (IR) dots, multiple Kinects can interfere with each other, causing noisy and inaccurate depth. Thus, we mask out the depth sensor/IR projector from the rightmost Kinect, and only collect depth corresponding to the left views. This is suitable for our needs, as we only need to estimate depth for the input (left) sequence.

Kinect depth is also susceptible to “holes” in the data caused typically by surface properties, disoccluded/interfered IR pattern, or because objects are simply too far away from the device. For training, we disregard all pixels of the videos which contain holes, and for visualization, we fill the holes in using a naïve horizontal dilation approach.

6 Experiments

In this section, we show results of experiments involving single image and video depth extraction. We also discuss the importance of scale associated with training and the effect of the number of candidates used for depth hypothesis.

Table 1 Comparison of depth estimation errors on the Make3D range image dataset

Method	Rel	\log_{10}	RMS
Depth MRF [30]	0.530	0.198	16.7
Make3D [31]	0.370	0.187	–
Feedback cascades [16]	–	–	15.2
Semantic labels [20]	0.375	0.148	–
Depth transfer (ours)	0.361	0.148	15.1

Using our single image technique, our method achieves state-of-the-art results in each metric (*rel* is relative error, *RMS* is root mean squared error; details in text)

Bold values indicate the best result for a given error metric

6.1 Single Image Results

We evaluate our technique on two single image RGBD datasets: the Make3D range image dataset [31] and the NYU depth dataset [25].

6.1.1 Make3D Range Image Dataset

Of the 534 images in the Make3D dataset, we use 400 for testing and 134 for training (the same as was done before, e.g., [16, 20, 30, 31]). We report error for three common metrics in Table 1. Denoting \mathbf{D} as estimated depth and \mathbf{D}^* as ground truth depth, we compute *relative (rel) error* $\frac{|\mathbf{D} - \mathbf{D}^*|}{\mathbf{D}^*}$, *log₁₀ error* $|\log_{10}(\mathbf{D}) - \log_{10}(\mathbf{D}^*)|$, and *root mean squared (RMS) error* $\sqrt{\sum_{i=1}^N (\mathbf{D}_i - \mathbf{D}_i^*)^2 / N}$. Error measures are averaged over all pixels/images in the test set. Our estimated depth maps are computed at 345×460 pixels (maintaining the aspect ratio of the Make3D dataset input images).

Our method is as good as or better than the state-of-the-art for each metric. Note that previously no method achieved state-of-the-art results in more than one metric. We show several examples in Fig. 7. Thin structures (e.g., trees and pillars) are usually recovered well; however, fine structures are occasionally missed due to spatial regularization (such as the poles in the bottom-right image of Fig. 7).

6.1.2 NYU Depth Dataset

We report additional results for the NYU depth dataset [25], which consists of 1449 indoor RGBD images captured with a Kinect. Holes from the Kinect are disregarded during training (candidate searching and warping), and are not included in our error analysis.



Fig. 7 Single image results obtained on test images from the Make3D dataset. Each result contains the following four images (*from left to right*): original photograph, ground truth depth from the dataset, our inferred depth, and our synthesized anaglyph image. The depth images are shown in log scale. *Darker pixels* indicate nearby objects (black is roughly 1 m away) and *lighter pixels* indicate objects farther away (white is roughly 80 m away). Each pair of ground truth and inferred depths is displayed at the same scale

Table 2 Quantitative evaluation of our method and the methods of Konrad et al. on the NYU depth dataset

Method	Rel	\log_{10}	RMS
Depth transfer (NYU)	0.350	0.131	1.2
Depth transfer (MSR-V3D)	0.806	0.231	3.7
Depth fusion [13]	0.368	0.135	1.3
Depth fusion (no warp) [14]	0.371	0.137	1.3
NYU average depth [†]	0.491	0.327	4.3
NYU per-pixel average ^{††}	0.561	0.164	20.1

All methods are trained on the NYU dataset using a hold-one-out scheme, except for Depth Transfer^(MSR-V3D) which is trained using our own dataset (containing significantly different indoor scenes). Each method uses seven candidate images ($K = 7$). It is interesting that the per-pixel average (^{††}) performs worse than a single depth value ([†]), evidencing that these metrics are likely not very perceptual. Bold values indicate the best result for a given error metric.

Quantitative results are shown in Table 2, and Fig. 8 shows qualitative results. For comparison, we train our algorithm in two different ways and report results on each. One method trains (e.g., selects RGBD candidates) from the NYU depth dataset (holding out the particular example that is being tested), and the other method trains using all RGBD images found in MSR-V3D. We observe a significant degradation in results when training using our own dataset (MSR-V3D), likely because our dataset contains many fewer scenes than the NYU dataset, and a less diverse set of examples (NYU contains home, office, and many unique interiors, a total of 464; ours is primarily office-type scenes from four different buildings). This also suggests that generalization to interior scenes is much more difficult than outdoor, which coincides with the intuition that indoor scenes, on average, contain more variety than outdoor scenes.

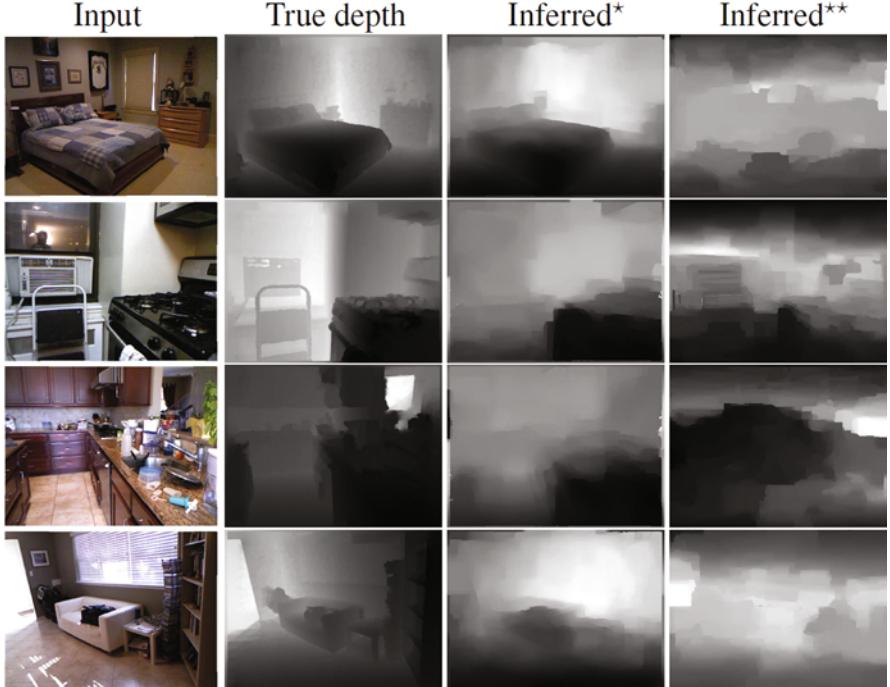


Fig. 8 Single image results obtained on the NYU Depth Dataset. Each result contains the following four images (*from left to right*): original photograph, ground truth depth from the dataset, our inferred depth trained on the NYU depth dataset (*), holding out the particular image), and our inferred depth trained on our MSR-V3D dataset (**). The top left result is in the fifth percentile (in terms of \log_{10} error), the top right is in the bottom fifth percentile, and both bottom results are near the median. Notice how the NYU-trained results are much better; this is likely due to the high variation of indoor images (MSR-V3D images appear very dissimilar to the NYU ones), and is also reflected in the quantitative results (Table 1). Holes in the true depth maps have been filled using the algorithm in [25], and each pair of ground truth and inferred depths are displayed at the same scale

Additionally, we compare our method to the single image approaches of Konrad et al. [13, 14]. Both of these methods choose candidate RGBD images using global image features (as in our work), but dissimilar from our optimization, they compute depth as the median value of the candidate depth images (per-pixel), and post-process their depths with a cross-bilateral filter. [13] applies SIFT flow to warp/align depth (similar to our method), while [14] opts not to for efficiency (their main focus is 2D-to-3D conversion rather than depth estimation). We show improvements over both of these methods, but more importantly, our depth and 2D-to-3D conversion methods apply to videos rather than only single images.

Finally, we compare all of these results to two baselines: the average depth value over the entire NYU dataset (\dagger) and the per-pixel average of the NYU depth (\ddagger). We

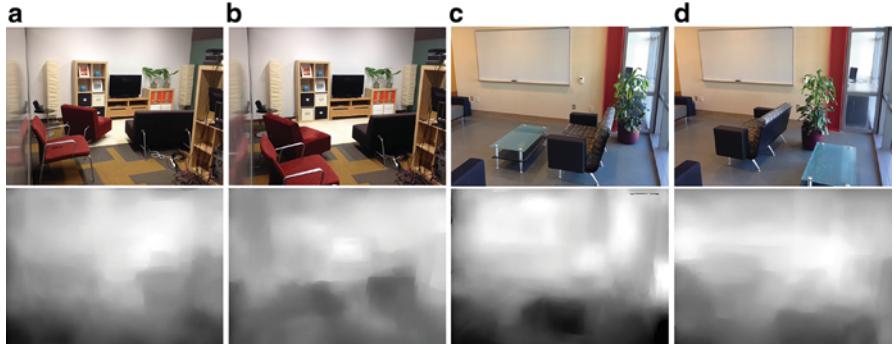


Fig. 9 Demonstration of our method on scenes where objects have been repositioned. The chairs and couch in **(a)** are moved closer in **(b)**, and furniture in **(c)** is repositioned in **(d)**. Although the depth maps contain noticeable errors, it is evident that our estimates are influenced by factors other than appearance. For example, although the chairs and couch in **(a)**, **(b)** have the same appearance, the estimated depth of **(b)** is noticeably closer in the chair/couch regions, as it should be

observe that our method significantly outperforms these baselines in three metrics (relative, \log_{10} , and RMS error) when trained on the NYU dataset (hold-one-out). Our error rates increase by training on our dataset (MSR-V3D), but this training method still outperforms the baselines in several metrics. Note that these baselines do use some knowledge from the NYU dataset, unlike our method trained on MSR-V3D.

One concern of our method may be its reliance on appearance. In Fig. 9, we demonstrate that although appearance partially drives our depth estimates, other factors such as spatial relationships, scale, and orientation necessarily contribute. For example, photographing the same scene with objects in different configurations will lead to different depth estimates, as would be expected.

6.2 Video Results

Our technique works well for videos of many types scenes and video types (Figs. 1, 4, 5, 10, 11, and 12). We use the dataset we collected in Sect. 5 to validate our method for videos (we know of no other existing methods/datasets to compare to). This dataset contains ground truth depth and stereo image sequences for four different buildings (referred to as Buildings 1, 2, 3, and 4), and to gauge our algorithm’s ability to generalize, *we only use data from Building 1 for training*. We still generate results for Building 1 by holding each particular example out of the training set during inference.

We show quantitative results in Table 3 and qualitative results in Fig. 11. We calculate error using the same metrics as in our single image experiments, and to make these results comparable with Table 1, we globally rescale the ground truth

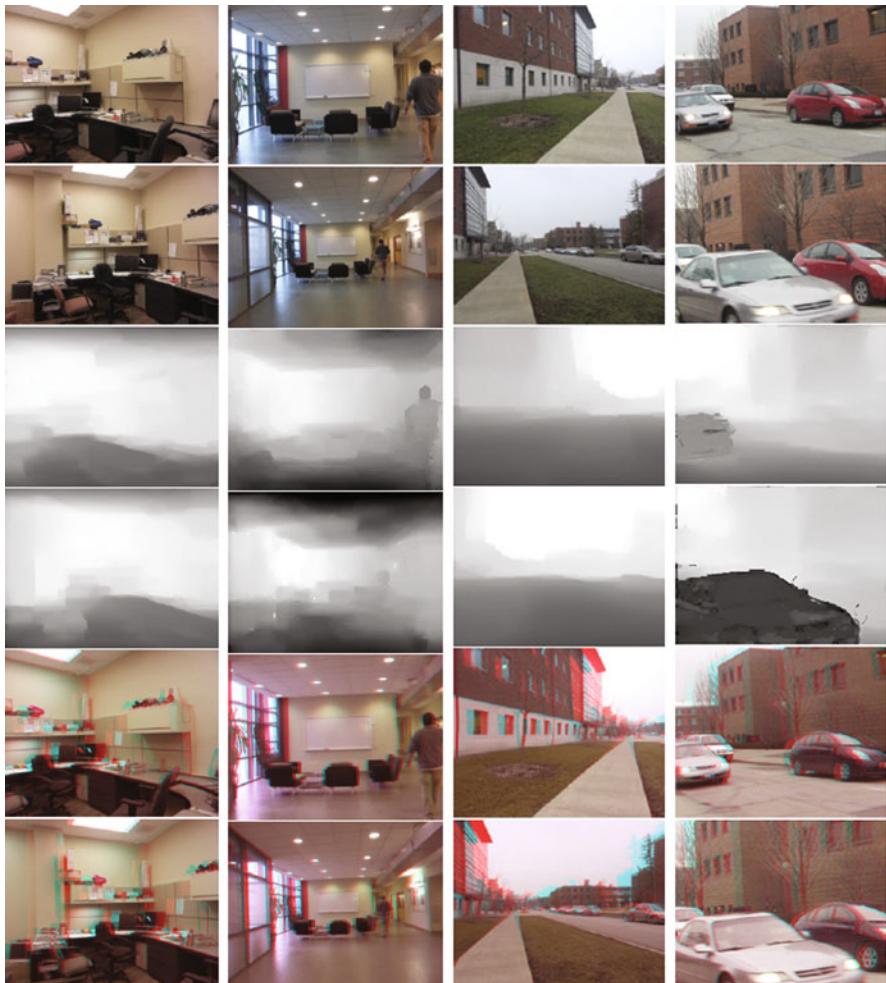


Fig. 10 Results obtained on four different sequences captured with a rotating camera and/or variable focal length. We show the input frames (*top rows*), inferred depth (*middle rows*) and inferred 3D anaglyph (*bottom rows*). Notice that the sequences are time-coherent and that moving objects are not ignored

and inferred depths to match the range of the Make3D database (roughly 1–81 m). As expected, the results from Building 1 are the best, but our method still achieves reasonable errors for the other buildings as well.

Figure 11 shows a result from each building in our dataset (top left is Building 1). As the quantitative results suggest, our algorithm performs very well for this building. In the remaining examples, we show results of videos captured in the other three buildings, all which contain vastly different colors, surfaces, and structures



Fig. 11 Video results obtained on test images for each building in our stereo-RGBD dataset (buildings 1–4, *from left to right and top to bottom*). For each result (*from left to right*): original photograph, ground truth depth, our inferred depth, ground truth anaglyph image, and our synthesized anaglyph image. Because the ground truth 3D images were recorded with a fixed interocular distance (roughly 5 cm), we cannot control the amount of “pop-out,” and the 3D effect is subtle. However, this is a parameter we can set using our automatic approach to achieve a desired effect, which allows for an enhanced 3D experience. Note also that our algorithm can handle multiple moving objects (*top*)

from the Building 1. Notice that even for these images our algorithm works well, as evidenced by the estimated depth and 3D images.

We demonstrate our method on videos containing parallax in Fig. 12. Such videos can be processed with structure from motion (SfM) and multiview stereo algorithms; e.g., the dense depth estimation method of Zhang et al. [40]. We visually compare

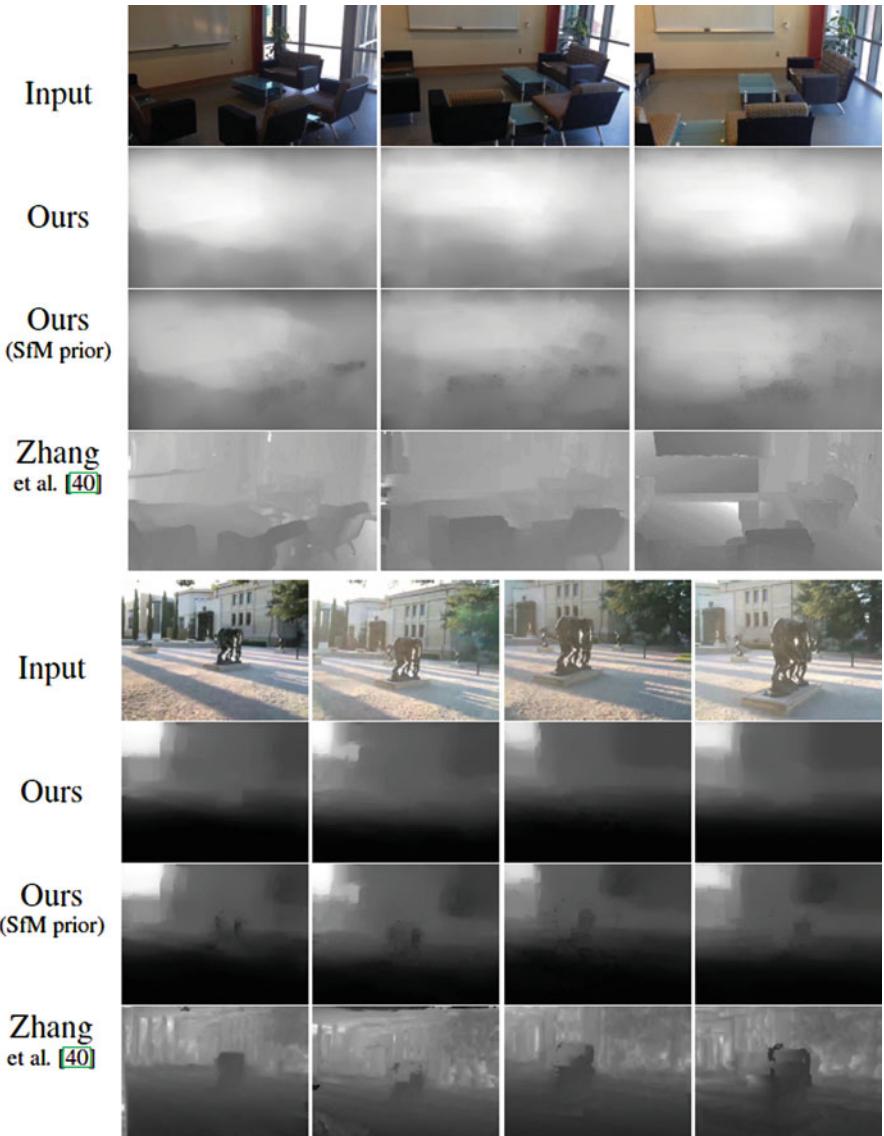


Fig. 12 Comparison of our method on videos containing parallax. For each input, we show the depth estimated using our method with no modification, and the depth estimated when our method is bootstrapped with SfM (e.g. a sparse depth map, calculated using SfM, is used as the prior). We also compare these results to the dense depth estimates of Zhang et al. [40], whose method works only for videos with parallax. When SfM estimates are poor (*left sequence*), multiview stereo methods may perform worse than our method, but can be quite good given decent SfM estimates and mostly Lambertian scenes (*right sequence*). Overall, SfM seems to help estimate relative depth near boundaries, whereas our method seems to better estimate global structure

Table 3 Error averaged over our stereo-RGBD dataset

Dataset	Rel	\log_{10}	textbfRMS	PSNR
Building 1 ^a	0.196	0.082	8.271	15.6
Building 2	0.394	0.135	11.7	15.6
Building 3	0.325	0.159	15.0	15.0
Building 4	0.251	0.136	15.3	16.4
Outdoors ^b	—	—	—	15.2
All	0.291	0.128	12.6	15.6

^aBuilding used for training (results for Building 1 trained using a hold-one-out scheme). ^bNo ground truth depth available

our method to the method of Zhang et al., as well as a version of our method where the depth prior comes from a sparse SfM point cloud. Specifically, we solve for camera pose, track, and triangulate features using publicly available code from Zhang et al. Then, we project the triangulated features into each view and compute their depth; this sparse depth map is used as the prior term in Eq. (6), and for videos with parallax, we increase the prior weight ($\beta = 10^3$) and turn off motion segmentation ($\eta = 0$). We note that in most cases of parallax, a multiview stereo algorithm is preferable to our solution, yet in some cases our method seems to perform better qualitatively.

As further evaluation, a qualitative comparison between our technique and the publicly available version of Make3D is shown in Fig. 13. Unlike Make3D, our technique is able to extract the depth of the runner throughout the sequence, in part because Make3D does not incorporate temporal information.

Our algorithm also does not require video training data to produce video results. We can make use of static RGBD images (e.g., Make3D dataset) to train our algorithm for video input, and we show several outdoor video results in Figs. 1, 4, 5, and 12. Even with static data from another location, our algorithm is usually able to infer accurate depth and stereo views.

Since we collected ground truth stereo images in our dataset, we also compare our synthesized right view (from our 2D-to-3D algorithm, see Sect. 7) with the actual right view. We use peak signal-to-noise ratio (PSNR) to measure the quality of the reconstructed views, as shown in Table 3. We could not acquire depth outdoors, and we use this metric to compare our outdoor and indoor results.

6.3 Importance of Training Scale

One implicit hypothesis our algorithm makes is that *training with only a few “similar” images is better than training with a large set of arbitrary images*. This is encoded by our nearest neighbor candidate search: we choose k similar



Fig. 13 Comparison between our technique and the publicly available version of Make3D (<http://make3d.cs.cornell.edu/code.html>). Make3D depth inference is trained to produce depths of resolution 55×305 (bilinearly resampled for visualization), and we show results of our algorithm at the input native resolution. The anaglyph images are produced using the technique in Sect. 7. Depths displayed at same scale

Table 4 Importance of training set size

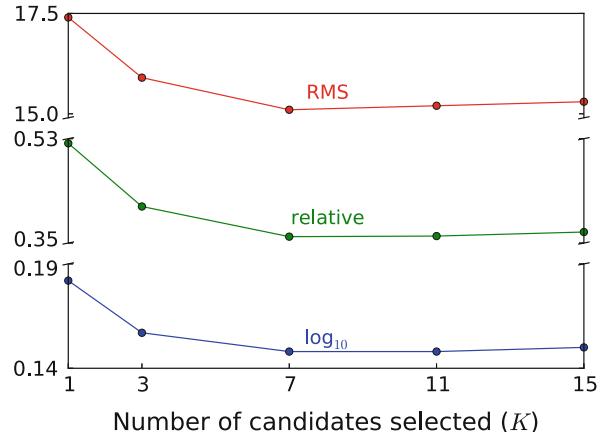
Training set	($N = 64$)	Rel	\log_{10}	RMS
GIST candidates	($K = 7$)	0.431	0.164	15.9
Full dataset	($K = 64$)	0.475	0.207	20.3

We select a subset of 64 images from the Make3D dataset, and compare our method using only seven candidate images (selected using GIST features) versus using the entire dataset (64 candidates). The results suggest that selectively retraining based on the target image's content can significantly improve results

images (in our work, based on GIST features), and use *only* these images to sample/transfer depth from. Conversely, Saxena et al. [31] train a parametric model for predicting depth using their entire dataset (Make3D); Liu et al. [20] found that training different models for each semantic classes (tree, sky, etc.) improves results. The results in Table 4 are good evidence that training on only similar *scenes* improves results (rather than only similar semantic classes as in [20], and on the entire dataset [31]).

We verify this further with another experiment: we created a sub-dataset by randomly choosing 64 images from the Make3D dataset, then we compared the results of our method using only similar images for training (seven nearest neighbors) and the results of our method trained using the entire dataset (64 nearest neighbors) during inference. As the results in Table 4 suggest, training using only

Fig. 14 Effect of the number of chosen candidates (K). Errors are reported on the Make3D dataset with varying values of K . For this dataset, $K = 7$ is optimal, but increasing K beyond 7 does not significantly degrade results



similar scenes greatly improves quantitative results. In addition, since our “training” is simply a nearest neighbor query (which can be done quickly in seconds), it can be orders of magnitude faster than retraining parametric models.

6.4 Effect of K (Number of Candidates)

We also evaluate how our technique behaves given different values of K , i.e., how many candidate images are selected prior to inference. On the Make3D dataset, we evaluate three error metrics (same as above: relative, \log_{10} , and RMS) averaged over the entire dataset using different values of K . Figure 14 shows the results, and we see that $K = 7$ is optimal for this dataset, but we still achieve comparable results with $K \geq 7$.

Empirically, we find that K acts as a smoothing parameter. This fits with the intuition that more candidate images will likely lead to diversity in the candidate set, and since the inferred depth is in some sense sampled from all candidates, the result will be smoother as K increases.

7 Application: 2D-to-3D

In recent years, 3D⁵ videos have become increasingly popular. Many feature films are now available in 3D, and increasingly more personal photography devices are now equipped with stereo capabilities (from point-and-shoots to attachments for

⁵The presentation of stereoscopic (left+right) video to convey the sense of depth.

video cameras and SLRs). Distributing user generated content is also becoming easier. YouTube has recently incorporated 3D viewing and uploading features, and many software packages have utilities for handling and viewing 3D file formats, e.g., Fujifilm’s FinePixViewer.

As 3D movies and 3D viewing technology become more widespread, it is desirable to have techniques that can convert legacy 2D movies to 3D in an efficient and inexpensive way. Currently, the movie industry uses expensive solutions that tend to be manual-intensive. For example, it was reported that the cost of converting (at most) 20 min of footage for the movie *Superman Returns* was \$10 million.⁶

Our technique can be used to automatically generate the depth maps necessary to produce the stereoscopic video (by warping each input frame using its corresponding depth map). To avoid generating holes at disocclusions in the view synthesis step, we adapt and extend Wang et al.’s technique [35]. They developed a method that takes as input a single image and per-pixel disparity values, and intelligently warps the input image based on the disparity such that highly salient regions remain unmodified. Their method was applied only to single images; we extend this method to handle video sequences as well.

7.1 Automatic Stereoscopic View Synthesis

After estimating depth for a video sequence (or a single image), we perform depth image-based rendering (DIBR) to synthesize a new view for stereoscopic display. A typical strategy for DIBR is simply reprojecting pixels based on depth values to a new, synthetic camera, but such methods are susceptible to large “holes” at disocclusions. Much work has been done to fill these holes (e.g., [2, 12, 23, 42]), but visual artifacts still remain in the case of general scenes.

We propose a novel extension to a recent DIBR technique which uses image warping to overcome problems such as disocclusions and hole filling. Wang et al. [35] developed a method that takes as input a single image and per-pixel disparity values, and intelligently warps the input image based on the disparity such that highly salient regions remain unmodified. This method is illustrated in Fig. 15. The idea is that people are less perceptive of errors in low saliency regions, and thus disocclusions are covered by “stretching” the input image where people tend not to notice artifacts. This method was only applied to single images, and we show how to extend this method to handle video sequences in the following text.

Given an input image and depth values, we first invert the depth to convert it to disparity, and scale the disparity by the maximum disparity value:

$$\mathbf{W}_0 = \frac{\mathbf{W}_{\max}}{\mathbf{D} + \epsilon}, \quad (13)$$

⁶See http://en.wikipedia.org/wiki/Superman_Returns.

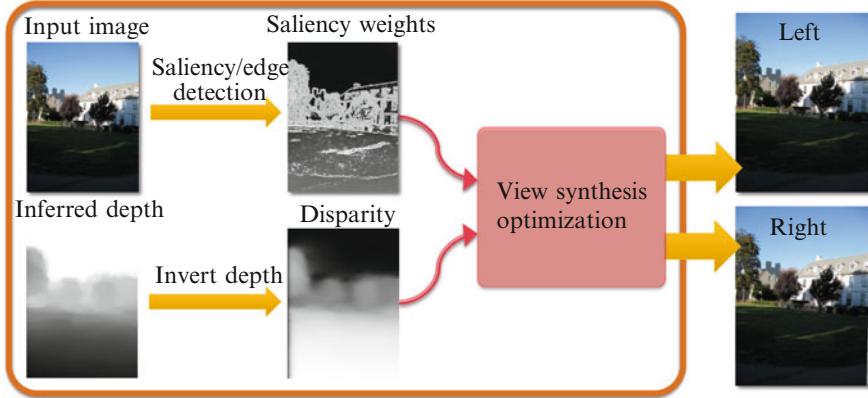


Fig. 15 Summary of the view synthesis procedure for a single image, as described by Wang et al. [35]. Given an image and corresponding depth, we compute salient regions and disparity, and compute stereoscopic images by warping the input image. We extend this method to handle videos, as in Eq. (14)

where $\mathbf{W}_0 = \{W_1, \dots, W_n\}$, $\mathbf{D} = \{D_1, \dots, D_n\}$ is the initial disparity and depth (resp) for each of the n frames of the input, and \mathbf{W}_{\max} is a parameter which modulates how much objects “pop-out” from the screen when viewed with a stereoscopic device. Increasing this value not only enhances the “3D” effect, but also causes eye strain or problems with fusing the stereo images if set too high. We set $\epsilon = 0.01$.

Then, to implement the saliency preserving warp (which in turn defines two newly synthesized views), minimize the following unconstrained, quadratic objective:

$$\begin{aligned} Q(\mathbf{W}_i) &= \sum_{i \in \text{pixels}} Q_{\text{data}}(\mathbf{W}_i) + Q_{\text{smooth}}(\mathbf{W}_i), \\ Q_{\text{data}}(\mathbf{W}_i) &= l_i (\mathbf{W}_i - \mathbf{W}_{0,i})^2, \\ Q_{\text{smooth}}(\mathbf{W}_i) &= \lambda (s_{x,i} \|\nabla_x \mathbf{W}_i\|^2 + s_{y,i} \|\nabla_y \mathbf{W}_i\|^2) \\ &\quad + \mu s_{t,i} \|\nabla_{\text{flow}} \mathbf{W}_i\|^2, \end{aligned} \quad (14)$$

where l_i is a weight based on image saliency and initial disparity values that constrains disparity values corresponding to highly salient regions and very close objects to remain unchanged, and is set to $l_i = \frac{W_{0,i}}{\mathbf{W}_{\max}} + (1 + e^{-(\|\nabla \mathbf{L}_i\| - 0.01)/0.002})^{-1}$. The Q_{smooth} term contains the same terms as in our spatial and temporal smoothness functions in our depth optimization’s objective function, and λ and μ control the weighting of these smoothness terms in the optimization; we set $\lambda = \mu = 10$. As in Eq. (4), we set $s_{x,i} = (1 + e^{(\|\nabla_x \mathbf{L}_i\| - 0.05)/0.01})^{-1}$, $s_{y,i} = (1 + e^{(\|\nabla_y \mathbf{L}_i\| - \mu_L)/\sigma_L})^{-1}$, and $s_{t,i} = (1 + e^{-(\|\nabla_{\text{flow}} \mathbf{L}_i\| - \mu_L)/\sigma_L})^{-1}$, where $\nabla_x \mathbf{L}_i$ and $\nabla_y \mathbf{L}_i$ are image gradients

in the respective dimensions (at pixel i), and $\nabla_{\text{flow}} \mathbf{L}_i$ is the flow difference across neighboring frames (gradient in the flow direction). We set $\mu_L = 0.05$ and $\sigma_L = 0.01$. With this formulation, we ensure spatial and temporal coherence and most importantly that highly salient regions remain intact during view warping.

After optimization, we divide the disparities by two ($\mathbf{W} \leftarrow \frac{\mathbf{W}}{2}$), and use these halved values to render the input frame(s) into two new views (corresponding to the stereo left and right views). We choose this method, as opposed to only rendering one new frame with larger disparities, because people are less perceptive of many small artifacts when compared with few large artifacts [35]. For rendering, we use the anisotropic pixel splatting method described by Wang et al. [35], which “splats” input pixels into the new view (based on \mathbf{W}) as weighted, anisotropic Gaussian blobs.

With the two synthesized views, we can convert to any 3D viewing format, such as anaglyph or interlaced stereo. For the results in this paper, we use the anaglyph format as cyan/red anaglyph glasses are more widespread than polarized/autostereoscopic displays (used with interlaced 3D images). To reduce eye strain, we shift the left and the right images such that the nearest object has zero disparity, making the nearest object appear at the display surface, and all other objects appear *behind* the display. This is commonly known as the “window” metaphor [15].

7.2 2D-to-3D Results

Our estimated depth is good enough to generate compelling 3D images, and representative results are shown in Figs. 10 and 11. We also demonstrate that our algorithm may be suitable for feature films in Fig. 16. More diverse quantities of



Fig. 16 Several clips from the feature film *Charade*. Each result contains (*from top to bottom*): the original frames, estimated depth, and estimated anaglyph (red/cyan) automatically generated by our algorithm. Some imperfections in depth are conveniently masked in the 3D image due to textureless or less salient regions

training are required to achieve commercial quality conversion; however, even with a small amount of data, we can generate plausible depth maps and create convincing 3D sequences automatically.

Recently, YouTube has released an automatic 2D-to-3D conversion tool, and we compared our method to theirs on several test sequences. Empirically, we noticed that the YouTube results have a much more subtle 3D effect. Both results are available online at <http://kevinkarsch.com/depthtransfer>.

Our algorithm takes roughly 1 min per 640×480 frame (on average) using a parallel implementation on a quad-core 3.2 GHz processor.

8 Discussion

Our results show that our depth transfer algorithm works for a large variety of indoor and outdoor sequences using a practical amount of training data. Note that our algorithm works for arbitrary videos, not just those with no parallax. However, videos with arbitrary camera motion and static scenes are best handled with techniques such as [41]. In Fig. 17, we show that our algorithm requires some similar data in order to produce decent results (i.e., training with outdoor images for an indoor query is likely to fail). However, our algorithm can robustly handle large

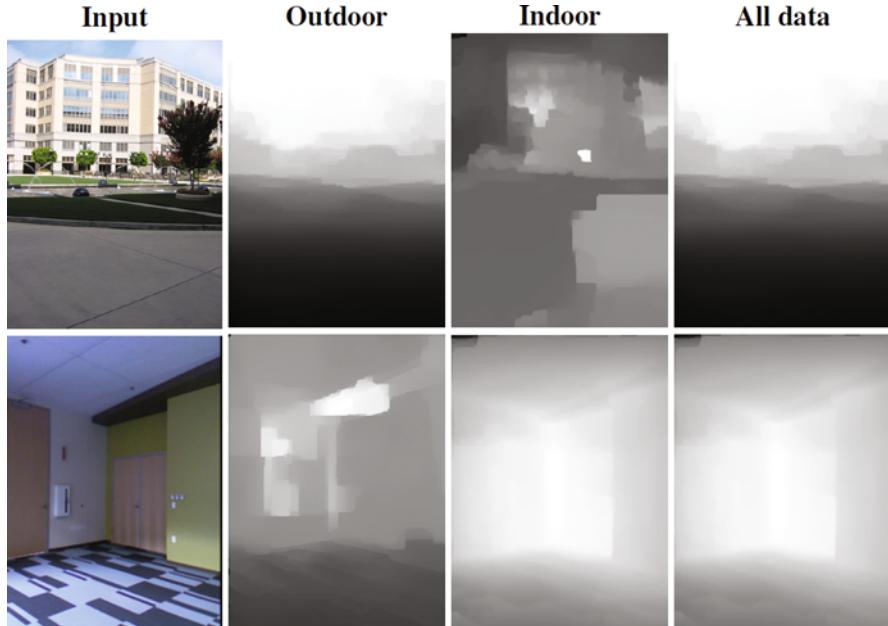


Fig. 17 Effect of using different training data for indoor and outdoor images. While the results are best if the proper dataset is used, we also get good results even if we combine all of the datasets

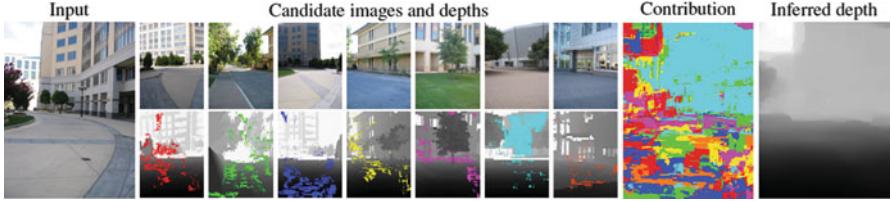


Fig. 18 Candidate contribution for depth estimation. For an input image (*left*), we find top-matching candidate RGBD images (*middle*), and infer depth (*right*) for the input using our technique. The contribution image is color coded to show the sources; *red pixels* indicate that the left-most candidate influenced the inferred depth image the most, *orange indicates* contribution from the right-most candidate, etc.



Fig. 19 Example failure cases. *Left:* thin or floating objects (pole and basketball) are ignored. *Right:* input image is too different from training set

amounts of depth data with little degradation of output quality. The only issue is that more data requires more comparisons in the candidate search.

This robustness is likely due to the features we use when determining candidate images as well as the design of our objective function. In Fig. 18, we show an example query image, the candidates retrieved by our algorithm, and their contribution to the inferred depth. By matching GIST features, we detect candidates that contain features consistent with the query image, such as building facades, sky, shrubbery, and similar horizon location. Notice that the depth of the building facade in the input comes mostly from another similarly oriented building facade (teal), and the ground plane and shrubbery depth come almost solely from other candidates' ground and tree depths.

In some cases, our motion segmentation misses or falsely identifies moving pixels. This can result in inaccurate depth and 3D estimation, although our spatio-temporal regularization [Eqs. (5), (10)] helps to overcome this. Our algorithm also assumes that moving objects contact the ground, and thus may fail for airborne objects (see Fig. 19).

Due to the serial nature of our method (depth estimation followed by view synthesis), our method is prone to propagating errors through the stages. For example, if an error is made during depth estimation, the result may be visually implausible. It would be ideal to use knowledge of how the synthesized views should look in order to correct issues in depth.

9 Concluding Remarks

We have demonstrated a fully automatic technique to estimate depths for videos. Our method is applicable in cases where other methods fail, such as those based on motion parallax and structure from motion, and works even for single images and dynamics scenes. Our depth estimation technique is novel in that we use a nonparametric approach, which gives qualitatively good results, and our single image algorithm quantitatively outperforms existing methods. Using our technique, we also show how we can generate stereoscopic videos for 3D viewing from conventional 2D videos. Specifically, we show how to generate time coherent, visually pleasing stereo sequences using our inferred depth maps. Our method is suitable as a good starting point for converting legacy 2D feature films into 3D.

Acknowledgements We would like to thank Tom Blank for his critical help in creating our dual-Kinect data collection system.

References

1. Batra, D., Saxena, A.: Learning the right model: efficient max-margin learning in laplacian crfs. In: CVPR (2012)
2. Colombari, A., Fusiello, A., Murino, V.: Continuous parallax adjustment for 3D-TV. In: IEEE Eur. Conf. Vis. Media Prod, pp. 194–200 (2005)
3. Delage, E., Lee, H., Ng, A.: A dynamic Bayesian network model for autonomous 3D reconstruction from a single indoor image. In: CVPR (2006)
4. Guttmann, M., Wolf, L., Cohen-Or, D.: Semi-automatic stereo extraction from video footage. In: ICCV (2009)
5. Han, F., Zhu, S.C.: Bayesian reconstruction of 3D shapes and scenes from a single image. In: IEEE HLK (2003)
6. Hassner, T., Basri, R.: Example based 3D reconstruction from single 2D images. In: CVPR Workshop on Beyond Patches, pp. 15–22 (2006)
7. He, K., Sun, J., Tang, X.: Single image haze removal using dark channel prior. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(12), 2341–2353 (2011). doi:[10.1109/TPAMI.2010.168](https://doi.org/10.1109/TPAMI.2010.168)
8. Heikkilä, M., Pietikainen, M.: A texture-based method for modeling the background and detecting moving objects. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(4), 657–662 (2006)
9. Hoiem, D., Efros, A., Hebert, M.: Automatic photo pop-up. In: ACM SIGGRAPH (2005)
10. Hoiem, D., Stein, A., Efros, A., Hebert, M.: Recovering occlusion boundaries from a single image. In: ICCV (2007)
11. Horry, Y., Anjyo, K., Arai, K.: Tour into the picture: using a spidery mesh interface to make animation from a single image. In: SIGGRAPH (1997)
12. Klein Gunnewiek, R., Berretty, R.P., Barenbrug, B., Magalhães, J.: Coherent spatial and temporal occlusion generation. In: Proc. SPIE 7237, Stereoscopic Displays and Applications XX, vol. 723713 (2009)
13. Konrad, J., Brown, G., Wang, M., Ishwar, P., Wu, C., Mukherjee, D.: Automatic 2d-to-3d image conversion using 3d examples from the internet. In: SPIE 8288, Stereoscopic Displays and Applications, vol. 82880F (2012). doi:[10.1117/12.766566](https://doi.org/10.1117/12.766566)

14. Konrad, J., Wang, M., Ishwar, P.: 2d-to-3d image conversion by learning depth from examples. In: 3DCINE (2012)
15. Koppal, S., Zitnick, C., Cohen, M., Kang, S., Ressler, B., Colburn, A.: A viewer-centric editor for 3D movies. *IEEE Comput. Graph. Appl.* **31**, 20–35 (2011)
16. Li, C., Kowdle, A., Saxena, A., Chen, T.: Towards holistic scene understanding: feedback enabled cascaded classification models. In: NIPS (2010)
17. Liao, M., Gao, J., Yang, R., Gong, M.: Video stereolization: combining motion analysis with user interaction. *IEEE Trans. Vis. Comput. Graph.* **18**(7), 1079–1088 (2012)
18. Liu, C.: Beyond pixels: exploring new representations and applications for motion analysis. Ph.D. thesis, MIT (2009)
19. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing: label transfer via dense scene alignment. In: CVPR (2009)
20. Liu, B., Gould, S., Koller, D.: Single image depth estimation from predicted semantic labels. In: CVPR (2010)
21. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing via label transfer. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(12), 2368–2382 (2011)
22. Liu, C., Yuen, J., Torralba, A.: SIFT flow: dense correspondence across scenes and its applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(5), 978–994 (2011)
23. Luo, K., Li, D., Feng, Y., M., Z.: Depth-aided inpainting for disocclusion restoration of multi-view images using depth-image-based rendering. *J. Zhejiang Univ. Sci. A* **10**(12), 1738–1749 (2009)
24. Maire, M., Arbelaez, P., Fowlkes, C., Malik, J.: Using contours to detect and localize junctions in natural images. In: CVPR (2008)
25. Nathan Silberman Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV (2012)
26. Oh, B., Chen, M., Dorsey, J., Durand, F.: Image-based modeling and photo editing. In: SIGGRAPH (2001)
27. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int. J. Comput. Vis.* **42**, 145–175 (2001)
28. Rubinstein, M., Liu, C., Freeman, W.: Annotation propagation: automatic annotation of large image databases via dense image correspondence. In: ECCV (2012)
29. Rubinstein, M., Joulin, A., Kopf, J., Liu, C.: Unsupervised joint object discovery and segmentation in internet images. In: CVPR (2013)
30. Saxena, A., Chung, S.H., Ng, A.Y.: Learning depth from single monocular images. In: Advances in Neural Information Processing Systems 18 (2005). http://books.nips.cc/papers/files/nips18/NIPS2005_0684.pdf
31. Saxena, A., Sun, M., Ng, A.: Make3D: learning 3D scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(5), 824–840 (2009)
32. Sheikh, Y., Javed, O., Kanade, T.: Background subtraction for freely moving cameras. In: ICCV (2009)
33. Tappen, M., Liu, C.: A bayesian approach to alignment-based image hallucination. In: ECCV (2012)
34. Van Pernis, A., DeJohn, M.: Dimensionalization: converting 2D films to 3D. In: SPIE 6803, Stereoscopic Displays and Applications XIX, vol. 68030T (2008). doi:[10.1117/12.766566](https://doi.org/10.1117/12.766566)
35. Wang, O., Lang, M., Frei, M., Hornung, A., Smolic, A., Gross, M.: StereoBrush: interactive 2D to 3D conversion using discontinuous warps. In: SBIM (2011)
36. Ward, B., Kang, S.B., Bennett, E.P.: Depth director: a system for adding depth to movies. *IEEE Comput. Graph. Appl.* **31**(1), 36–48 (2011)
37. Wu, C., Frahm, J.M., Pollefeys, M.: Repetition-based dense single-view reconstruction. In: CVPR (2011)
38. Zhang, L., Dugas-Phocion, G., Samson, J.S., Seitz, S.: Single view modeling of free-form scenes. *J. Vis. Comput. Animat.* **13**(4), 225–235 (2002)
39. Zhang, G., Dong, Z., Jia, J., Wan, L., Wong, T.T., Bao, H.: Refilming with depth-inferred videos. *IEEE Trans. Vis. Comput. Graph.* **15**(5), 828–840 (2009)

40. Zhang, G., Jia, J., Wong, T.T., Bao, H.: Consistent depth maps recovery from a video sequence. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**, 974–988 (2009)
41. Zhang, G., Jia, J., Hua, W., Bao, H.: Robust bilayer segmentation and motion/depth estimation with a handheld camera. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(3), 603–617 (2011)
42. Zhang, L., Vazquez, C., Knorr, S.: 3D-TV content creation: automatic 2D-to-3D video conversion. *IEEE Trans. Broadcast.* **57**(2), 372–383 (2011)

Nonparametric Scene Parsing via Label Transfer

Ce Liu, Jenny Yuen, and Antonio Torralba

Abstract While there has been a lot of recent work on object recognition and image understanding, the focus has been on carefully establishing mathematical models for images, scenes, and objects. In this chapter, we propose a novel, nonparametric approach for object recognition and scene parsing using a new technology we name *label transfer*. For an input image, our system first retrieves its nearest neighbors from a large database containing fully annotated images. Then, the system establishes dense correspondences between the input image and each of the nearest neighbors using the dense SIFT flow algorithm (Liu et al., 33(5):978–994, 2011 Chap. 2), which aligns two images based on local image structures. Finally, based on the dense scene correspondences obtained from the SIFT flow, our system warps the existing annotations, and integrates multiple cues in a Markov random field framework to segment and recognize the query image. Promising experimental results have been achieved by our nonparametric scene parsing system on challenging databases. Compared to existing object recognition approaches that require training classifiers or appearance models for each object category, our system is easy to implement, has few parameters, and embeds contextual information naturally in the retrieval/alignment procedure.

1 Introduction

Scene parsing, or recognizing and segmenting objects in an image, is one of the core problems of computer vision. Traditional approaches to object recognition begin by specifying an object model, such as template matching [8, 48], constellations

C. Liu (✉)
Google Research, Cambridge, MA, USA
e-mail: celiu@google.com

J. Yuen
Facebook, Inc., Menlo Park, CA, USA
e-mail: jenny@csail.mit.edu

A. Torralba
Massachusetts Institute of Technology, Cambridge, MA, USA
e-mail: torralba@csail.mit.edu

[14, 15], bags of features [18, 23, 43, 44], or shape models [2, 3, 13], etc. These approaches typically work with a fixed number of object categories and require training generative or discriminative models for each category given training data. In the parsing stage, these systems try to align the learned models to the input image and associate object category labels with pixels, windows, edges, or other image representations. Recently, context information has also been carefully modeled to capture the relationship between objects at the semantic level [19, 21]. Encouraging progress has been made by these models on a variety of object recognition and scene parsing tasks.

However, these learning-based methods do not, in general, scale well with the number of object categories. For example, to include more object categories in an existing system, we need to train new models for the new categories and, typically adjust system parameters. Training can be a tedious job if we want to include thousands of object categories in a scene parsing system. In addition, the complexity of contextual relationships among objects also increases rapidly as the quantity of object categories expands.

Recently, the emergence of large databases of images has opened the door to a new family of methods in computer vision. Large database-driven approaches have shown the potential for nonparametric methods in several applications. Instead of training sophisticated parametric models, these methods try to reduce the inference problem for an unknown image to that of matching to an existing set of annotated images. In [40], the authors estimate the pose of a human relying on 0.5 million training examples. In [20], the proposed algorithm can fill holes on an input image by introducing elements that are likely to be semantically correct through searching a large image database. In [36], a system is designed to infer the possible object categories that may appear in an image by retrieving similar images in a large database [37]. Moreover, the authors in [46] showed that with a database of 80 million images, even simple SSD match can give semantically meaningful parsing for 32×32 images.

In this chapter, we propose a novel, nonparametric scene parsing system to transfer the labels from existing samples in a large database to annotate an image, as illustrated in Fig. 1. For a query image (a), our system first retrieves the top matches in a large, annotated image database using a combination of GIST matching [33] and SIFT flow [26]. Since these top matches are labeled, we transfer the annotation (c) of the top matches to the query image and obtain the scene parsing result in (d). For comparison, the ground truth user annotation of the query is displayed in (e). Our system is able to generate promising scene parsing results if images from the same scene type as the query are retrieved in the annotated database.

However, it is nontrivial to build an efficient and reliable scene parsing system using dense scene alignment. To account for the multiple annotation suggestions from the top matches, a Markov random field (MRF) model is used to merge multiple cues (e.g. likelihood, prior and spatial smoothness) into a robust annotation. Promising experimental results are achieved on images from the LabelMe database [37].

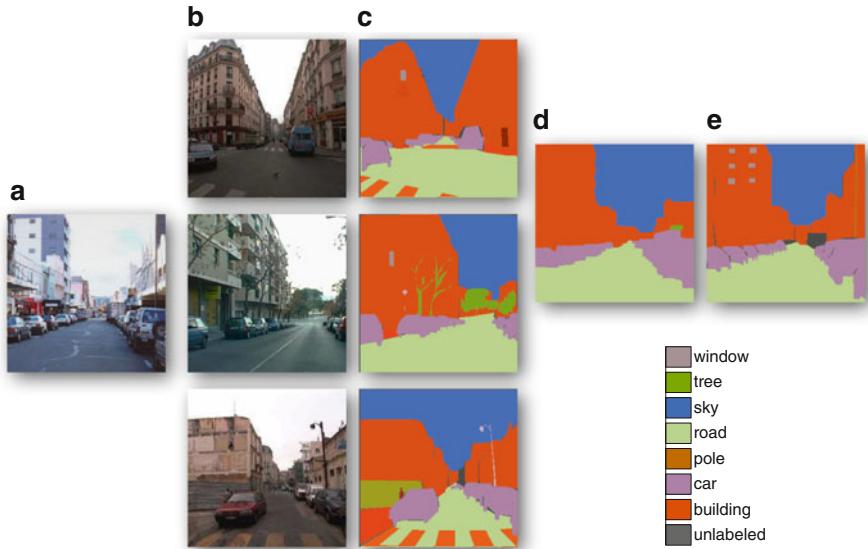


Fig. 1 For a query image (a), our system finds the top matches (b) (three are shown here) using scene retrieval and SIFT flow matching algorithm [26, 28]. The annotations of the top matches (c) are transferred and integrated to parse the input image as shown in (d). For comparison, the ground truth user annotation of (a) is shown in (e)

Our goal is to explore the performance of scene parsing through the transfer of labels from existing annotated images, rather than building a comprehensive object recognition system. We show, however, that the performance of our system outperforms existing approaches [8, 42] on our databases. Our code and databases can be downloaded at <http://people.csail.mit.edu/celiu/LabelTransfer/>.

This chapter is organized as follows. In Sect. 2 we briefly survey the object recognition and detection literature. After giving a system overview in Sect. 3, we describe in details each component of our system in Sect. 4. Thorough experiments are conducted in Sect. 5 for evaluation, and in-depth discussion is provided in Sect. 6. We conclude our chapter in Sect. 7.

2 Related Work

Object recognition is an area of research that has greatly evolved over the last decade. Many works focusing on single-class modeling such as faces [11, 47, 48], digits, characters, and pedestrians [2, 8, 24] have been proven successful and in some cases the problems have been mostly deemed as solved. Recent efforts have turned to mainly focus in the area of multi-class object recognition. In creating an object detection system, there are many basic building blocks to take into account;

feature description and extraction is the first stepping stone. Examples of descriptors include gradient-based features such as SIFT [29] and HOG[8], shape context [2] and patch statistics [41] among many. Consequently, selected feature descriptors can further be applied to images in either a sparse [2, 15, 18] manner by selecting the top key points containing the highest response from the feature descriptor, or densely by observing feature statistics across the image [39, 50].

Sparse key point representations are often matched among pairs of images. Since the generic problem of matching two sets of key points is NP-hard, approximation algorithms have been developed to efficiently compute key point matches minimizing error rates (e.g., the pyramid match kernel [18], and vocabulary trees [31, 32]). On the other hand, dense representations have been handled by modeling distributions of the visual features over neighborhoods in the image or in the image as a whole [23, 39, 50].

At a higher level, we can also distinguish two types of object recognition approaches: *parametric* approaches that consist of learning generative/discriminative models, and *nonparametric* approaches that rely on image retrieval and matching. In the parametric family we can find numerous template-matching methods, where classifiers are trained to discriminate between an image window containing an object or a background [8]. However, these methods assume that objects are mostly rigid and are susceptible to little or no deformation. To account for articulated objects, constellation models have been designed to model objects as ensembles of parts [13–15, 49], consider spatial information [7], depth ordering information [52], and multi-resolution modes [34]. Recently, a new idea of integrating humans in the loop via crowd sourcing for visual recognition of specialized classes such as plants and animal species has emerged [5]; this method integrates the description of an object in less than 20 discriminative questions that humans can answer after visually inspecting the image.

In the realm of nonparametric methods we find systems such as Video Google [43], a system that allows users to specify a visual query of an object in a video and subsequently retrieves instances of the same object across the movie. Another non-parametric system is the one in [36], where a previously unknown query image is matched against a densely labeled image database; the nearest neighbors are used to build a label probability map for the query, which is further used to prune out object detectors of classes that are unlikely to take place in the image. Nonparametric methods have also been widely used in web data to retrieve similar images. For example, in [16], a customized distance function is used at a retrieval stage to compute the distance between a query image and images in the training set, which subsequently cast votes to infer the object class of the query.

Recently, several works have also considered contextual information in object detections to clean and reinforce individual results. Among contextual cues that have been used are object-level co-occurrences, spatial relationships [6, 9, 17, 30, 35], and 3D scene layout [22]. For a more detailed and comprehensive study and benchmark of contextual works, we refer to [10].

An earlier version of our work appeared at [27]; in this chapter, we will explore the label-transfer framework in-depth with more thorough experiments and insights.

Other recent papers have also introduced similar ideas. For instance, in [45], over-segmentation is performed to the query image and segment-based classifiers trained on the nearest neighbors are applied to recognize each segment. In [38], scene boundaries are discovered by the common edges shared by nearest neighbors.

3 System Overview

The core idea of our nonparametric scene parsing system is recognition-by-matching. To parse an input image, we match the visual objects in the input image to the images in a database. If images in the database are annotated with object category labels, and if the matching is semantically meaningful, i.e. *building* corresponds to *building*, *window* to *window*, *person* to *person*, then we can simply transfer the labels of the images in the database to parse the input. Nevertheless, we need to deal with many practical issues in order to build a reliable system.

Figure 2 shows the pipeline of our system, which consists of the following three algorithmic modules:

- **Scene retrieval:** given a query image, use *scene retrieval* techniques to find a set of *nearest neighbors* that share similar scene configuration (including objects and their relationships) with the query.
- **Dense scene alignment:** establish *dense scene correspondence* between the query image and each of the retrieved nearest neighbors. Choose the nearest neighbors with the top matching scores as *voting candidates*.
- **Label transfer:** warp the annotations from the voting candidates to the query image according to estimated dense correspondence. Reconcile multiple labeling and impose spatial smoothness under a MRF model.

Although we are going to choose concrete algorithms for each module in this chapter, any algorithm that fits to the module can be plugged into our nonparametric scene parsing system. For example, we use SIFT flow for dense scene alignment, but it would also suffice to use sparse feature matching and then propagate sparse correspondences to produce dense counterparts.

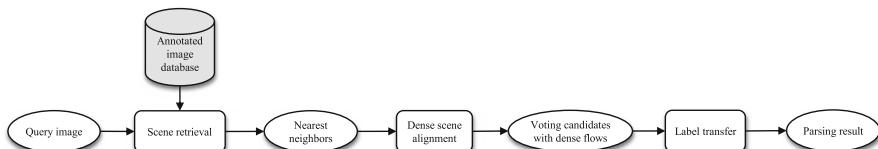


Fig. 2 System pipeline. There are three key algorithmic components (rectangles) in our system: *scene retrieval*, *dense scene alignment*, and *label transfer*. The ovals denote data representations

A key component of our system is a large, dense, and annotated image database.¹ In this chapter, we use two sets of databases, both annotated using the LabelMe online annotation tool [37], to build and evaluate our system. The first is the LabelMe Outdoor (LMO) database [27], containing 2688 fully annotated images, most of which are outdoor scenes including street, beach, mountains, fields, and buildings. The second is the SUN database [51], containing 9566 fully annotated images, covering both indoor and outdoor scenes; in fact, LMO is a subset of SUN. We use the LMO database to explore our system in-depth, and also report the results on the SUN database.

Before jumping into the details of our system, it is helpful to look at the statistics of the LMO database. The 2688 images in LMO are randomly split to 2488 for training and 200 for testing. We chose the top 33 object categories with the most labeled pixels. The pixels that are not labeled, or labeled as other object categories, are treated as the 34th category: “unlabeled.” The per pixel frequency count of these object categories in the training set is shown at the top of Fig. 3. The color of each bar is the average RGB value of the corresponding object category from the training data with saturation and brightness boosted for visualization purposes. The top 10 object categories are *sky*, *building*, *mountain*, *tree*, *unlabeled*, *road*, *sea*, *field*, *grass*, and *river*. The spatial priors of these object categories are displayed at the bottom of Fig. 3, where white denotes zero probability and the saturation of color is directly proportional to its probability. Note that consistent with common knowledge, *sky* occupies the upper part of image grid and *field* occupies the lower part. Furthermore, there are only limited samples for the *sun*, *cow*, *bird*, and *moon* classes.

4 System Design

In this section we will describe each module of our nonparametric scene parsing system.

4.1 Scene Retrieval

The objective of scene retrieval is to retrieve a set of nearest neighbors in the database for a given query image. There exist several ways for defining a nearest neighbor set. The most common definition consists of taking the K closest points to the query (K -NN). Another model, ϵ -NN, widely used in texture synthesis [12, 25], considers all of the neighbors within a distance denoted by $1 + \epsilon$ from the query. We generalize these two types to (K, ϵ) -NN, and define it as

¹Other scene parsing and image understanding systems also require such a database. We do not require more than others.

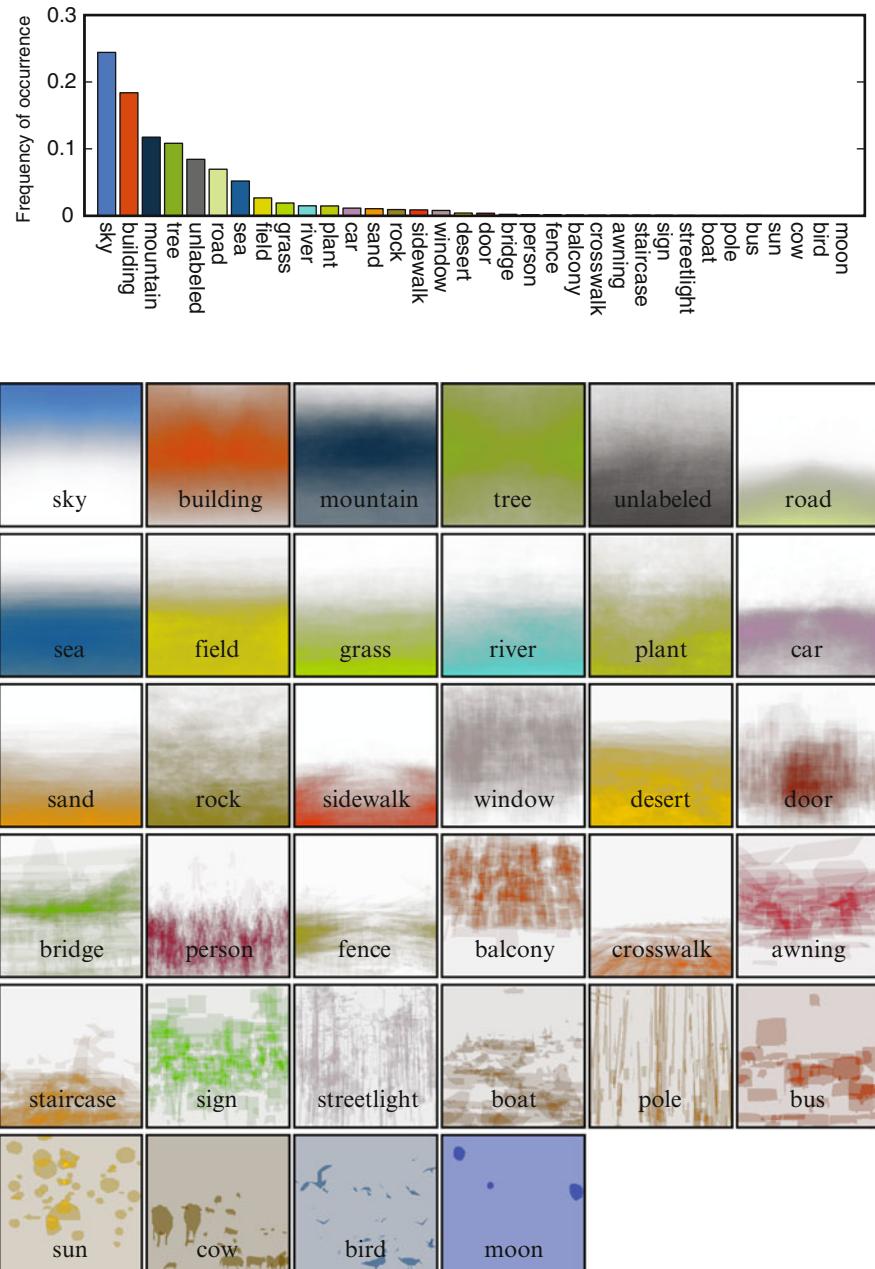


Fig. 3 *Above:* the per-pixel frequency counts of the object categories in our dataset (sorted in descending order). The color of each bar is the average RGB value of each object category from the training data with saturation and brightness boosted for visualization. *Bottom:* the spatial priors of the object categories in the database. White means zero and the *saturated color* means high probability

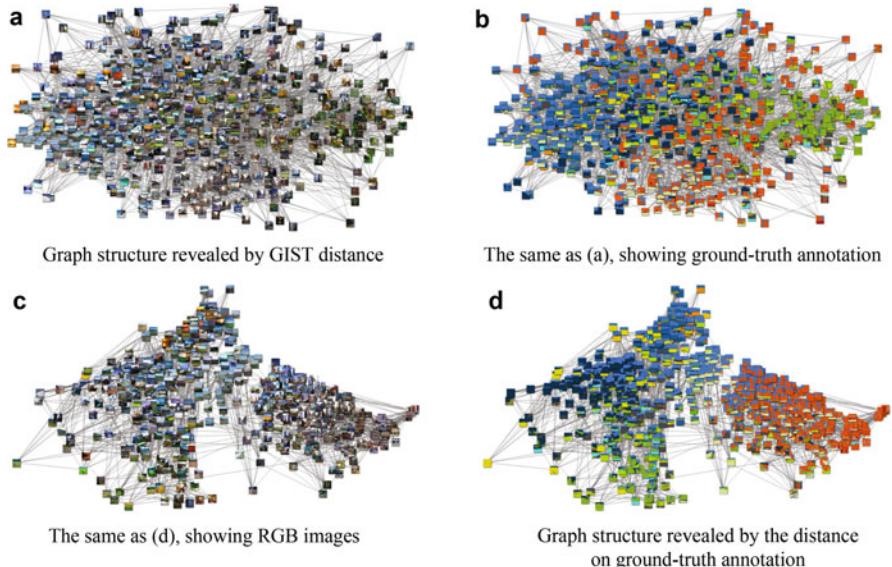


Fig. 4 The structure of a database depends on image distance metric. *Top*: the $\langle K, \epsilon \rangle$ -NN graph of the LabelMe Outdoor (LMO) database visualized by scaled MDS using GIST feature as distance. *Bottom*: the $\langle K, \epsilon \rangle$ -NN graph of the same database visualized using the pyramid histogram intersection of ground truth annotation as distance. *Left*: RGB images; *right*: annotation images. Notice how the ground truth annotation emphasizes the underlying structure of the database. In (c) and (d), we see that the image content changes from urban, streets (*right*), to highways (*middle*), and to nature scenes (*left*) as we pan from right to left. Eight hundred images are randomly selected from LMO for this visualization

$$\begin{aligned} \mathcal{N}(x) &= \{y_i \mid \text{dist}(x, y_i) \leqslant (1 + \epsilon)\text{dist}(x, y_1), \\ y_1 &= \arg \min \text{dist}(x, y_i), i \leqslant K\}. \end{aligned} \quad (1)$$

As $\epsilon \rightarrow \infty$, $\langle K, \infty \rangle$ -NN is reduced to K -NN. As $K \rightarrow \infty$, $\langle \infty, \epsilon \rangle$ -NN is reduced to ϵ -NN. However, $\langle K, \epsilon \rangle$ -NN representation gives us the flexibility to deal with the density variation of the graph, as shown in Fig. 5. We will show how K affects the performance in the experimental section. In practice, we found that $\epsilon = 5$ is a good parameter and we will use it through our experiments. Nevertheless, dramatic improvement of $\langle K, \epsilon \rangle$ -NN over K -NN is not expected as sparse samples are few in our databases.

We haven not yet defined the distance function $\text{dist}(\cdot, \cdot)$ between two images. Measuring image similarities/distances is still an active research area; a systematic study of image features for scene recognition can be found in [51]. In this chapter, three distances are used: Euclidean distance of GIST [33], spatial pyramid histogram intersection of HOG visual words [23], and spatial pyramid histogram intersection of the ground truth annotation. For the HOG distance, we use the standard pipeline of computing HOG features on a dense grid and quantizing features to visual words

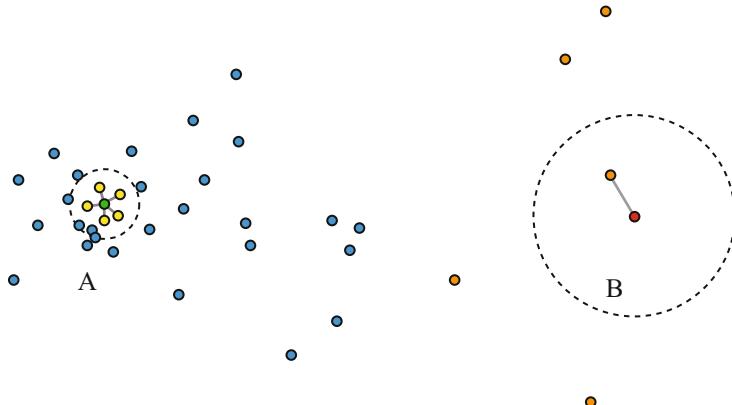


Fig. 5 An image database can be nonuniform as illustrated a some random 2D points. The *green node* (*A*) is surrounded by dense neighbors, whereas the *red node* (*B*) resides in a sparse area. If we use K -NN ($K = 5$), then some samples (*orange nodes*) far away from the query (*B*) can be chosen as neighbors. If, instead, we use ϵ -NN and choose the radius as shown in the picture, then there can be too many neighbors for a sample such as (*A*). The combination, $\langle K, \epsilon \rangle$ -NN, shown as *gray-edges*, provides a good balance for these two criteria

over a set of images using k-means clustering. The ground truth-based distance metric is used to estimate an upper bound of our system for evaluation purposes. Both the HOG and the ground truth distances are computed in the same manner. The ground truth distance is computed by building histograms of pixel-wise labels. To include spatial information, the histograms are computed by dividing an image into 2×2 windows and concatenating the four histograms into a single vector. Histogram intersection is used to compute the ground truth distance. We obtain the HOG distance by replacing pixel-wise labels with HOG visual words.

In Fig. 4, we show the importance of the distance metric as it defines the neighborhood structure of the large image database. We randomly selected 200 images from the LMO database and computed pair-wise image distances using GIST (top) and the ground truth annotation (bottom). Then, we use multidimensional scaling (MDS) [4] to map these images to points on a 2D grid for visualization. Although the GIST descriptor is able to form a reasonably meaningful image space where semantically similar images are clustered, the image space defined by the ground truth annotation truly reveals the underlying structures of the image database. This will be further examined in the experimental section.

4.2 SIFT Flow for Dense Scene Alignment

As our goal is to transfer the labels of existing samples to parse an input image, it is essential to find the dense correspondence for images across scenes. In our previous work [26], we have demonstrated that SIFT flow is capable of establishing

semantically meaningful correspondences among two images by matching local SIFT descriptors. We further extended SIFT flow into a hierarchical computational framework to improve the performance [27]. In this section, we will provide a brief explanation of the algorithm; for a detailed description, we refer to [28].

Similar to optical flow, the task of SIFT flow is to find dense correspondence between two images. Let $\mathbf{p} = (x, y)$ contain the spatial coordinate of a pixel, and $\mathbf{w}(\mathbf{p}) = (u(\mathbf{p}), v(\mathbf{p}))$ be the flow vector at \mathbf{p} . Denote s_1 and s_2 as the per-pixel SIFT descriptor [29] for two images,² and ε contains all the spatial neighborhood (a four-neighbor system is used). The energy function for SIFT flow is defined as:

$$E(\mathbf{w}) = \sum_{\mathbf{p}} \min \left(\|s_1(\mathbf{p}) - s_2(\mathbf{p} + \mathbf{w}(\mathbf{p}))\|_1, t \right) \quad (2)$$

$$+ \sum_{\mathbf{p}} \eta(|u(\mathbf{p})| + |v(\mathbf{p})|) \quad (3)$$

$$+ \sum_{(\mathbf{p}, \mathbf{q}) \in \varepsilon} \min \left(\lambda |u(\mathbf{p}) - u(\mathbf{q})|, d \right) + \min \left(\lambda |v(\mathbf{p}) - v(\mathbf{q})|, d \right), \quad (4)$$

which contains a *data term*, *small displacement term* and *smoothness term* (*a.k.a.* spatial regularization). The *data term* in Eq. (2) constrains the SIFT descriptors to be matched along with the flow vector $\mathbf{w}(\mathbf{p})$. The *small displacement term* in Eq. (3) constrains the flow vectors to be as small as possible when no other information is available. The *smoothness term* in Eq. (4) constrains the flow vectors of adjacent pixels to be similar. In this objective function, truncated L1 norms are used in both the data term and the smoothness term to account for matching outliers and flow discontinuities, with t and d as the threshold, respectively.

While SIFT flow has demonstrated the potential for aligning images across scenes [26], the original implementation scales poorly with respect to the image size. In SIFT flow, a pixel in one image can literally match to any other pixel in another image. Suppose the image has h^2 pixels, then the time and space complexity of the belief propagation algorithm to estimate the SIFT flow is $O(h^4)$. As reported in [26], the computation time for 145×105 images with an 80×80 searching neighborhood is 50 s. The original implementation of SIFT flow would require more than 2 h to process a pair of 256×256 images in our database with a memory usage of 16 GB to store the data term. To address the performance drawback, a coarse-to-fine SIFT flow matching scheme was designed to significantly improve the performance. As illustrated in Fig. 6, the basic idea consists of estimating the flow at a coarse level of image grid, and then gradually propagating and refining the flow from coarse to fine; please refer to [28] for details. As a result, the complexity of this coarse-to-fine algorithm is $O(h^2 \log h)$, a significant speed up compared to $O(h^4)$. The matching

²SIFT descriptors are computed at each pixel using a 16×16 window. The window is divided into 4×4 cells, and image gradients within each cell are quantized into a 8-bin histogram. Therefore, the pixel-wise SIFT feature is a 128-D vector.

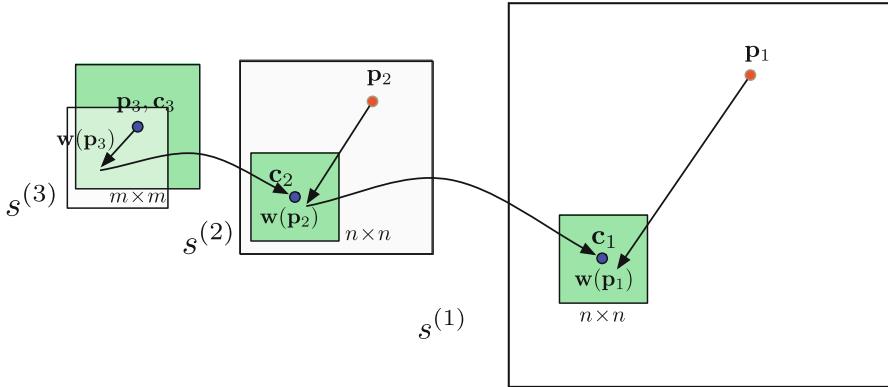


Fig. 6 An illustration of our coarse-to-fine pyramid SIFT flow matching. The *green square* denotes the searching window for \mathbf{p}_k at each pyramid level k . For simplicity, only one image is shown here, where \mathbf{p}_k is on image s_1 , and \mathbf{c}_k and $\mathbf{w}(\mathbf{p}_k)$ are on image s_2 . The details of the algorithm can be found in [28]

between two 256×256 images take 31s on a workstation with two quad-core 2.67 GHz Intel Xeon CPUs and 32 GB memory, in a C++ implementation. We also discovered that the coarse-to-fine scheme not only runs significantly faster, but also achieves lower energies most of the time compared to the ordinary matching algorithm.

Some SIFT flow examples are shown in Fig. 8, where dense SIFT flow fields (in column (f)) are obtained between the query images (a) and the nearest neighbors (c). It is trial to verify that the warped SIFT images (h) based on the SIFT flows (f) look very similar to the SIFT images (b) of the inputs (a), and that the SIFT flow fields (f) are piecewise smooth. The essence of SIFT flow is manifested in column (g), where the same flow field is applied to warp the RGB image of the nearest neighbor to the query. SIFT flow is trying to hallucinate the structure of the query image by smoothly shuffling the pixels of the nearest neighbors. Because of the intrinsic similarities within each object categories, it is not surprising that through aligning image structures often objects of the same categories are matched. In addition, it is worth noting that one object in the nearest neighbor can correspond to multiple objects in the query since the flow is asymmetric. This allows reuse of labels to parse multiple object instances.

4.3 Scene Parsing Through Label Transfer

Now that we have a large database of annotated images and a technique for establishing dense correspondences across scenes, we can transfer the existing annotations to a query image through dense scene alignment. For a given query

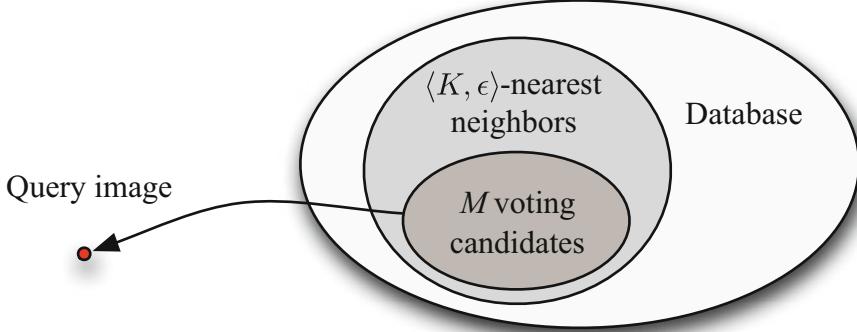


Fig. 7 For a query image, we first find a $\langle K, \epsilon \rangle$ -nearest neighbor set in the database using GIST matching [33]. The nearest neighbors are re-ranked using SIFT flow matching scores, and form a top M -voting candidate set. The annotations are transferred from the voting candidates to the query image

image, we retrieve a set of $\langle K, \epsilon \rangle$ -nearest neighbors in our database using GIST matching [33]. We then compute the SIFT flow from the query to each nearest neighbor, and use the achieved minimum energy (defined in Eq. (4)) to re-rank the $\langle K, \epsilon \rangle$ -nearest neighbors. We further select the top M re-ranked retrievals ($M \leq K$) to create our voting candidate set. This voting set will be used to transfer its contained annotations into the query image. This procedure is illustrated in Fig. 7.

Under this setup, scene parsing can be formulated as the following label transfer problem. For a query image I with its corresponding SIFT image s , we have a set of voting candidates $\{s_i, c_i, w_i\}_{i=1:M}$, where s_i , c_i , and w_i are the SIFT image, annotation, and SIFT flow field (from s to s_i) of the i th voting candidate. c_i is an integer image where $c_i(\mathbf{p}) \in \{1, \dots, L\}$ is the index of object category for pixel \mathbf{p} . We want to obtain the annotation c for the query image by transferring c_i to the query image according to the dense correspondence w_i .

We build a probabilistic MRF model to integrate multiple labels, prior information of object category, and spatial smoothness of the annotation to parse image I . Similar to that of [42], the posterior probability is defined as:

$$\begin{aligned} -\log P(c|I, s, \{s_i, c_i, w_i\}) &= \sum_{\mathbf{p}} \psi(c(\mathbf{p}); s, \{s'_i\}) + \alpha \sum_{\mathbf{p}} \lambda(c(\mathbf{p})) \\ &\quad + \beta \sum_{\{\mathbf{p}, \mathbf{q}\} \in \epsilon} \phi(c(\mathbf{p}), c(\mathbf{q}); I) + \log Z, \end{aligned} \quad (5)$$

where Z is the normalization constant of the probability. This posterior contains three components, i.e. likelihood, prior, and spatial smoothness.

The *likelihood* term is defined as

$$\psi(c(\mathbf{p}) = l) = \begin{cases} \min_{i \in \Omega_{\mathbf{p},l}} \|s_i(\mathbf{p}) - s_i(\mathbf{p} + \mathbf{w}(\mathbf{p}))\| & \Omega_{\mathbf{p},l} \neq \emptyset \\ \tau, & \Omega_{\mathbf{p},l} = \emptyset \end{cases} \quad (6)$$

where $\Omega_{\mathbf{p},l} = \{i; c_i(\mathbf{p} + \mathbf{w}(\mathbf{p})) = l\}, l = 1, \dots, L$ is the index set of the voting candidates whose label is l after being warped to pixel \mathbf{p} . τ is set to be the value of the maximum difference of SIFT feature: $\tau = \max_{s_1, s_2, \mathbf{p}} \|s_1(\mathbf{p}) - s_2(\mathbf{p})\|$.

The *prior* term is $\lambda(c(\mathbf{p}) = l)$ and indicates the prior probability that object category l appears at pixel \mathbf{p} . This is obtained from counting the occurrence of each object category at each location in the training set.

$$\lambda(c(\mathbf{p}) = l) = -\log \text{hist}_l(\mathbf{p}) \quad (7)$$

where $\text{hist}_l(\mathbf{p})$ is the spatial histogram of object category l .

The *smoothness* term is defined to bias the neighboring pixels into having the same label in the event that no other information is available, and the probability depends on the edge of the image: the stronger luminance edge, the more likely that the neighboring pixels may have different labels.

$$\phi(c(\mathbf{p}), c(\mathbf{q})) = \delta[c(\mathbf{p}) \neq c(\mathbf{q})] \left(\frac{\xi + e^{-\gamma \|I(\mathbf{p}) - I(\mathbf{q})\|^2}}{\xi + 1} \right) \quad (8)$$

where $\gamma = (2 < \|I(\mathbf{p}) - I(\mathbf{q})\|^2 >)^{-1}$ [42].

Notice that the energy function is controlled by four parameters, K and M that decide the mode of the model, and α and β that control the influence of spatial prior and smoothness. Once the parameters are fixed, we again use BP-S algorithm to minimize the energy. The algorithm converges in 2 s on a workstation with two quad-core 2.67 GHz Intel Xeon CPUs.

A significant difference between our model and that in [42] is that we have fewer parameters because of the nonparametric nature of our approach, whereas classifiers were trained in [42]. In addition, color information is not included in our model at the present as the color distribution for each object category is diverse in our databases.

5 Experiments

Extensive experiments were conducted to evaluate our system. We shall first report the results on a small scale database which we will refer to as the LMO database in Sect. 5.1; this database will aid us for an in-depth exploration of our model. Furthermore, we will report results on the SUN database, a larger and more challenging dataset in Sect. 5.2.

5.1 LabelMe Outdoor Database

As mentioned in Sect. 3, the LMO database consists of 2688 outdoor images, which have been randomly split into 2466 training and 200 test images. The images are densely labeled with 33 object categories using the LabelMe online annotation tool. Our scene parsing system is illustrated in Fig. 8. The system retrieves a $\langle K, \epsilon \rangle$ -nearest neighbor set for the query image (a), and further selects M voting candidates containing minimum SIFT matching scores. For illustration purposes we set $M = 3$ here. The original RGB image, SIFT image, and annotation of the voting candidates are shown in (c), (d), and (e), respectively. The SIFT flow field is visualized in (f) using the same visualization scheme as in [26], where hue indicates orientation and saturation indicates magnitude. After we warp the voting candidates into the query with respect to the flow field, the warped RGB (g) and SIFT image (h) are very close to the query (a) and (b), respectively. Combining the warped annotations in (i), the system outputs the parsing of the query in (j), which is close to the ground truth annotation in (k).

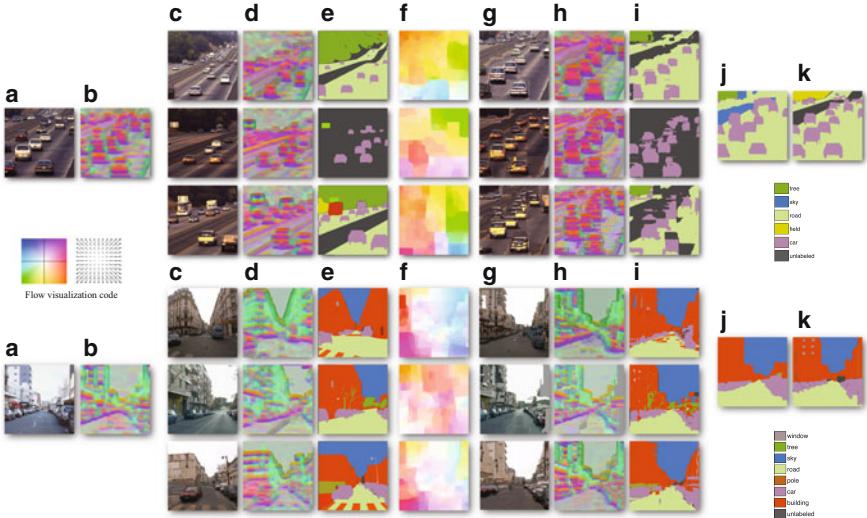


Fig. 8 System overview. For a query image, our system uses scene retrieval techniques such as [33] to find $\langle K, \epsilon \rangle$ -nearest neighbors in our database. We apply coarse-to-fine SIFT flow to align the query image to the nearest neighbors, and obtain top M as voting candidates ($M = 3$ here). (c)–(e): the RGB image, SIFT image, and user annotation of the voting candidates. (f): the inferred SIFT flow field, visualized using the color scheme shown on the *left* (hue: orientation; saturation: magnitude). From (g)–(i) are the warped version of (c)–(e) with respect to the SIFT flow in (f). Notice the similarity between (a) and (g), (b) and (h). Our system combines the voting from multiple candidates and generates scene parsing in (j) by optimizing the posterior. (k): the ground truth annotation of (a)

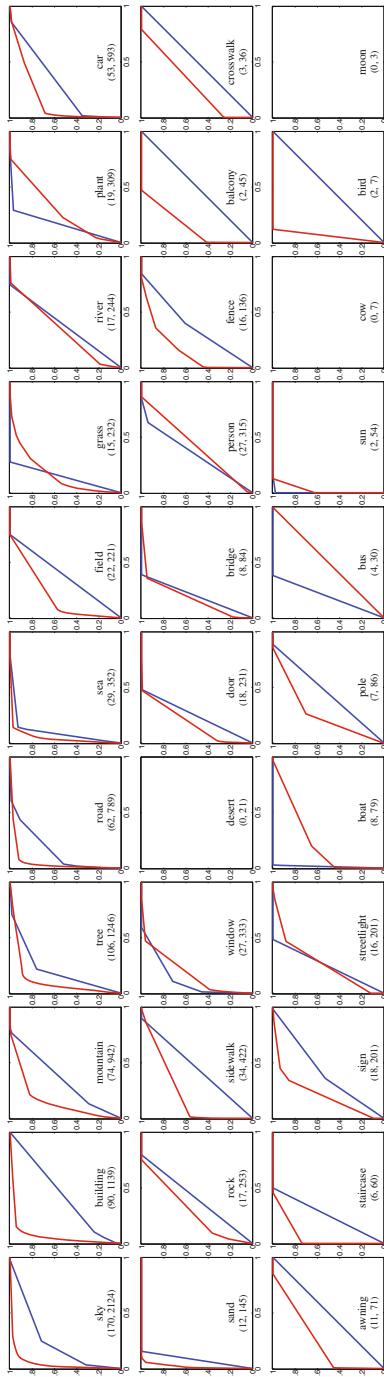


Fig. 9 The ROC curve of each individual pixel-wise binary classifier. *Red curve*: our system after being converted to binary classifiers; *blue curve*: the system in [8]. We used the convex hull to make the ROC curves strictly concave. The number (n, m) underneath the name of each plot is the quantity of the object instances in the test and training set, respectively. For example, (170, 2124) under “sky” means that there are 170 test images containing sky, and 2124 training images containing sky (there are in total 2488 training images and 200 test images). Our system obtains reasonable performance for objects with sufficient samples in both training and test sets, e.g., *sky*, *building*, *mountain*, and *tree*. We observe truncation in the ROC curves where there are not enough test samples, e.g., *field*, *sea*, *river*, *grass*, *plant*, *car*, and *sand*. The performance is poor for objects without enough training samples, e.g., *crosswalk*, *sign*, *boat*, *pole*, *sun*, and *bird*. The ROC does not exist for objects without any test samples, e.g., *desert*, *cow*, and *moon*. In comparison, our system outperforms or equals [8] for all object categories except for *grass*, *plant*, *boat*, *person*, and *bus*. The performance of [8] on our database is low because the objects have drastically different poses and appearances

(a) *Evaluation criterion*

We use average pixel-wise recognition rate \bar{r} (similar to precision or true positive) to evaluate the performance of our system, computed as

$$\bar{r} = \frac{1}{\sum_i m_i} \sum_i \sum_{\mathbf{p} \in \Lambda_i} \mathbf{1}(o(\mathbf{p}) = a(\mathbf{p}), a(\mathbf{p}) > 0), \quad (9)$$

where for pixel \mathbf{p} in image i , the ground truth annotation is $a(\mathbf{p})$ and system output is $o(\mathbf{p})$; for unlabeled pixels $a(\mathbf{p}) = 0$. Notation Λ_i is the image lattice for test image i , and $m_i = \sum_{\mathbf{p} \in \Lambda_i} \mathbf{1}(a(\mathbf{p}) > 0)$ is the number of labeled pixels for image i (some pixels are unlabeled). We also compute the per-class average rate r_l as

$$r_l = \frac{\sum_i \sum_{\mathbf{p} \in \Lambda_i} \mathbf{1}(o(\mathbf{p}) = a(\mathbf{p}), a(\mathbf{p}) = l)}{\sum_i \sum_{\mathbf{p} \in \Lambda_i} \mathbf{1}(a(\mathbf{p}) = l)}, \quad l = 1, \dots, L. \quad (10)$$

(b) *Results and comparisons*

Some label transfer results are shown in Fig. 10. The input image from the test set is displayed in column (a). We show the best match, its corresponding annotation, and the warped best match in (b), (c) and (d), respectively. While the final labeling constitutes the integration of the top M matches, the best match can provide the reader an intuition of the process and final result. Notice how the warped image (d) looks similar to the input (a), indicating that SIFT flow successfully matches image structures. The scene parsing results output by our system are listed in column (e) with parameter setting $K = 85$, $M = 9$, $\alpha = 0.06$, $\beta = 20$. The ground truth user annotation is listed in (f). Notice that the gray pixels in (f) are “unlabeled,” but our system does not generate “unlabeled” output. For samples 1, 5, 6, 8, and 9, our system generates reasonable predictions for the pixels annotated as “unlabeled.” The average pixel-wise recognition rate of our system is 76.67 % by excluding the “unlabeled” class [42]. Some failure examples from our system are shown in Fig. 11 when the system fails to retrieve images with similar object categories to the query, or when the annotation is ambiguous.

Overall, our system is able to predict the right object categories in the input image with a segmentation fit to image boundary, even though the best match may look different from the input, e.g. 2, 11, 12, and 17. If we divide the object categories into *stuff* (e.g., *sky*, *mountains*, *tree*, *sea* and *field*) and *things* (e.g., *cars*, *sign*, *boat* and *bus*) [1, 21], our system generates much better results for *stuff* than for *things*. The recognition rate for the top 7 object categories (all are “stuff”) is 82.720 %. This is because in our current system, we only allow one labeling for each pixel, and smaller objects tend to be overwhelmed by the labeling of larger objects. We plan to build a recursive system in our future work to further retrieve *things* based on the inferred *stuff*.

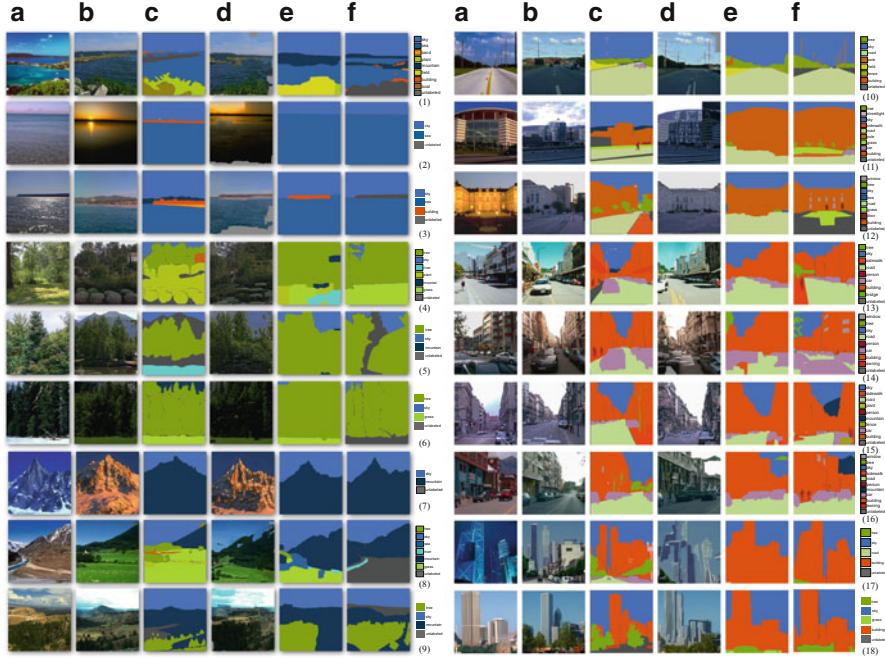


Fig. 10 Some scene parsing results output from our system. (a): query image; (b): the best match from nearest neighbors; (c): the annotation of the best match; (d): the warped version of (b) according to the SIFT flow field; (e): the inferred per-pixel parsing after combining multiple voting candidates; (f): the ground truth annotation of (a). The dark gray pixels in (f) are “unlabeled.” Notice how our system generates a reasonable parsing even for these “unlabeled” pixels

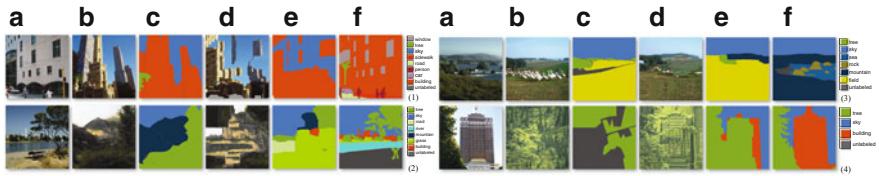


Fig. 11 Some typical failures. Our system fails when no good matches can be retrieved in the database. In (2), for example, since the best matches do not contain river, the input image is mistakenly parsed as a scene of grass, tree, and mountain in (e). The ground truth annotation is in (f). The failure may also come from ambiguous annotations, for instance in (3), where the system outputs field for the bottom part whereas the ground truth annotation is mountain

For comparison purposes, we downloaded and executed the texton-boost code from [42] using the same training and test data with the MRF turned off. The overall pixel-wise recognition rate of their system on our data set is 51.67 %, and the per-class rates are displayed in Fig. 12c. For fairness we also turned off the MRF model as well as spatial priors in our framework by setting $\alpha = \beta = 0$, and plotted the corresponding results in Fig. 12f. Clearly, our system outperforms [42] in

terms of both overall and per-class recognition rate. Similar performance to texton-boost is achieved by matching color instead of matching dense SIFT descriptors in our system, as shown in Fig. 12b. The recognition rate of the class *grass* and *sand* dramatically increases through matching color because color is the salient feature for these categories. However, the performance drops for other color-variant categories. This result supports the importance of matching appearance-invariant features in our label transfer system.

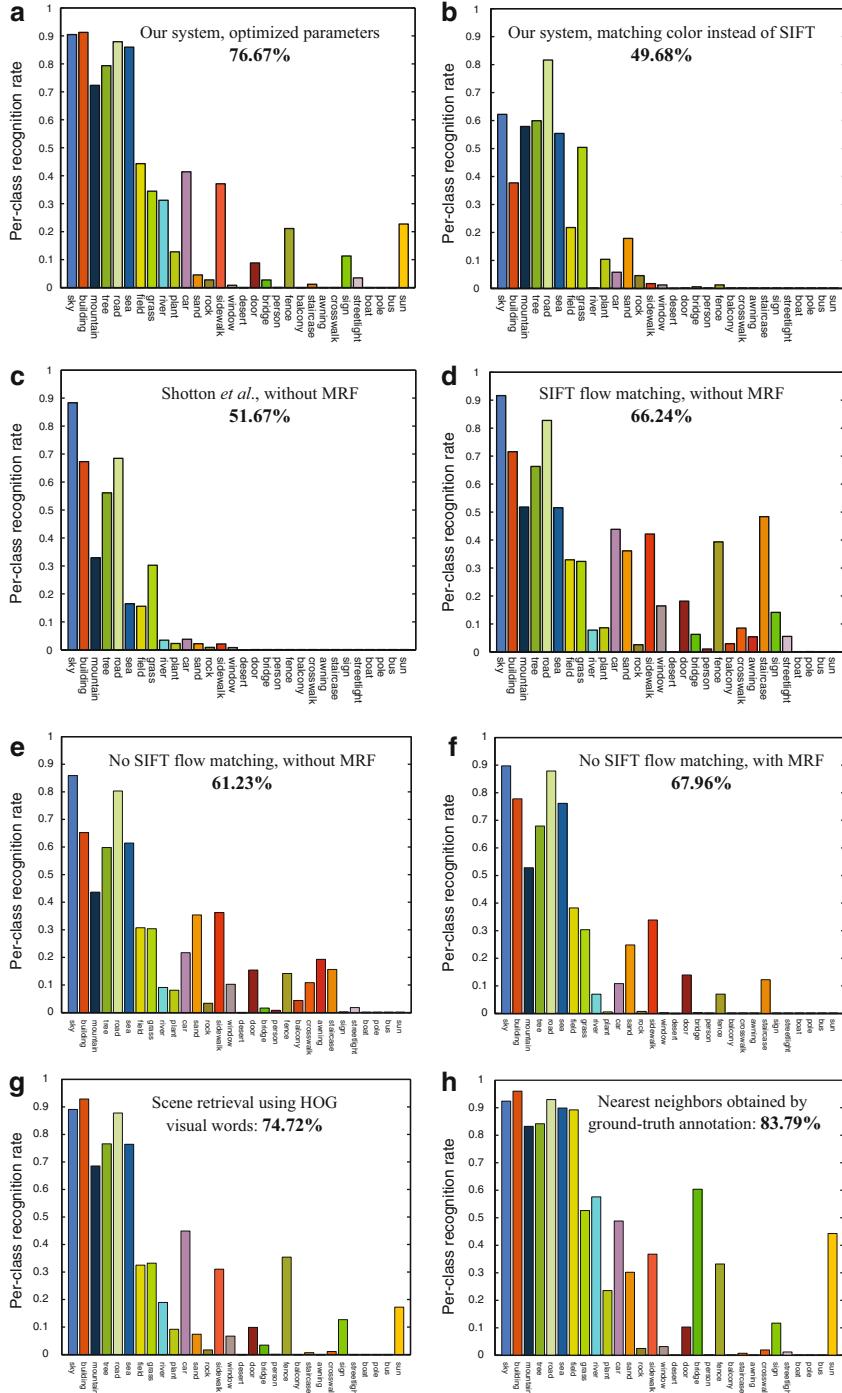
We also compared the performance of our system with a classifier-based system [8]. We downloaded their code and trained a classifier for each object category using the same training data. We converted our system into a binary object detector for each class by only using the per-class likelihood term. The per-class ROC curves of our system (red) and theirs (blue) are plotted in Fig. 9. Except for five object categories, *grass*, *plant*, *boat*, *person*, and *bus*, our system outperforms or equals theirs.

(c) Parameter selection

Since the SIFT flow module is essential to our system, we first test spatial smoothness coefficient λ in Eq. (4), which determines matching results. We compute the average pixel-wise recognition rate as a function of λ , shown in Fig. 13a. We first turn off the MRF model in the label transfer module by setting $\alpha = \beta = 0$, and find that when $\lambda = 0.7$, the maximal recognition rate is achieved. Then we turn on the MRF model by setting $\alpha = 0.1$, $\beta = 60$, and find that $\lambda = 0.7$ leads to a good performance as well. Therefore, we fix $\lambda = 0.7$ throughout our experiments.

We investigated the performance of our system by varying the parameters K , M , α and β . We have found that the influence of ϵ is smaller than that of K when ϵ is set such that most samples have K nearest neighbors. We vary $K = 1, 3, 5, 7, 9$, and $M = 1, 5, 10, \dots, 100$. For each combination of K and M ($K \leq M$), coordinate descend is used to find the optimal parameter of α and β by maximizing the recognition rate. We plot the recognition rate as a function of K for a variety of M 's in Fig. 13b. Overall, the recognition rate increases as more nearest neighbors are retrieved ($K \uparrow$) and more voting candidates are used ($M \uparrow$) since, obviously, multiple candidates are needed to transfer labels to the query. However, the recognition rate drops as K and M continue to increase as more candidates may introduce noise to the label transfer process. In particular, the recognition rate drops when K increases, suggesting that scene retrieval does not only serve as a way to obtain neighbors

Fig. 12 We study the performance of our system in-depth. (a) Our system with the parameters optimized for pixel-wise recognition rate. (b) Our system, matching RGB instead of matching dense SIFT descriptors. (c) The performance of [42], with the Markov random field component turned off, trained and tested on the same datasets as (a). From (d) to (f) we show the importance of SIFT flow matching and the MRF for label transfer by turning them on and off. In (g) and (h) we show the system performance affected by other scene retrieval methods. The performance in (h) shows the upper limit of our system, by adopting ideal scene retrieval using ground truth annotation (of course, the ground truth annotation is not available in practice). See text for more details



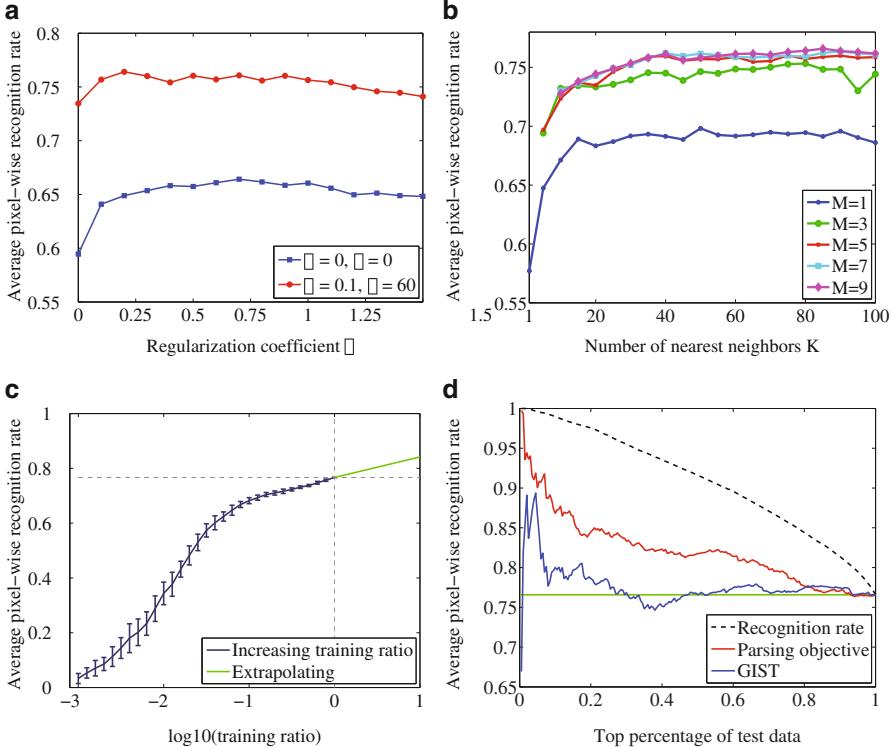


Fig. 13 (a): Recognition rate as a function of the spatial smoothness coefficient λ under two settings for α and β . (b): Recognition rate as a function of number of nearest neighbors K and the number of voting candidates M . Clearly, prior and spatial smoothness help improve the recognition rate. The fact that the curve drops down as we further increase K indicates that SIFT flow matching cannot replace scene retrieval. (c): Recognition rate as a function of the log training ratio while the test set is fixed. A subset of training samples are randomly drawn from the entire training set according to the training ratio to test how the performance depends on the size of the database. (d): Recognition rate as a function of the proportion of the top ranked test images according to metrics including GIST, the parsing objective in Eq. (5) and the recognition rate (with ground truth annotation). The black, dashed curve with recognition rate as sorting metric is the ideal case, and the parsing objective is better than GIST. These curves suggest the system is somewhat capable of distinguishing good parsing results from bad ones

for SIFT flow, but also rule out some bad images that SIFT flow would otherwise choose. The maximum performance is obtained when $K = 85$ and $M = 9$.

Because the regularity of the database is the key to the success, we remove the SIFT flow matching, i.e. set the flow vector to be zero for every pixel, and obtain an average recognition rate of 61.23 % without MRF and 67.96 % with MRF, shown in Fig. 12d,f, respectively. This result is significant because SIFT flow is the bottleneck of the system in terms of speed. A fast implementation of our system consists of removing the dense scene alignment module, and simply performing a grid-to-grid

label transfer (the likelihood term in the label transfer module still comes from SIFT descriptor distance).

How would different scene retrieval techniques affect our system? Other than the GIST distance used for retrieving nearest neighbors for the results in Fig. 12, we also use the spatial pyramid histogram intersection of HOG visual words and of the ground truth annotation, with the corresponding per-class recognition rate displayed in Fig. 12g,h, respectively. For this database, GIST performs slightly better than HOG visual words. We also explore an upper bound of the label transfer framework in the ideal scenario of having access to perfect scene matching. In particular, we retrieve the nearest neighbors for each image using their ground truth annotations; please refer to Sect. 4.1 for the details. This upper bound is an 83.79 % recognition rate.

To further understand our data-driven system, we evaluated the performance of the system as a function of the ratio of training samples while fixing the test set. For each fixed ratio, we formed a small training database randomly drawing samples from the original database and evaluated the performance of the system under this database. This experiment was performed 15 times for each ratio to obtain a mean and standard deviation of its performance shown in Fig. 13c. Clearly, the recognition rate depends on the size of the training database. Using the last 10 data points for extrapolation, we found that if we increase the training data by 10 times (corresponding to 1 on the horizontal axis), the recognition rate may increase to 84.16 %.³ Note, however, that this linear extrapolation does not consider potential saturation issues as it can be observed when more than 10 % of training samples were used. This indicates that the training quantity is reasonable for this database.

Another aspect we evaluated is the capacity to detect *good* and *bad* parsing results. For this purpose, we re-rank the test image using three metrics: recognition rate (the ideal metric; evaluated with respect to ground truth), the parsing objective in Eq. (5) after energy minimization, and the average GIST-based distance to the nearest neighbors. After ranking, we computed the accumulated average recognition rate as a function of the ratio of testing samples, as shown in Fig. 13b. If we use the parsing objective as a metric, for example, then the average recognition rate can be greater than 80 % when only the top 75 % parsing results are picked. The system can reject the remaining 25 % with low scores.

5.2 SUN Database

We further evaluated the performance of our system on the SUN database [51], which contains 9556 images of both indoor and outdoor scenes. This database contains a total of 515 object categories; the pixel frequency counts for the top 100 categories are displayed in Fig. 15a. The data corpus is randomly split into 8556

³This extrapolation is different from moving to a larger database in Sect. 5.2, where indoor scenes are included. This number is anticipated only when images similar to the LMO database are added.

images for training and 1000 for testing. The structure of the database is visualized in Fig. 14 using the same technique to plot Fig. 5, where the image distance is measured by the ground truth annotation. Notice the clear separation of indoor (left) and outdoor (right) scenes in this database. Moreover, the images are not evenly distributed in the space; they tend to reside around a few clusters. This phenomenon is consistent with human perception, drawing a clear separation between outdoor and indoor images.

Some scene parsing results are displayed in Fig. 16 in the same format as Fig. 10. Since the SUN database is a super set of the LMO database, the selection of results is slightly biased towards indoor and activity scenes. Overall, our system performs reasonably parsing these challenging images.

We also plot the per-class performance in Fig. 15. In (a), we show the pixel-wise frequency count of the top 100 object categories. Similar to LMO, this prior distribution is heavily biased towards stuff-like classes, e.g. *wall*, *sky*, *building*, *floor*, and *tree*. In (b), the performance is achieved when the ground truth annotation is used for scene retrieval. Again, the average 64.45 % recognition rate reveals the upper limit and the potential of our system in the idealized case of perfect nearest neighbor retrieval. In (c) and (d), the performance using HOG and GIST features for scene retrieval is plotted, suggesting that the HOG visual words features outperform GIST for this larger database. This is consistent with the discovery that HOG feature is the best among a set of features including GIST in scene recognition in the SUN database [51].

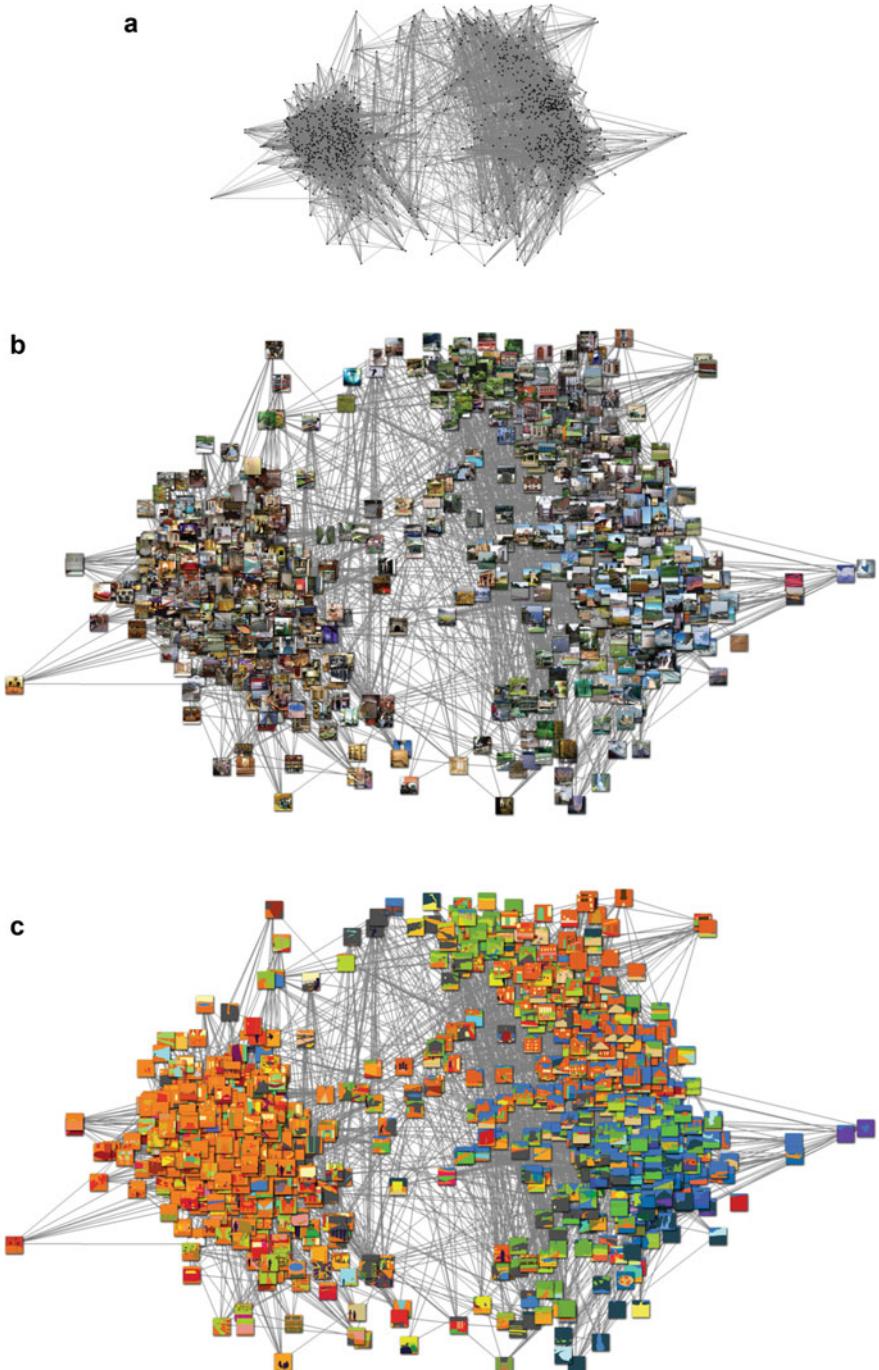
Overall, the recognition rate on the SUN database is lower than that on the LMO database. A possible explanation for this phenomenon is that indoor scenes contain less regularity compared to outdoor ones, and there are 515 object categories in SUN whereas there are only 33 categories in LMO.

6 Discussion

I. Label transfer: an open, database-driven framework for image understanding

A unique characteristic of our nonparametric scene parsing system is its openness: to support more object categories, one can simply add more images of the new categories into the database without requiring additional training. This is an

Fig. 14 Visualization of the SUN database [51] using 1200 random images. Notice that the SUN database is larger but not necessarily denser than the LMO database. We use the spatial pyramid histogram intersection distance of the ground truth annotation to measure the distance between the images and project them to a 2D space using scaled multidimensional scaling (MDS). Clearly the images are clustered to indoor (*left*) and outdoor (*right*) scenes, and there is smooth transition in between. In the outdoor cluster, we observe the change from *garden*, *street*, to *mountain* and *valley* as we move from top to bottom. (a) Nodes and edges. (b) RGB images. (c) Ground-truth annotations



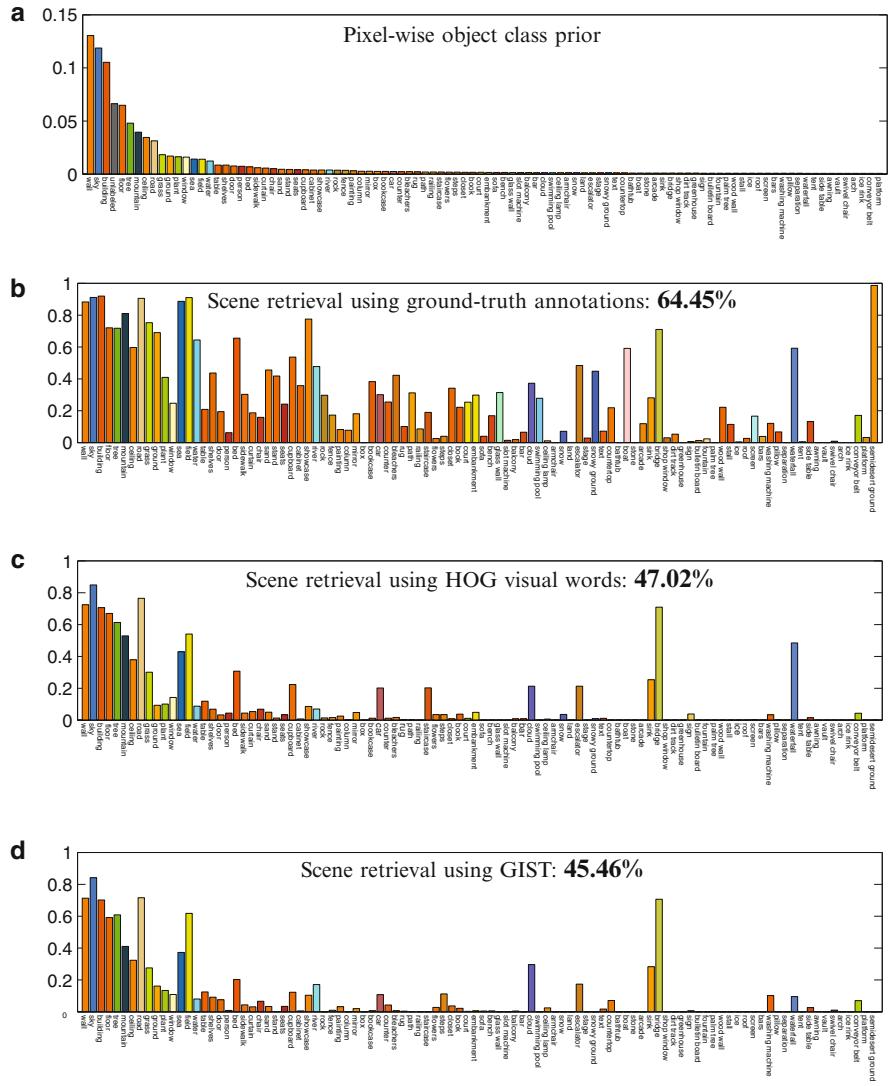
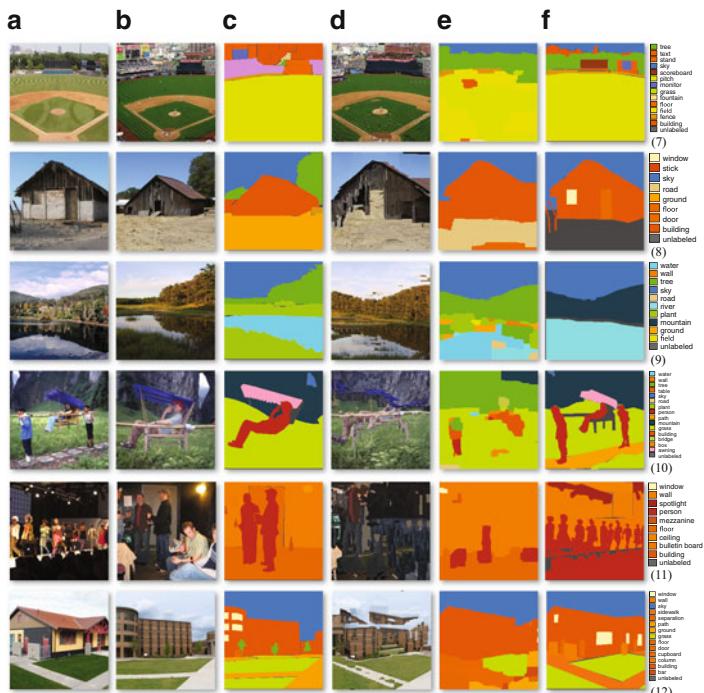
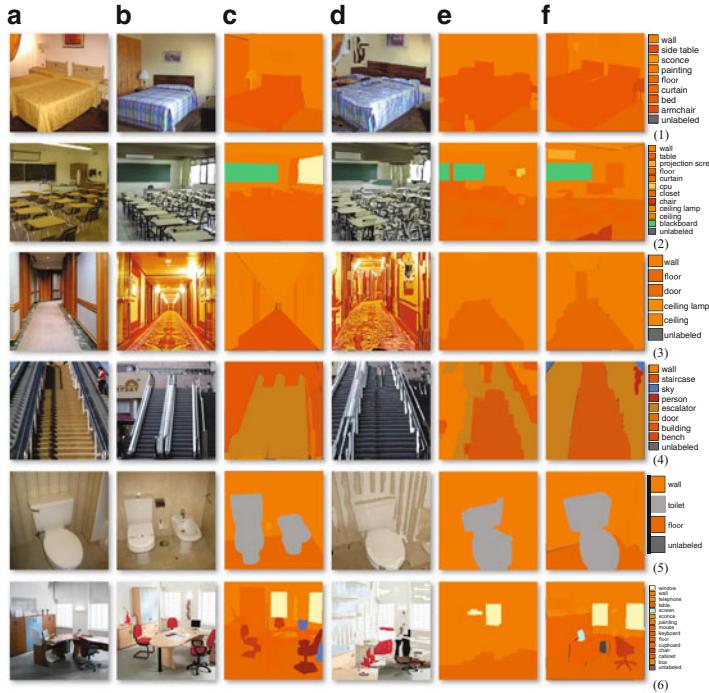


Fig. 15 The per-class recognition rate of running our system on the SUN database. (a) Pixel frequency histogram of the top 100 object categories. (b): The per-class recognition rate when the ground truth annotation is used for scene retrieval. This is the upper limit performance that can be achieved by our system. (c) and (d): per-class recognition rate using HOG and GIST for scene retrieval, respectively. The HOG visual words features generate better results for this larger database



advantage over classical learning-based systems where all the classifiers have to be retrained when new object categories are inserted into the database.

Although there is no parametric model (probabilistic distributions or classifiers) of object appearances in our system, the ability of recognizing objects depends on reliable image matching across different scenes. When good matches are established between objects in the query and objects in the nearest neighbors in the annotated database, the known annotation naturally explains the query image as well. We chose SIFT flow [28] to establish a semantically meaningful correspondence between two different images. Nonetheless, this module can easily be substituted by other or better scene correspondence methods.

Although context is not explicitly modeled in our system, our label transfer-based scene parsing system naturally embeds contextually coherent label sets. The nearest neighbors retrieved in the database and re-ranked by SIFT flow scores mostly belong to the same type of scene category, implicitly ensuring contextual coherence. Using Fig. 16 (9) as an example, we can see that even though the reflection of the mountain has been misclassified to *field*, *ground*, *tree*, and *plant*, the parsing result is context-coherent.

II. The role of scene retrieval

Our nonparametric scene parsing system depends largely on the scene retrieval technique, through which the nearest neighbors of the query image in the large database are obtained. We have tried two popular techniques, GIST, and HOG visual words, and have found that GIST-based retrieval yields higher performance in the LMO database, whereas HOG visual words tend to retrieve better neighbors in the SUN database. We also show the upper-bound performance of our system by using the ground truth annotation for scene retrieval. This upper-bound provides an intuition of the efficacy of our system given an ideal scene retrieval system. The recent advances in this area by combining multiple kernels [51] point out promising directions for scene retrieval.

III. Better evaluation criterion

Presently, we use a simple criterion, pixel-wise recognition rate, to measure the performance of our scene parsing system. A pixel is correctly recognized only when the parsing result is exactly the same as the ground truth annotation. However, human annotation can be ambiguous. For instance, in the parsing example depicted in Fig. 16 (9), the pixel-wise recognition rate is low because the *mountain* is recognized as *tree*, and the *water* is recognized as *river*. While in our current evaluation framework these pixels are considered misclassified, this parsing would

Fig. 16 Some scene parsing results on the SUN database following the same notation as in Fig. 10. Note that the LabelMe Outdoor database is a subset of the SUN database, but the latter contains a larger proportion of indoor and activity-based scenes. As it is in all cases, the success of the final parsing depends on the semantic similarity of the query in (a) to the warped support image (d). Example (10) and (11) are two failure examples where many people are not correctly labeled

be considered accurate when evaluated by a human. A more precise evaluation criterion would take synonyms into account. Another example is shown in Fig. 16 (12), where the windows are not present in the parsing result. Therefore, the *window* pixels are labeled wrong because they are classified as *building*, which is a more favorable label than, for example, *car*, as windows tend to appear on top of buildings. A superior evaluation criterion should also consider co-occurrence and occlusion relationships. We leave these items as future work.

IV. Nonparametric vs. parametric approaches

In this chapter, we have demonstrated promising results of our nonparametric scene parsing system using label transfer by showing how it outperforms existing recognition-based approaches. However, we do not believe that our system alone is the ultimate answer to image understanding since it does not work well for small objects such as *person*, *window*, *bus*, etc., which can be better handled using detectors. Moreover, pixel-wise classifiers such as *textonboost* can also be useful when good matching cannot be established or good nearest neighbors can hardly be retrieved. Therefore, a natural future step is to combine these methods for scene parsing and image understanding.

7 Conclusion

We have presented a novel, nonparametric scene parsing system to integrate and transfer the annotations from a large database to an input image via dense scene alignment. A coarse-to-fine SIFT flow matching scheme is proposed to reliably and efficiently establish dense correspondences between images across scenes. Using the dense scene correspondences, we warp the pixel labels of the existing samples to the query. Furthermore, we integrate multiple cues to segment and recognize the query image into the object categories in the database. Promising results have been achieved by our scene alignment and parsing system on challenging databases. Compared to existing approaches that require training for each object category, our nonparametric scene parsing system is easy to implement, has only a few parameters, and embeds contextual information naturally in the retrieval/alignment procedure.

References

1. Adelson, E.H.: On seeing stuff: the perception of materials by humans and machines. In: SPIE, Human Vision and Electronic Imaging VI, pp. 1–12 (2001)
2. Belongie, S., Malik, J., Puzicha, J.: Shape context: a new descriptor for shape matching and object recognition. In: Advances in Neural Information Processing Systems (NIPS) (2000)
3. Berg, A., Berg, T., Malik, J.: Shape matching and object recognition using low distortion correspondence. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005)

4. Borg, I., Groenen, P.: *Modern Multidimensional Scaling: Theory and Applications*, 2nd edn. Springer, New York (2005)
5. Branson, S., Wah, C., Babenko, B., Schroff, F., Welinder, P., Perona, P., Belongie, S.: Visual recognition with humans in the loop. In: European Conference on Computer Vision (ECCV) (2010)
6. Choi, M.J., Lim, J.J., Torralba, A., Willsky, A.: Exploiting hierarchical context on a large database of object categories. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010)
7. Crandall, D., Felzenszwalb, P., Huttenlocher, D.: Spatial priors for part-based recognition using statistical models. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2005)
9. Desai, C., Ramanan, D., Fowlkes, C.: Discriminative models for multi-class object layout. In: IEEE International Conference on Computer Vision (ICCV) (2009)
10. Divvala, S.K., Hoiem, D., Hays, J.H., Efros, A.A., Hebert, M.: An empirical study of context in object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
11. Edwards, G., Cootes, T., Taylor, C.: Face recognition using active appearance models. In: European Conference on Computer Vision (ECCV) (1998)
12. Efros, A.A., Leung, T.: Texture synthesis by non-parametric sampling. In: IEEE International Conference on Computer Vision (ICCV) (1999)
13. Felzenszwalb, P., Huttenlocher, D.: Pictorial structures for object recognition. *Int. J. Comput. Vis.* **61**(1), 55–79 (2005)
14. Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2008)
15. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2003)
16. Frome, A., Singer, Y., Malik, J.: Image retrieval and classification using local distance functions. In: Advances in Neural Information Processing Systems (NIPS) (2006)
17. Galleguillos, C., McFee, B., Belongie, S., Lanckriet, G.R.G.: Multi-class object localization by combining local contextual interactions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010)
18. Grauman, K., Darrell, T.: Pyramid match kernels: Discriminative classification with sets of image features. In: IEEE International Conference on Computer Vision (ICCV) (2005)
19. Gupta, A., Davis, L.S.: Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In: European Conference on Computer Vision (ECCV) (2008)
20. Hays, J., Efros, A.A.: Scene completion using millions of photographs. *ACM SIGGRAPH* **26**(3) (2007)
21. Heitz, G., Koller, D.: Learning spatial context: using stuff to find things. In: European Conference on Computer Vision (ECCV) (2008)
22. Hoiem, D., Efros, A., Hebert, M.: Putting objects in perspective. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2006)
23. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. II, pp. 2169–2178 (2006)
24. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proc. IEEE* **86**(11), 2278–2324 (1998)
25. Liang, L., Liu, C., Xu, Y.Q., Guo, B.N., Shum, H.Y.: Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph. (TOG)* **20**(3), 127–150 (2001)
26. Liu, C., Yuen, J., Torralba, A., Sivic, J., Freeman, W.T.: SIFT flow: dense correspondence across different scenes. In: European Conference on Computer Vision (ECCV) (2008)

27. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing: label transfer via dense scene alignment. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2009)
28. Liu, C., Yuen, J., Torralba, A.: SIFT flow: dense correspondence across different scenes and its applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(5), 978–994 (2011)
29. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **60**(2), 91–110 (2004)
30. Murphy, K.P., Torralba, A., Freeman, W.T.: Using the forest to see the trees: a graphical model relating features, objects, and scenes. In: Advances in Neural Information Processing Systems (NIPS) (2003)
31. Nister, D., Stewenius, H.: Scalable recognition with a vocabulary tree. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2006)
32. Obdrzalek, S., Matas, J.: Sub-linear indexing for large scale object recognition. In: British Machine Vision Conference (2005)
33. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int. J. Comput. Vis.* **42**(3), 145–175 (2001)
34. Park, D., Ramanan, D., Fowlkes, C.: Multiresolution models for object detection. In: European Conference on Computer Vision (ECCV) (2010)
35. Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., Belongie, S.: Objects in context. In: IEEE International Conference on Computer Vision (ICCV) (2007)
36. Russell, B.C., Torralba, A., Liu, C., Fergus, R., Freeman, W.T.: Object recognition by scene alignment. In: Advances in Neural Information Processing Systems (NIPS) (2007)
37. Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T.: LabelMe: a database and web-based tool for image annotation. *Int. J. Comput. Vis.* **77**(1–3), 157–173 (2008)
38. Russell, B.C., Efros, A.A., Sivic, J., Freeman, W.T., Zisserman, A.: Segmenting scenes by matching image composites. In: Advances in Neural Information Processing Systems (NIPS) (2009)
39. Savarese, S., Winn, J., Criminisi, A.: Discriminative object class models of appearance and shape by correlatons. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2006)
40. Shakhnarovich, G., Viola, P., Darrell, T.: Fast pose estimation with parameter sensitive hashing. In: IEEE International Conference on Computer Vision (ICCV) (2003)
41. Shechtman, E., Irani, M.: Matching local self-similarities across images and videos. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2007)
42. Shotton, J., Winn, J., Rother, C., Criminisi, A.: Textonboost for image understanding: multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Int. J. Comput. Vis.* **81**(1), 2–23 (2009)
43. Sivic, J., Zisserman, A.: Video Google: a text retrieval approach to object matching in videos. In: IEEE International Conference on Computer Vision (ICCV) (2003)
44. Suderth, E., Torralba, A., Freeman, W.T., Willsky, W.: Describing visual scenes using transformed dirichlet processes. In: Advances in Neural Information Processing Systems (NIPS) (2005)
45. Tighe, J., Lazebnik, S.: Superparsing: Scalable nonparametric image parsing with superpixels. In: European Conference on Computer Vision (ECCV) (2010)
46. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: a large dataset for non-parametric object and scene recognition. In: IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2008)
47. Turk, M., Pentland, A.: Face recognition using eigenfaces. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (1991)
48. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2001)
49. Weber, M., Welling, M., Perona, P.: Unsupervised learning of models for recognition. In: European Conference on Computer Vision (ECCV) (2000)
50. Winn, J., Criminisi, A., Minka, T.: Object categorization by learned universal visual dictionary. In: IEEE International Conference on Computer Vision (ICCV) (2005)

51. Xiao, J., Hays, J., Ehinger, K., Oliva, A., Torralba, A.: SUN database: large-scale scene recognition from abbey to zoo. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010)
52. Yang, Y., Hallman, S., Ramanan, D., Fowlkes, C.: Layered object detection for multi-class segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010)

Joint Inference in Weakly-Annotated Image Datasets via Dense Correspondence

Michael Rubinstein, Ce Liu, and William T. Freeman

Abstract We present a principled framework for inferring pixel labels in weakly annotated image datasets. Most previous, example-based approaches to computer vision rely on a large corpus of densely labeled images. However, for large, modern image datasets, such labels are expensive to obtain and are often unavailable. We establish a large-scale graphical model spanning all labeled and unlabeled images, then solve it to infer pixel labels *jointly* for all images in the dataset while enforcing consistent annotations over similar visual patterns. This model requires significantly less labeled data and assists in resolving ambiguities by propagating inferred annotations from images with stronger local visual evidences to images with weaker local evidences. We apply our proposed framework to two computer vision problems: image annotation with semantic segmentation, and object discovery and co-segmentation (segmenting multiple images containing a common object). Extensive numerical evaluations and comparisons show that our method consistently outperforms the state of the art in automatic annotation and semantic labeling, while requiring significantly less labeled data. In contrast to previous co-segmentation techniques, our method manages to discover and segment objects well even in the presence of substantial amounts of noise images (images not containing the common object), as typical for datasets collected from Internet search.

This work was done while Michael Rubinstein was a PhD student at MIT, during two summer internships at Microsoft Research, and while Ce Liu was a researcher at Microsoft Research.

M. Rubinstein (✉) • C. Liu
Google Research, Cambridge, MA
e-mail: mrub@google.com; celiu@google.com

W.T. Freeman
Google Research, Cambridge, MA
MIT CSAIL
e-mail: wfreeman@google.com

1 Introduction

Natural images consist of many repetitive patterns, such as corners, boundaries and textures, as well as repetitive parts, objects, and scenes. Such repetitions occur not only within an image, but also across images. For example, when querying an image search engine using a textual phrase, we often obtain many visually similar images consisting of the object or scene of interest.

There are two main approaches in computer vision to model such repetitive patterns. One approach—the *parametric approach*—explicitly learns a dictionary of visual patterns and their variations. Such parametric models have been successfully used for texture synthesis [56], image denoising [57], and object recognition [10, 12]. The other approach is the *nonparametric approach*, which attempts to build a graph for the patterns such that each pattern is connected to its lookalikes, also known as its “*neighbors*.” Information can then be conveniently propagated or transferred from the nearest neighbors to the query pattern without the need to explicitly model the pattern. Such methods have been widely used for super resolution [13], texture synthesis [29], and image understanding [21, 30], and, despite their relative simplicity, they often turn out to perform better than the parametric, model-based approaches.

Recent techniques for establishing dense *correspondences* between images of different scenes, such as SIFT flow [31] and PatchMatch [2], have facilitated the design and implementation of such nonparametric, information-transfer systems, where the information can be labels [30], motion [31], depth [21], and pixel color [47]. The idea of information transfer is illustrated at the top of Fig. 1. For a query image x , the system first finds a set of images x_i that are visually similar to x within a dataset of images, where each x_i is associated with some known information y_i (e.g., semantic labels, depth, or motion). After dense correspondence is established between x and each x_i , each y_i is warped to x based on the computed correspondence, and an estimate of y for x is typically obtained by integrating multiple warped y_i ’s. Such a system performs well in generating the function $x \rightarrow y$.

The main drawback of information-transfer methods, however, is that they rely on regularities in a large corpus of training images for which the information to be transferred (e.g. depth, motion, 3D) is “clean” and known. In large, modern image datasets, such information is expensive to obtain and is often noisy or unavailable. Moreover, when classifying multiple new images, these methods typically solve

Fig. 1 Information transfer in fully annotated and weakly annotated datasets. *Top:* a standard example-based framework for computer vision. Information such as class labels, depth, or motion is transferred by means of pixel correspondences from a large pool of labeled images to an unlabeled query. *Bottom:* our framework for joint inference in weakly annotated datasets. A large graphical model is established spanning all labeled and unlabeled images, then solved to infer annotations jointly in all the images

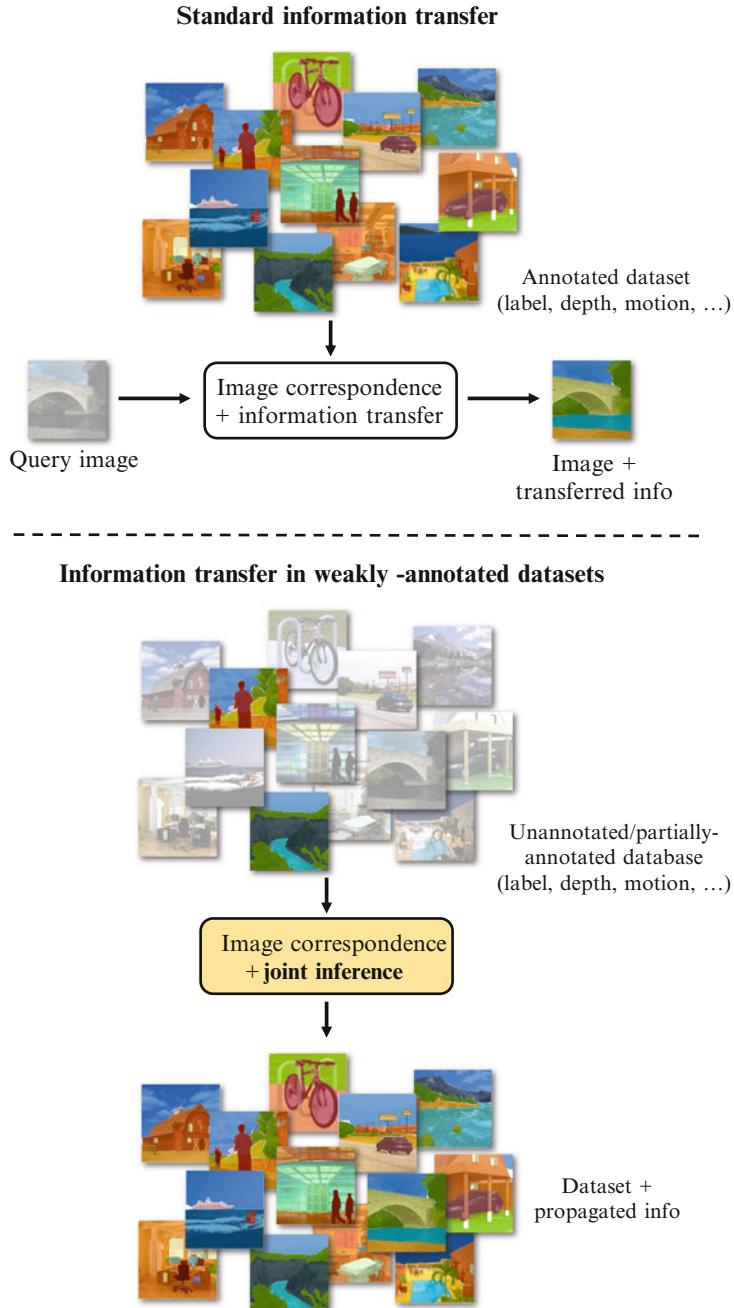


Fig. 1 (continued)

for each new image independently, which often results in inconsistent annotations across images due to visual ambiguities.

We therefore propose a new framework for dealing with weakly annotated datasets. In such datasets, it may be that none of the closest neighbors of an image is labeled, and so traditional, correspondence-based approaches cannot be used. Instead, we gradually infer the pixel labels and propagate the information through the dataset *jointly* in all the images. In essence, our framework can be seen as an extension of the seminal work of Freeman et al. [13] to cases with scarce training data.

By means of dense image correspondence, we establish a large-scale Markov Random Field model (MRF) spanning all labeled and unlabeled images (Fig. 3), and infer pixel labels jointly for the entire dataset rather than for a single image. Pixel correspondences are used to capture the visual variability of semantically related features across the dataset, and for inferring labels that are consistent over similar image patterns across different images.

Our model is effectively optimized by efficient belief propagation algorithms embedded within an expectation–maximization (EM) scheme, alternating between estimating the likelihood of pixel labels and pixel-wise label inference, while optionally refining the image graph structure during the optimization. Our optimization technique is not fundamentally new, but is designed to make full use of the available (sparse) annotations, and leverages modern computer architectures to scale efficiently for parallel computation.

To illustrate the potential breadth of our framework, we apply it to two important computer vision applications (Fig. 2). The first is *image annotation and semantic labeling*, where the goal is to automatically annotate many images with a set of word tags and a pixel-wise map showing where each word tag occurs (Fig. 2a). The second is *object discovery and segmentation*, in which we seek to automatically segment multiple images containing a common object (Fig. 2b). We consider the first application in a weakly labeled setup, where only sparse image-level tags and relatively few (or none) pixel labels are given. In the latter application, we assume that other than the fact that the image set contains some common object, there is no additional information available on the images or the common object class. Both applications can be useful for automatic generation of large-scale training sets for object detectors/classifiers, data-driven image synthesis, as well as for improving image-to-text relevance and Internet image search. They can also support applications in other domains, such as robotics, surveillance, and public safety.

We show how each of these two problems can be casted as a specific configuration of our proposed joint inference framework, and demonstrate that existing approaches to these problems do not perform well in more challenging, weakly labeled scenarios. For image annotation, we conducted extensive experiments on standard large-scale datasets, namely LabelMe [41], ESP [52], and IAPR [15], showing that our system consistently outperforms the state of the art in automatic annotation and semantic labeling, while requiring significantly less labeled data. For object discovery, our algorithm produces state-of-the-art results on the established MSRC and iCoseg co-segmentation datasets, and provides considerable

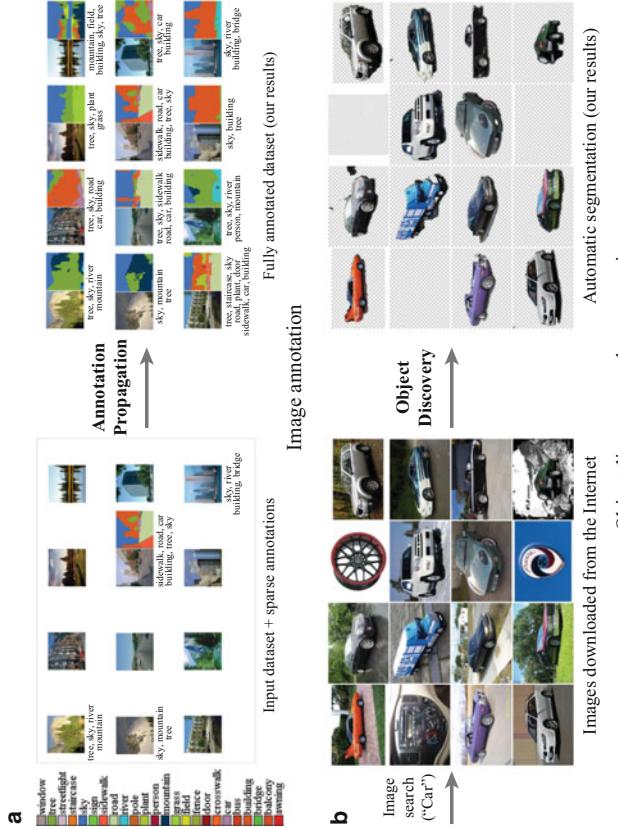


Fig. 2 Applications supported by our framework. **(a)** Automatic annotation (Sect. 4): the input is a weakly annotated image dataset with sparse image-level tags and few (or none) pixel labels, and the output dataset consists of a set of word tags for each image and a pixel-wise map showing where each word tag occurs. **(b)** Object discovery and segmentation (Sect. 5): the input is a large image dataset containing a common object, such as the set of images returned by an image search engine for a given query (e.g., when searching for “car”), and the goal is to label each pixel in the dataset according to whether or not it belongs to the underlying common object

improvement over previous co-segmentation methods on several new challenging Internet datasets containing rigid and non-rigid object categories.

The rest of the chapter is organized as follows. In Sect. 2 we review previous work related to our approach and the applications we explored. In Sect. 3 we formulate our framework. In Sects. 4 and 5 we apply the framework to the aforementioned applications. In each of these sections we first formulate the problem as a specific configuration of our framework, and then present experiments and results. We conclude our findings in Sect. 6.

A prior version of this work appeared in the 12th European Conference on Computer Vision (ECCV), Florence 2012 [38] and in the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland 2013 [39]. Results, comparisons, and code are available on the project web pages at <http://people.csail.mit.edu/mrub/annotation> and <http://people.csail.mit.edu/mrub/ObjectDiscovery>.

2 Related Work

2.1 Image Graphs

Large-scale image graphs are becoming a fundamentally important representation for image datasets. Several works have utilized it in the past for exploring and navigating image collections. For example, “Image Webs” [16] discovers corresponding regions between images and uses spectral graph theory to capture the connectivity in an image dataset. The discovered connectivity is then used for revealing global structures in the dataset (such as paths linking between images), and for supporting a Photo-Tourism-style navigation [45]. Videoscapes [49] added the temporal domain to the image graph construction, aligning video frames not only in space, but also in time, in order to interactively explore unstructured video collections.

Recently, Liu et al. [32] proposed a label propagation algorithm for joint image parsing, combining relationships among patches and mutual information in sub-graphs of the image graph. Faktor and Irani [9] efficiently estimate corresponding regions between images to automatically cluster an image dataset. Kim et al. [24] jointly discover matching images and parse their content in multiple photo streams to detect collective storylines. All these work, which were applied to specific problems, demonstrate the power of exploiting regularities and structures in image datasets to perform nontrivial inference tasks.

2.2 Image Annotation and Semantic Segmentation

In computer vision, scholars have investigated image annotation in two directions. One methodology uses *image similarities* to transfer textual annotation from labeled

images to unlabeled samples, under the assumption that *similar images should have similar annotations*. Notably, Makadia et al. [33] recently proposed a simple baseline approach for auto-annotation based on global image features and a greedy algorithm for transferring tags from similar images. ARISTA [53] automatically annotates a web dataset of billions of images by transferring tags via near-duplicates. Although those methods have clearly demonstrated the merits of using similar images to transfer annotations, they do so only between globally very similar images, and cannot account for locally similar patterns.

Consequently, the other methodology focused on dense annotation of images, known as *semantic labeling* [30, 42, 43, 48], where correspondences between text and local image features are established for annotation propagation: *similar local features should have similar labels*. These methods often aim to label each pixel in an image using models learned from a training database. In [4] and [11], relationships between text and visual words are characterized by conventional language translation models. More recently, Shotton et al. [42] proposed to train a discriminative model based on the texton representation, and to use conditional random fields (CRF) to combine various cues to generate spatially smooth labeling. The authors later extended their approach [43] by using randomized decision forests for a significant speedup. Liu et al. [30] proposed a nonparametric approach to semantic labeling, where text labels are transferred from a labeled database to parse a query image via dense scene correspondences.

Since predicting annotation from image features is by nature ambiguous (e.g., textureless regions can be *sky*, *wall*, or *ceiling*), such methods rely on regularities in a large corpus of training data of pixel-wise densely labeled images. However, for large image databases, high-quality pixel labels are very expensive to obtain. Furthermore, the annotation is typically computed *independently* for each test image, which often results in inconsistent annotations due to visual ambiguities.

2.3 Object Discovery and Segmentation

The task of simultaneously segmenting multiple images is known as *co-segmentation*, where joint segmentation essentially serves as a means of compensating for the lack of supervisory data, allowing to infer the visual properties of the foreground object even in the absence of a priori information about the object or the images.

While numerous co-segmentation methods have been proposed, they were shown to work well mostly on small datasets, namely MSRC and iCoseg, containing salient and similar objects. In fact, in most of the images in those datasets the foreground can be quite easily separated from the background based on each image alone (i.e., without co-segmentation, see Sect. 5.4).

However, Internet image collections, such as the ones returned by image search engines for a given user query, are significantly larger and more diverse (Fig. 2b). Not only do the objects in images downloaded from the Internet exhibit drastically

different style, color, texture, shape, pose, size, location, and view-point; but such image collections also contain many *noise* images—images which do not contain the object of interest at all. These challenges, as we demonstrate, pose great difficulties on existing co-segmentation techniques. In particular, most co-segmentation methods assume every image contains the object of interest, and hence are unable to handle dataset noise.

Object discovery has been intensively studied in computer vision. In a supervised setup, objects were treated as topics and images as documents, and generative models such as Latent Dirichlet Allocation (LDA) and Hierarchical Pitman-Yor (HPY) have been used to learn the distribution and segmentation of multiple classes simultaneously [40, 44]. Winn and Jojic [54] propose a generative model for the distribution of mask, edge, and color for visual objects with respect to a smooth deformation field. Although good object recovery results were reported, the model is limited to particular views of an object.

Recently, PageRank [18] was used to discover regions of interest in a bounding box representation [22], and self-similarities were used to discover a common pattern in several images [1]. Although in these works no generative models were used to learn the distribution of visual objects, reliable matching and saliency are found to be helpful for object discovery. The notions of matching and saliency were also successfully applied by Fakor et al. [9], a work done in parallel to ours, for unsupervised discovery of image categories.

Co-segmentation was first introduced by Rother et al. [37], who used histogram matching to simultaneously segment the same object in two different images. Since then, numerous methods were proposed to improve and refine the co-segmentation [3, 17, 19, 34], many of which work in the context of a pair of images with the exact same object [17, 34, 37] or require some form of user interaction [3, 6].

These techniques were later extended in various ways. Joulin et al. [19] used a discriminative clustering framework that can handle multiple images, and Kim et al. [25] proposed an optimization which scales up to even larger datasets. Vicente et al. [50] introduced the notion of “objectness” to the co-segmentation framework, showing that requiring the foreground segment to be an *object* often improves co-segmentation results significantly. All these techniques, however, maintain the strong assumption that the object is present in all of the images, which is not true for Internet image collections.

Other methods were proposed to handle images which might not contain the common object, either implicitly [20] or explicitly [23]. In particular, Kim and Xing [23] show promising results given additional user input, but do not show significant improvement in the unsupervised setting. It is clear that in the context of image search and web browsing, user input cannot be used.

Co-segmentation was also explored in weakly supervised setups with multiple object categories [27, 38]. While image annotations may facilitate object discovery and segmentation, image tags are often noisy, and bounding boxes or class labels are usually unavailable. In this work we show that it is plausible to automatically discover visual objects from the Internet using image search alone.

3 Joint Inference via Dense Correspondence

In this section we describe our basic joint inference framework. We will start by defining the terminology and setup that will guide us through the rest of the chapter.

The input to our framework is a dataset $\Omega = (\mathbf{I}, \mathbf{V}, \mathbf{A})$ that is comprised of N RGB images $\mathbf{I} = \{I_1, \dots, I_N\}$, a finite vocabulary $\mathbf{V} = \{l_1, \dots, l_L\}$ of L possible *labels* each pixel can attain, and possibly additional image *annotations* $\mathbf{A} = \{\mathbf{A}_1, \dots, \mathbf{A}_N\}$. Image annotations can include a range of auxiliary information about an image. For example, it can be a collection of textual words describing the image, a time stamp specifying when the image was taken, GPS coordinates indicating where it was taken, and even pixel-level information, such as the location of faces or other objects in the image. In this chapter, we will consistently refer to semantic, pixel-level information as “labeling” and to textual, image-level annotation as “tags.” Our goal is to produce the labelings $\mathbf{C} = \{\mathbf{c}_1, \dots, \mathbf{c}_N\}$ for all the images in the dataset, where for pixel $\mathbf{x} = (x, y)$, $\mathbf{c}_i(\mathbf{x}) \in \{1, \dots, L\}$ indexes into the vocabulary \mathbf{V} . We formulate this discrete labeling problem within an optimization framework to solve for the most likely labels for all pixels in the dataset.

To exploit the dataset structure and similarity between image regions, we establish correspondences between pixels in different images. We denote by \mathbf{w}_{ij} the correspondence field—or flow field—from image I_i to image I_j , mapping each pixel in I_i to a pixel in I_j . For small datasets, we can estimate the correspondences between any pair of images, however for large datasets such computation is generally prohibitive. Therefore, we restrict the correspondences of each image I_i to a subset of the images, \mathcal{N}_i , that are most similar to it, based on *global* image statistics that are more efficient to compute. In our experiments we fixed the size of \mathcal{N}_i of each image I_i to be the same constant, K , however in general this size can be allowed to vary. Finally, we denote by \mathbf{W} the set of all pixel correspondences in the dataset: $\mathbf{W} = \cup_{i=1}^N \cup_{j \in \mathcal{N}_i} \mathbf{w}_{ij}$.

Given the input image dataset, Ω , the pixel correspondences \mathbf{W} , and additional parameters of the model, Θ (will be defined shortly), we define the cost function, $E(\mathbf{C}; \Omega, \mathbf{W}, \Theta)$ for the joint labeling \mathbf{C} , as:

$$E(\mathbf{C}; \Omega, \mathbf{W}, \Theta) = \sum_{i=1}^N \sum_{\mathbf{x} \in \Lambda_i} \left[\underbrace{\Phi^i(\mathbf{x})}_{\text{Likelihood (local evidence)}} + \underbrace{\Phi_\theta^i(\mathbf{x}, \Theta)}_{\text{Model parameters}} \right. \\ \left. + \underbrace{\sum_{\mathbf{y} \in \mathcal{N}_x^i} \lambda_{\text{int}} \Psi_{\text{int}}^i(\mathbf{x}, \mathbf{y})}_{\text{Intra-image compatibility}} + \underbrace{\sum_{j \in \mathcal{N}_i} \lambda_{\text{ext}} \Psi_{\text{ext}}^{ij}(\mathbf{x}, \mathbf{x} + \mathbf{w}_{ij}(\mathbf{x}))}_{\text{Inter-image compatibility}} \right], \quad (1)$$

where \mathcal{N}_x^i is the spatial neighbors of pixel \mathbf{x} (we use the four pixels directly connect to \mathbf{x} as the spatial neighborhood), and Λ_i is image I_i ’s lattice.

This objective function defines a (directed) graphical model over the entire image dataset (Fig. 3). The likelihood term, $\Phi^i(x)$, captures the cost of assigning

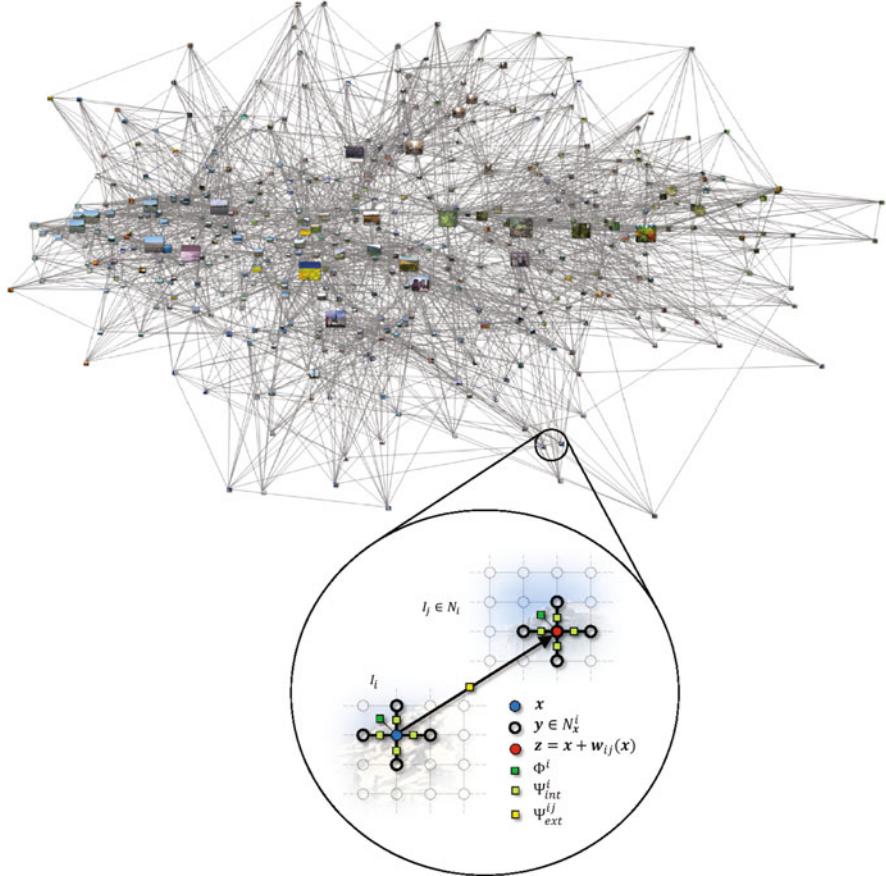


Fig. 3 The graphical model. Each pixel in the dataset is represented by a node in the graph, connected to spatially adjacent pixels in its image and corresponding pixels in similar images, which indicate statistical dependency. Connections between images are depicted by edges in the image graph on the *left* (these edges are directed, although visualized in this figure as undirected for clarity). For each such connection, dense pixel correspondences are computed, connecting each pixel in the source image to some pixel in the target image (*right*). The image graph is shown here for the LabelMe Outdoors dataset [41] using 400 sampled images and $K = 10$. The size of each image corresponds to its visual pagerank score (Sect. 4.4)

the label $\mathbf{c}_i(\mathbf{x})$ to pixel \mathbf{x} in image I_i . The definition of this term is problem-specific. Φ_θ^i is a unary energy term that accounts for additional parameters of the model. In our implementations (will be described in the upcoming sections) these parameters include per-image color models, as well as dataset-wide parameters such as spatial distribution of labels, and label co-occurrences. These parameters are estimated during the optimization, and provide a simple mechanism to account for higher-order and longer-range connections between nodes in the graph [13, 26]. The regularization terms, Ψ_{int}^i and Ψ_{ext}^{ij} , penalize discontinuous labeling *within* the

image and *between* images, respectively, subject to image structures and similarities between corresponding pixels. λ_{int} and λ_{ext} balance the contribution of these terms.

This graphical model extends traditional discrete MRF formulations used extensively in computer vision (see, e.g., [13, 30, 36, 42], and also [46] for an in-depth review) in two important ways: (a) it involves an *inter-image* compatibility term, regularizing the solution *across* images and not just within each image, and (b) it involves *all* the images in the dataset as opposed to just a single one. Optimizing this objective function collectively using all the images is key for inferring plausible pixel labels in cases where only sparse and/or noisy information is given about the images.

Equation (1) encapsulates a gigantic inference problem, and its optimization is by no means trivial. For a dataset containing 10^4 images, each of size 256×256 , there are 6.55×10^8 nodes (pixels). Each node has an order of 10^2 edges, and so there are in total 6.55×10^{10} edges in the graph! We designed an efficient parallel message passing algorithm to solve this huge graph inference problem, which will be described in the upcoming sections. The algorithm is comprised of belief propagation algorithms embedded in a coordinate descent scheme. The objective function is highly non-convex and this optimization is not guaranteed to reach the global minimum, however we show that it yields plausible solutions that improve the state of the art for the applications we explored.

In the upcoming sections, we will demonstrate how this framework can be applied to two computer vision applications: semantic labeling, and object discovery and segmentation. Each of these problems can be casted as a specific configuration of this framework, and demonstrates different aspects in which it can be utilized. For example, pixel correspondences are used only for regularization (Ψ_{ext}^{ij}) in semantic labeling, but are used for both regularization and as part of the likelihood function (Φ^i) in object discovery; the image graph is constructed once in our implementation of semantic labeling, but is iteratively updated and refined in object discovery, as the lower complexity of the latter problem allows to accommodate updates to the image graph during the optimization.

4 Application: Annotation Propagation

In this section we describe how to apply the inference framework we presented in the previous section to one specific problem—semantic segmentation of images in weakly labeled datasets. The method exploits visual similarities among the different images to help auto-annotation succeed with relatively few human-provided labels.

4.1 Formulation

Following the definitions in Sect. 3, we assume each pixel can obtain one of $L + 1$ possible labels: $\mathbf{V} = \{l_1, \dots, l_L, \emptyset\}$, where the additional label \emptyset denotes the pixel is determined to be *unlabeled*, in case it cannot be associated with any other label with sufficient confidence. Initially, we may be given annotations in the form of image tags and pixel labels for some images in the dataset, $\mathbf{A} = \{\mathbf{T}_t, \mathbf{C}_t\}$, where we denote by \mathbf{I}_t and \mathbf{T}_t , and \mathbf{I}_l and \mathbf{C}_l , the corresponding subsets of *tagged* images with their tags, and *labeled* images with their labels, respectively. The set of tags of an image is comprised of words from the vocabulary \mathbf{V} , which can be specified by a user, or obtained from text surrounding an image on a webpage. Notice that unlike traditional approaches that propagate known pixel labels to new images, here we assume most of the pixels in the dataset are unlabeled. That is, $|\mathbf{C}_l|$ is assumed to be only a small fraction of the dataset size.

Since image tags are important for image indexing and search, we also return (and evaluate in our experiments in Sect. 4.5) the tags associated with the images, $\mathbf{T} = \{\mathbf{t}_1, \dots, \mathbf{t}_N : \mathbf{t}_i \subseteq \{1, \dots, L\}\}$, which we define directly as the set union of the pixel labels in the image: $\mathbf{t}_i = \cup_{\mathbf{x} \in \mathbf{A}_i} \mathbf{c}_i(\mathbf{x})$ (ignoring unlabeled pixels).

4.1.1 Image Graph

As previously mentioned, since computing (and storing) dense pixel correspondences between every pair of images is prohibitive for large datasets, we restrict the intra-image compatibility to a set of K images that are the most similar to the image. Here, similarly to Liu et al. [30], we define the set of nearest neighbors of each image I_i , \mathcal{N}_i , as its top $\langle K, \epsilon \rangle$ similar images, where K is the maximum number of neighbors, and ϵ is a threshold on the distance between the images (above which neighbors are discarded). We use L_2 -norm between Gist descriptors [35] as the image similarity measure, although other image-level measures such as bag of words histograms or spatial pyramids can be used.¹ Some analysis of the influence of the choice of K on the performance is given in Sect. 4.5.

Once the set of neighbors for each image is determined, we use SIFT-flow [31] to compute the pixel correspondences between an image and each of its neighbors. We use the original implementations of Gist and SIFT-flow as provided by the authors, which are available online.

¹In our experiments, we did not notice significant difference in the results when computing the nearest neighbor set using pyramid matching [28] instead of Gist.

4.1.2 Objective Function Terms

Likelihood

From tags associated with the images and possibly some pixel labels (if available), we need to define the likelihood term, $\Phi_i(\mathbf{x})$, that a pixel \mathbf{x} in image I_i attains the label $l \in \mathbf{V}$. For example, an image might be tagged with *car* and *road*, but their locations within the image are unknown. We characterize this *text-to-image correspondence* by means of local visual appearance. We leverage the large number of images and the available tags to correlate the dataset vocabulary with visual statistics. We first extract local image features for every pixel, and then learn a visual appearance model for each vocabulary word by utilizing visual commonalities among images with similar tags, as well as the given pixel labels, if available. The result is an estimate of the probability distribution over the labels, $P_a(\mathbf{x})$, at each pixel \mathbf{x} . This process is described in Sect. 4.2.

We then define the likelihood directly based on this distribution estimate:

$$\Phi^i(\mathbf{x}) = -\log P_a(\mathbf{x}). \quad (2)$$

Model Parameters.

For this application this term is comprised of three components:

$$\Phi_\theta^i(\mathbf{x}, \Theta) = -\log P_t^i(\mathbf{c}_i(\mathbf{x})) - \lambda_s \log P_s(\mathbf{x}) - \lambda_c \log P_c^i(\mathbf{x}), \quad (3)$$

where $P_t^i(\mathbf{c}_i(\mathbf{x}))$ is a *tag likelihood* term that estimates the probability of image I_i having the label $\mathbf{c}_i(\mathbf{x})$ somewhere in it (and thus having l as one of its tags), and $P_s(\mathbf{c}_i(\mathbf{x}))$ and $P_c^i(\mathbf{c}_i(\mathbf{x}))$ capture the probability of the label l occurring at pixel \mathbf{x} based on its relative spatial position and color, respectively. We use superscript i in P_t^i and P_c^i to emphasize that they are estimated separately for each image, while P_s is estimated globally for the entire dataset. λ_s, λ_c balance the contribution of P_s and P_c^i , respectively.

The term P_t^i is used to bias the labels used in an image towards ones with higher frequency and co-occurrence among its neighbors. We estimate the likelihood of image I_i having the label l as

$$P_t^i(l) = \frac{\beta}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \delta[l \in \mathbf{t}_j] + \frac{1-\beta}{Z} \sum_{j \in \mathcal{N}_i} \sum_{m \in \mathbf{t}_j} \mathbf{h}_o(l, m), \quad (4)$$

where the indicator function $[\cdot]$ is 1 when its argument is true, and 0 otherwise, \mathbf{h}_o is the $L \times L$ row-normalized tag co-occurrence matrix, computed from the current tag estimates and initialized from the known tags, and $Z = \sum_{j \in \mathcal{N}_i} |\mathbf{t}_j|$. The first term in Eq. (4) measures the frequency of word l among image I_i 's neighbors, and the second term is the mean co-occurrence rate of word l within its neighbors' tags. We typically set $\beta = 0.5$, assigning equal contribution to the two terms. This term is

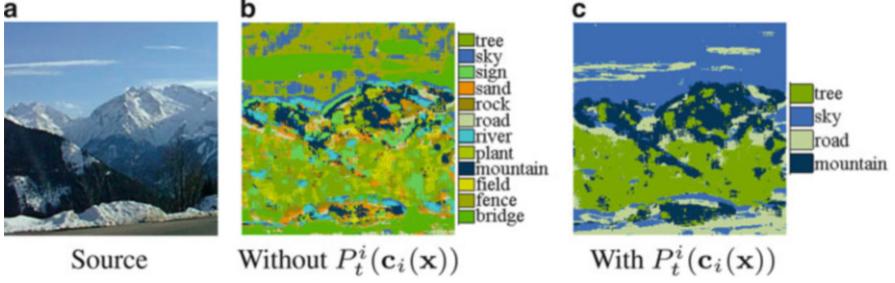


Fig. 4 The effect of tag likelihood, $P_t^i(\mathbf{c}_i(\mathbf{x}))$, on pixel classification, shown on an image from the LabelMe Outdoors dataset (see Sect. 4.5 for details on the experiment). (a) The source image. (b) MAP per-pixel classification using the learned appearance models only: $\max_{\mathbf{c}_i} P_a(\mathbf{c}_i(\mathbf{x}))$. (c) Similar to (b), with the additional tag likelihood term: $\max_{\mathbf{c}_i} \{P_a(\mathbf{c}_i(\mathbf{x})) + P_t^i(\mathbf{c}_i(\mathbf{x}))\}$. (a) Source. (b) Without tag likelihood. (c) With tag likelihood

inspired by Makadia et al. [33], but we do not set a hard threshold on the number of tags to infer for an image as they do. Figure 4 demonstrates the contribution of this term. It can be seen that when using this term we manage to obtain a much better initial guess for the labeling of the image (which will then be refined during the optimization).

Both the spatial and color terms are computed from the current pixel label estimates. The spatial location term is computed as

$$P_s(\mathbf{x}) = \mathbf{h}_s^{\mathbf{c}_i(\mathbf{x})}(\mathbf{x}), \quad (5)$$

where $\mathbf{h}_s^l(\mathbf{x})$ is the normalized spatial histogram of word l across all images in the dataset (Fig. 5). This term will assist in places where the appearance and pixel correspondence might not be as reliable.

The color term will assist in refining the labels internally within the image, and is computed as

$$P_c^i(\mathbf{x}) = \mathbf{h}_c^{i,\mathbf{c}_i(\mathbf{x})}(I_i(\mathbf{x})) \quad (6)$$

where $\mathbf{h}_c^{i,l}$ is the color histogram of word l in image I_i . We use 3D histograms of 64 bins in each of the color channels to represent $\mathbf{h}_c^{i,l}$ instead of the Gaussian mixture models used in [36] and [42].

Overall, the parameters of the model are $\Theta = \{\mathbf{h}_c^{i,l}, \mathbf{h}_s^l, \mathbf{h}_o\}$, $i = 1..N, l = 1..L$.

Regularization

The intra-image compatibility between neighboring pixels is defined based on tag co-occurrence and image structures. For image I_i and spatial neighbors $\mathbf{x}, \mathbf{y} \in \mathcal{N}_{\mathbf{x}}^i$,

$$\Psi_{\text{int}}^i(\mathbf{x}, \mathbf{y}) = -\lambda_o \log \mathbf{h}_o(\mathbf{c}_i(\mathbf{x}), \mathbf{c}_i(\mathbf{y}))$$

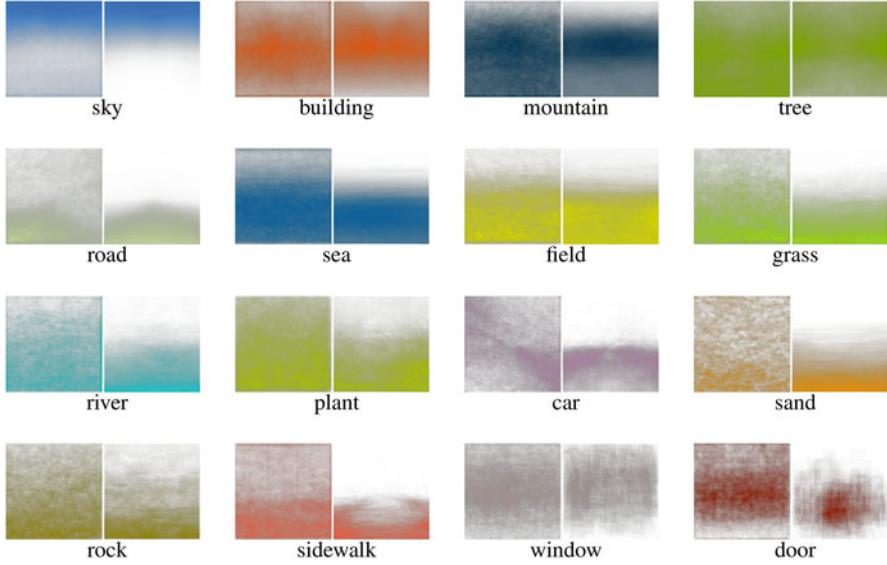


Fig. 5 An example of a model parameter, in this case the spatial distribution of words [Eq. (5)], estimated while propagating annotations. The spatial distributions are shown here for several words in the LMO dataset, ordered *from top left to bottom right* according to the frequency of the word (number of occurrences). For each word, the *left image* is the estimated spatial distribution and the *right image* is the true distribution according to human labeling. The color for each word is the average RGB value across all images according to the human labeling, with saturation corresponding to probability, from *white* (zero probability) to *saturated* (high probability)

$$+ \delta [\mathbf{c}_i(\mathbf{x}) \neq \mathbf{c}_i(\mathbf{y})] \lambda_{\text{int}} \exp(-\|I_i(\mathbf{x}) - I_i(\mathbf{y})\|_2^2). \quad (7)$$

Finally, we define the inter-image compatibility between a pixel \mathbf{x} in image I_i and its corresponding pixel $\mathbf{z} = \mathbf{x} + \mathbf{w}_{ij}(\mathbf{x})$ in image I_j as

$$\Psi_{\text{ext}}^{ij}(\mathbf{x}, \mathbf{z}) = \delta [\mathbf{c}_i(\mathbf{x}) \neq \mathbf{c}_i(\mathbf{z})] \frac{\alpha_j}{\alpha_i} \lambda_{\text{ext}} \exp(-\|S_i(\mathbf{x}) - S_j(\mathbf{z})\|_1), \quad (8)$$

where α_i, α_j are the image weights as defined in Appendix 4.2.2, and S_i are the (dense) SIFT descriptors for image I_i . Intuitively, better matching between corresponding pixels will result in higher penalty when assigning them different labels, weighted by the relative importance of the neighbor's label. Notice that SIFT features are used for the inter-image compatibility metric in Eq. (8) whereas RGB intensities are used for the intra-image compatibility in Eq. (7).

4.2 Text-to-Image Correspondence

4.2.1 Local Image Descriptors

We selected features used prevalently in object and scene recognition to characterize local image structures and color features. Structures are represented using both SIFT and HOG [7] features. We compute dense SIFT descriptors with 3 and 7 cells around a pixel to account for scales. We then compute HOG features and stack together neighboring HOG descriptors within 2×2 patches [55]. Color is represented using a 7×7 patch in L*a*b color space centered at each pixel. Stacking all the features yields a 527-dimensional descriptor $D_i(\mathbf{x})$ for every pixel \mathbf{x} in image I_i . We use PCA to reduce the descriptor to $d = 50$ dimensions, capturing approximately 80 % of the features' variance.

4.2.2 Learning Appearance Models

We use a generative model based on Gaussian mixtures to represent the distribution of the above continuous features. More specifically, we model each word in the database vocabulary using a full-covariance Gaussian Mixture Model (GMM) in the 50D descriptor space. Such models have been successfully applied in the past to model object appearance for image segmentation [8]. Note that our system is not limited to work with this particular model. In fact, we also experimented with a discriminative approach, Randomized Forests [14], previously used for semantic segmentation [43] as an alternative to GMM. We found that GMM produces better results than random forests in our system (see Sect. 4.5).

For pixel \mathbf{x} in image I_i , we define

$$P(D_i(\mathbf{x}); \Theta) = \sum_{l=1}^L \left(\rho_l \sum_{k=1}^M \pi_{l,k} \mathcal{N}(D_i(\mathbf{x}); \boldsymbol{\mu}_{l,k}, \boldsymbol{\Sigma}_{l,k}) \right) + \rho_\epsilon \mathcal{N}(D_i(\mathbf{x}); \boldsymbol{\mu}_\epsilon, \boldsymbol{\Sigma}_\epsilon), \quad (9)$$

where ρ_l is the weight of model (word) l in generating the feature $D_i(\mathbf{x})$, M is the number of components in each model ($M = 5$), and $\boldsymbol{\theta}_l = (\pi_{l,k}, \boldsymbol{\mu}_{l,k}, \boldsymbol{\Sigma}_{l,k})$ is the mixture weight, mean and covariance of component k in model l , respectively. We use a Gaussian outlier model with parameters $\boldsymbol{\theta}_\epsilon = (\boldsymbol{\mu}_\epsilon, \boldsymbol{\Sigma}_\epsilon)$ and weight ρ_ϵ . The intuition for the outlier model is to add an *unlabeled* word to the vocabulary \mathbf{V} . $\Theta = (\{\rho_l\}_{l=1:L}, \rho_\epsilon, \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_L, \boldsymbol{\theta}_\epsilon)$ is a vector containing all parameters of the model.

We optimize for Θ in the maximum likelihood sense using a standard EM algorithm. We initialize the models by partitioning the descriptors into L clusters using k-means and fitting a GMM to each cluster. The outlier model is initialized from randomly selected pixels throughout the database. We also explicitly restrict

each pixel to contribute its data to models of words corresponding to its estimated (or given) image tags only. That is, we clamp the posteriors to zero for all $l \notin \mathbf{t}_i$. For labeled images \mathbf{I}_l , we keep the posteriors fixed according to the given labels (setting zero probability to all other labels). To account for partial annotations, we introduce an additional weight α_i for all descriptors of image I_i , set to α_t , α_l or 1 (we use $\alpha_t = 5$, $\alpha_l = 10$) according to whether image I_i was tagged, labeled, or inferred automatically by the algorithm, respectively. More details can be found in the supplementary material. Given the learned model parameters, Θ , and an observed descriptor, $D_i(\mathbf{x})$, the probability of the pixel belonging to word l is computed by

$$P_a(c_i(\mathbf{x}) = l; D_i(\mathbf{x}), \Theta) = \frac{\rho_l \sum_{k=1}^M \pi_{l,k} \mathcal{N}(D_i(\mathbf{x}); \boldsymbol{\mu}_{l,k}, \boldsymbol{\Sigma}_{l,k})}{P(D_i(\mathbf{x}); \Theta)}, \quad (10)$$

where $P(D_i(\mathbf{x}); \Theta)$ is defined in Eq. (9).

Figure 6c shows an example pixel classification based on the model learned with this approach, and more results are available on the project web page.

4.3 Optimization

The optimization alternates between estimating the appearance model and propagating pixel labels. The appearance model is initialized from the images and partial annotations in the dataset. Then, we partition the message passing scheme into intra- and inter-image updates, parallelized by distributing the computation of each image to a different core. The belief propagation algorithm starts from spatial message passing (TRW-S) for each image for a few iterations, and then updates the outgoing messages from each image for several iterations. The inference algorithm iterates between message passing and estimating the color histograms in a GrabCut fashion [36], and converges in a few iterations. Once the algorithm converges, we compute the MAP labeling that determines both labels and tags for all the images.

4.4 Choosing Images to Annotate

As there is freedom to choose images to be labeled by the user, intuitively, we would want to strategically choose “image hubs” that have many similar images, since such images have many direct neighbors in the image graph to which they can propagate labels efficiently. We use visual pagerank [18] to find good images to label, again using the Gist descriptor as the image similarity measure. To make sure that images throughout the dataset are considered, we initially cluster the images and use a non-uniform damping factor in the visual rank computation (Equation 2 in [18]), assigning higher weight to the images closest to the cluster centers. Given

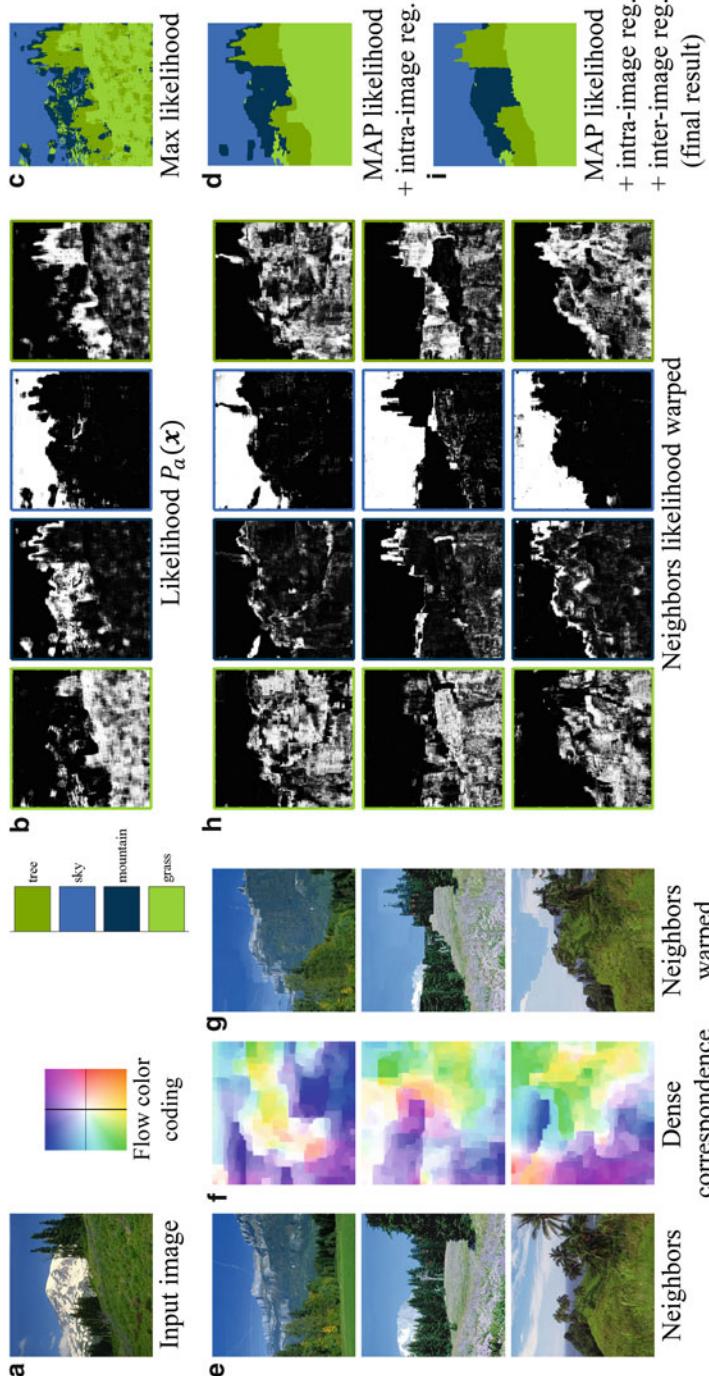


Fig. 6 Text-to-image and dense image correspondence. (a) An image from LabelMe Outdoors dataset. (b) Visualization of text-to-image likelihood P_a , for the four most probable labels, colored from black (low probability) to white (high probability). (c) The maximum likelihood classification (computed independently at each pixel). (d) The classification with spatial regularization [Eq. (7)]. (e)–(g) Nearest neighbors of the image in (a) and dense pixel correspondences with (a). (h) Same as (b), shown for each of the neighbors, warped towards the image (a) based on the computed correspondences. (i) The final MAP labeling using both intra- and inter-image regularization [Eq. (8)]



Fig. 7 Top-ranked images selected automatically for human annotation. The images are ordered from *left* (larger hub) to *right* (smaller hub). Underneath each image is its visual pagerank score (Sect. 4.4)

an annotation *budget* (r_t, r_l), where r_t denotes the percentage of images tagged in the dataset, and r_l denotes the percentage of the images labeled, we then set \mathbf{I}_l and \mathbf{I}_t as the r_l and r_t top ranked images, respectively. Figure 7 shows the top image hubs selected automatically with this approach, which nicely span the variety of scenes in the dataset.

4.5 Results

We conducted extensive experiments with the proposed method using several datasets: SUN [55] (9556 256×256 images, 522 words), LabelMe Outdoors (LMO, subset of SUN) [41] (2688 256×256 images, 33 words), the ESP game dataset [52] (21,846 images, 269 words), and IAPR benchmark [15] (19,805 images, 291 words). Since both LMO and SUN include dense human labeling, we use them to simulate human annotations for both training and evaluation. We use ESP and IAPR data as used by Makadia et al. [33]. They contain user tags but no pixel labeling.

We implemented the system using MATLAB and C++ and ran it on a small cluster of three machines with a total of 36 CPU cores. We tuned the algorithm's parameters on LMO dataset, and fixed the parameters for the rest of the experiments to the best performing setting: $\lambda_s = 1, \lambda_c = 2, \lambda_o = 2, \lambda_{\text{int}} = 60, \lambda_{\text{ext}} = 5$. In practice, five iterations are required for the algorithm to converge to a local minimum. The EM algorithm for learning the appearance model (Appendix 4.2.2) typically converge within 15 iterations, and the message passing algorithm (Sect. 4.3) converges in 50 iterations. Using $K = 16$ neighbors for each image gave the best result (Fig. 9c), and we did not notice significant change in performance for small modifications to ϵ (Sect. 4.1.1), d (Appendix 4.2.1), nor the number of GMM components in the appearance model, M [Eq. (9)]. For the aforementioned settings,

it takes the system 7 h to preprocess the LMO dataset (compute descriptors and the image graph) and 12 h to propagate annotations. The run times on SUN were 15 and 26 h, respectively.

4.5.1 Results on SUN and LMO

Figure 8a shows some annotation results on SUN using ($r_t = 0.5, r_l = 0.05$). All images shown were initially unannotated in the dataset. Our system successfully infers most of the tags in each image, and the labeling corresponds nicely to the image content. In fact, the tags and labels we obtain automatically are often remarkably accurate, considering that no information was initially available on those images. Some of the images contain regions with similar visual signatures, yet are still classified correctly due to good correspondences with other images. More results are available on the project web page.

To evaluate the results quantitatively, we compared the inferred labels and tags against human labels. We compute the global pixel recognition rate, r , and the class-average recognition rate \bar{r} for images in the subset $\mathbf{I} \setminus \mathbf{I}_t$, where the latter is used to counter the bias towards more frequent words. We evaluate tagging performance on the image set $\mathbf{I} \setminus \mathbf{I}_t$ in terms of the ratio of correctly inferred tags, P (precision), and the ratio of missing tags that were inferred, R (recall). We also compute the corresponding, unbiased class-average measures \bar{P} and \bar{R} .

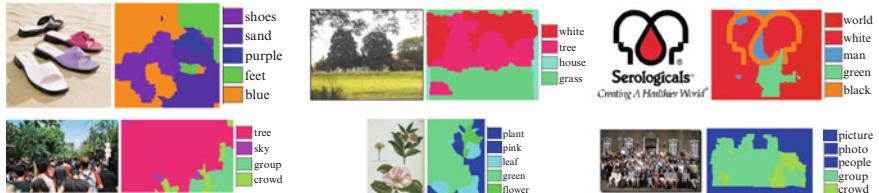
The global and class-average pixel recognition rates are 63 and 30 % on LMO, and 33 and 19 % on SUN, respectively. In Fig. 9 we show for LMO the confusion matrix and breakdown of the scores into the different words. The diagonal pattern in the confusion matrix indicates that the system recognizes correctly most of the pixels of each word, except for less frequent words such as *moon* and *cow*. The vertical patterns (e.g., in the column of *building* and *sky*) indicate a tendency to misclassify pixels into those words due to their frequent co-occurrence with other words in that dataset. From the per-class recognition plot (Fig. 9b) it is evident that the system generally performs better on words which are more frequent in the dataset.

4.5.2 Components of the Model

To evaluate the effect of each component of the objective function on the performance, we repeated the above experiment where we first enabled only the likelihood term (classifying each pixel independently according to its visual signature), and gradually added the other terms. The results are shown in Fig. 9d for varying values of r_l . Each term clearly assists in improving the result. It is also evident that image correspondences play important role in improving automatic annotation performance.

a

SUN [55] (9556 images, 522 words)

b

ESP [52] (21; 846 images, 269 words)

c

IAPR-TC12 [15] (19; 805 images, 291 words)

Fig. 8 Automatic annotation results (best viewed electronically). **(a)** SUN results were produced using ($r_t = 0.5, r_l = 0.05$) (4778 images tagged and 477 images labeled, out of 9556 images). All images shown were initially unannotated in the dataset. **(b), (c)** For ESP and IAPR, the same training set as in [33] was used, having ($r_t = 0.9, r_l = 0$). For each example we show the source image on the *left*, and the resulting labeling and tags on the *right*. The word colormap for SUN is the average pixel color based on the ground truth labels, while in ESP and IAPR each word is assigned an arbitrary unique color (since ground truth pixel labels are not provided with the datasets). Example failures are shown in the *last rows* of **(a)** and **(c)**. More results can be found in Fig. 2 and the project web page

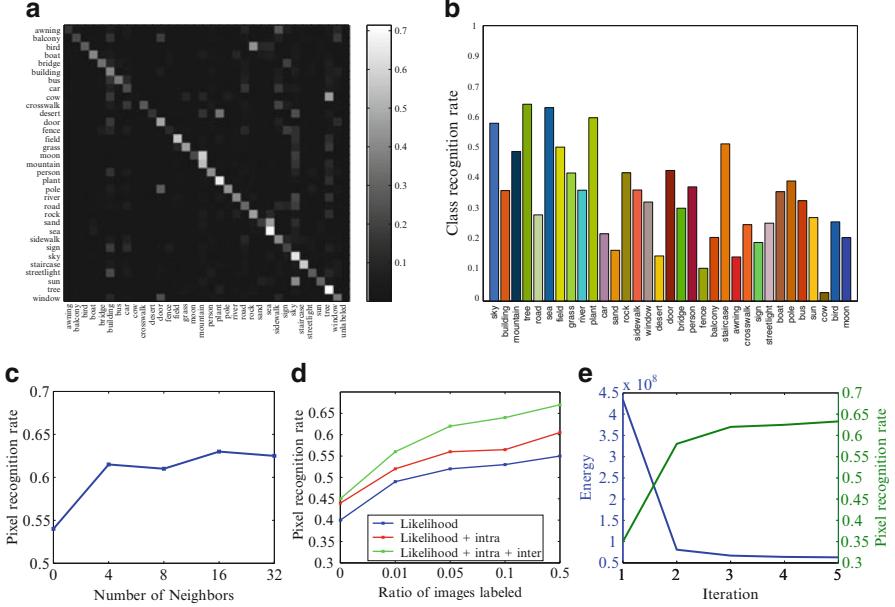


Fig. 9 Recognition rates on LMO dataset. **(a)** The pattern of confusion across the dataset vocabulary. **(b)** The per-class average recognition rate, with words ordered according to their frequency in the dataset (measured from the ground truth labels), colored as in Fig. 8. **(c)** Recognition rate as function of the number of nearest neighbors, K . **(d)** Recognition rate vs. ratio of labeled images (r_l) with different terms of the objective function enabled. **(e)** System convergence. Pixel recognition rate and the objective function energy [Eq. (1)] are shown over five iterations of the system, where one iteration is comprised of learning the appearance model and propagating pixel labels (see Sect. 4.3)

4.5.3 Comparison with State of the Art

We compared our tagging and labeling results with state of the art in semantic labeling—Semantic Texton Forests (STF) [43] and Label Transfer [30]—and image annotation (Makadia et al. [33]). We used our own implementation of [33] and the publicly available implementations of [43] and [30]. We set the parameters of all methods according to the authors’ recommendations, and used the same tagged and labeled images selected automatically by our algorithm as training set for the other methods (only tags are considered by Makadia et al. [33]). The results of this comparison are summarized in Table 1. On both datasets our algorithm shows clear improvement in both labeling and tagging results. On LMO, \bar{r} is increased by 5 % compared to STF, and \bar{P} is increased by 8 % over Makadia et al.’s baseline method. STF recall is higher, but comes at the cost of lower precision as more tags are associated on average with each image (Fig. 10). In [30], pixel recognition rate of 74.75 % is obtained on LMO at 92 % training/test split with all the training images containing dense pixel labels. However, in our more challenging

Table 1 Tagging and labeling performance on LMO and SUN datasets

	Labeling		Tagging			
	r	\bar{r}	P	\bar{P}	R	\bar{R}
Makadia [33]	—	—	53.87	30.23	60.82	25.51
STF [43]	52.83	24.9	34.21	30.56	73.83	58.67
LT [30]	53.1	24.6	41.07	35.5	44.3	19.33
AP (ours)	63.29	29.52	55.26	38.8	59.09	22.62
AP-RF	56.17	26.1	48.9	36.43	60.22	24.34
AP-NN	57.62	26.45	47.5	35.34	59.83	24.01

(a) Results on LMO

	Labeling		Tagging			
	r	\bar{r}	P	\bar{P}	R	\bar{R}
Makadia [33]	—	—	26.67	11.33	39.5	14.32
STF [43]	20.52	9.18	11.2	5.81	62.04	16.13
AP (ours)	33.29	19.21	32.25	14.1	47	13.74

(b) Results on SUN

r stands for the overall pixel recognition rate, and P and R for the precision and recall, respectively (numbers are given in percentages). The bar notation for each measure represents the per-class average (to account for class bias in the dataset). In (a), AP-RF replaces the GMM model in the annotation propagation algorithm with Random Forests [14]; AP-NN replaces SIFT-flow with nearest neighbor correspondences

(and realistic) setup, their pixel recognition rate is 53 % and their tag precision and recall are also lower than ours. Liu et al. [30] also report the recognition rate of TextonBoost [42], 52 %, on the same 92 % training/test split they used in their paper, which is significantly lower than the recognition rate we achieve, 63 %, while using only a fraction of the densely labeled images. The performance of all methods drops significantly on the more challenging SUN dataset, yet our algorithm still outperforms STF by 10 % in both \bar{r} and \bar{P} , and achieves 3 % increase in precision over [33] with similar recall.

We show qualitative comparison with STF in Fig. 10. Our algorithm generally assigns less tags per image in comparison with STF, for two reasons. First, dense correspondences are used to rule out improbable labels due to ambiguities in local visual appearance. Second, we explicitly bias towards more frequent words and word co-occurrences in Eq. (4), as those were shown to be key factors for transferring annotations [33].

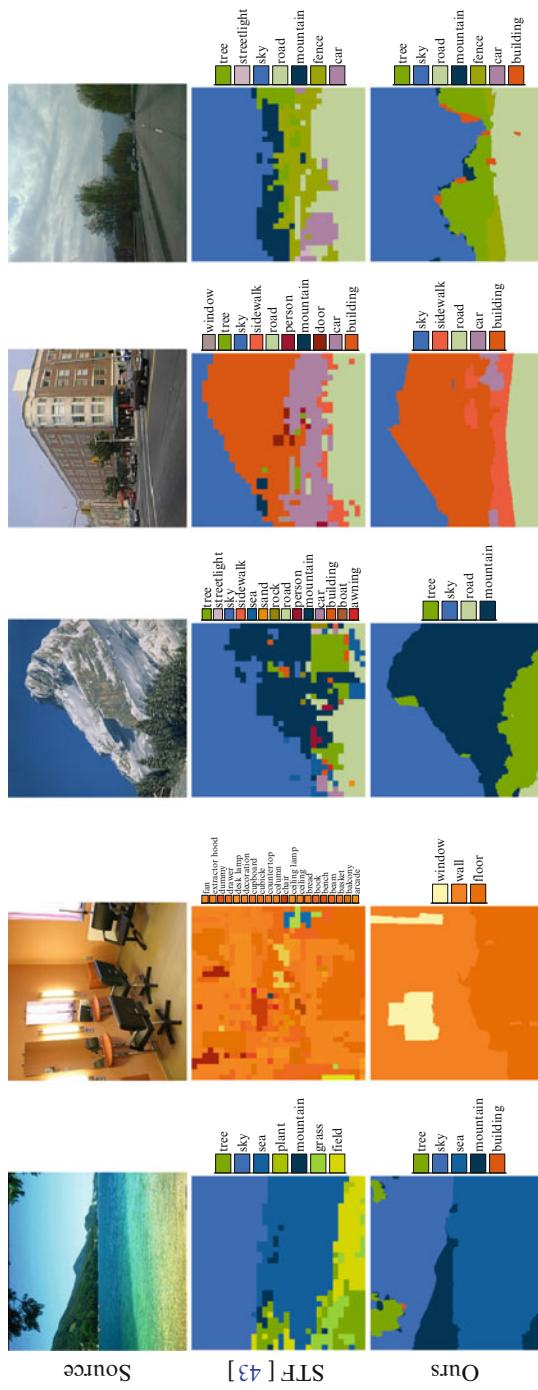


Fig. 10 Comparison with Semantic Texton Forests (STF) [43] on SUN dataset. More comparisons are available on the project web page

Table 2 Tagging performance on ESP and IAPR

	ESP				IAPR			
	P	R	\bar{P}	\bar{R}	P	R	\bar{P}	\bar{R}
Makadia [33]	22	25	—	—	28	29	—	—
AP (Ours)	24.17	23.64	20.28	13.78	27.89	25.63	19.89	12.23

P, R and \bar{P}, \bar{R} represent the global and per-class average precision and recall, respectively (numbers are in percentages). The top row is quoted from [33]

4.5.4 Results on ESP and IAPR

We also compared our tagging results with [33] on the ESP image set and IAPR benchmark using the same training set and vocabulary they used ($r_t = 0.9, r_l = 0$). Our precision (Table 2) is 2 % better than [33] on ESP, and is on par with their method on IAPR. IAPR contains many words that do not relate to particular image regions (e.g., *photo, front, range*), which do not fit our text-to-image correspondence model. Moreover, many images are tagged with colors (e.g., *white*), while the image correspondence algorithm we use emphasizes structure. Makadia et al. [33] assigns stronger contribution to color features, which seems more appropriate for this dataset. Better handling of such abstract keywords, as well as improving the quality of the image correspondences are both interesting directions for future work.

4.6 Training Set Size and Running Time

4.6.1 Training Set Size

As we are able to efficiently propagate annotations between images in our model, our method requires significantly less labeled data than previous methods (typically $r_t = 0.9$ in [33], $r_l = 0.5$ in [43]). We further investigate the performance of the system w.r.t. the training set size on SUN dataset.

We ran our system with varying values of $r_t = 0.1, 0.2, \dots, 0.9$ and $r_l = 0.1r_t$. To characterize the system performance, we use the F_1 measure, $F(r_t, r_l) = \frac{2PR}{P+R}$, with P and R the precision and recall as defined above. Following the user study by Vijayanarasimhan and Grauman [51], fully labeling an image takes 50 s on average, and we further assume that supplying tags for an image takes 20 s. Thus, for example, tagging 20 % and labeling 2 % of the images in SUN requires 13.2 human hours. The result is shown in Fig. 11. The best performance of [33], which requires roughly 42 h of human effort, can be achieved with our system with less than 20 human hours. Notice that our performance increases much more rapidly, and that both systems converge, indicating that beyond a certain point adding more annotations does not introduce new useful data to the algorithms.

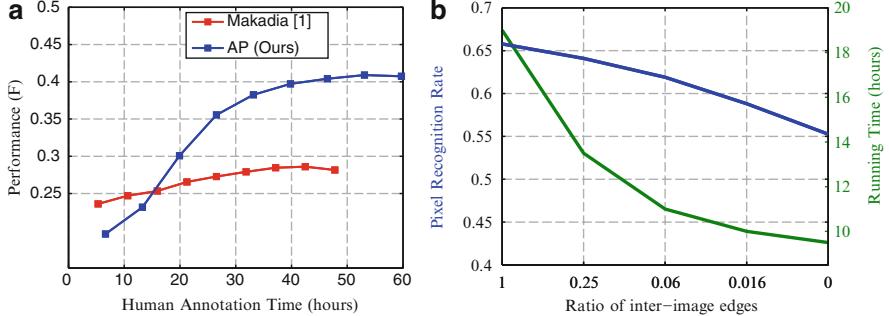


Fig. 11 (a) Performance F (F_1 measure; see text) as function of human annotation time of our method and Makadia et al. [33] on SUN dataset. (b) Performance and run time using sparse inter-image edges

4.6.2 Running Time

While it was clearly shown that dense correspondences facilitate annotation propagation, they are also one of the main sources of complexity in our model. For example, for 10^5 images, each 1 mega-pixel on average, and using $K = 16$, these add $O(10^{12})$ edges to the graph. This requires significant computation that may be prohibitive for very large image datasets. To address these issues, we have experimented with sparser inter-image connections using a simple sampling scheme. We partition each image I_i into small non-overlapping patches, and for each patch and image $j \in N_i$ we keep a single edge for the pixel with best match in I_j according to the estimated correspondence w_{ij} . Figure 11 shows the performance and running time for varying sampling rates of the inter-image edges (e.g., 0.06 corresponds to using 4×4 patches for sampling, thus using $\frac{1}{16}$ of the edges). This plot clearly shows that the running time decreases *much faster* than performance. For example, we achieve more than 30 % speedup while sacrificing 2 % accuracy by using only $\frac{1}{4}$ of the inter-image edges. Note that intra-image message passing is still performed in full pixel resolution as before, while further speedup can be achieved by running on sparser image grids.

4.7 Limitations

Some failure cases are shown in Fig. 12. Occasionally the algorithm mixes words with similar visual properties (e.g., *door* and *window*, *tree* and *plant*), or semantic overlap (e.g., *building* and *balcony*). We noticed that street and indoor scenes are generally more challenging for the algorithm. Dense alignment of arbitrary images is a challenging task, and incorrect correspondences can adversely affect the solution. In Fig. 12c we show examples of inaccurate annotation due to incorrect correspondences. More failure cases are available on the project web page.

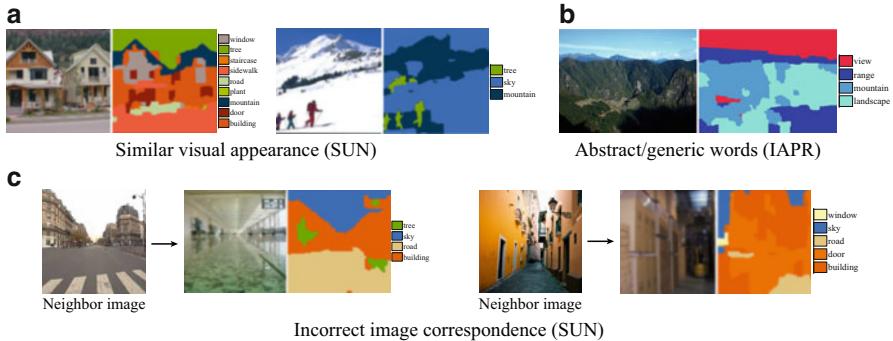


Fig. 12 Example failure cases. (a) Our system occasionally misclassifies pixels with classes that have similar visual appearance. Here, the rooftops in the left image are incorrectly labeled as “mountain”, and the people in the right image are incorrectly labeled as “tree.” (b) Generic and abstract words, such as “view,” do not fit well our model that assumes labels correspond to specific regions in the image (and that every pixel belongs to at most one class). (c) Incorrect correspondences may introduce errors. Here we show two examples of outdoor images that get connected to indoor images in the image graph, thereby leading to misinterpretation of the scene (floor gets labeled as “road,” boxes in a warehouse labeled as “building”)

5 Application: Object Discovery and Segmentation

In this section we describe how to use the framework to infer foreground regions among a set of images containing a common object. A particular use case is image datasets returned from Internet search. Such datasets vary considerably in their appearance, and typically include many noise images that do not contain the object of interest (Fig. 2). Our goal is to automatically discover and segment out the common object in the images, with no additional information on the images or the object class.

5.1 Formulation

As we want to label each pixel as either belonging or not belonging to the common object, the dataset vocabulary in this case is comprised of only two labels: $\mathbf{V} = \{\text{“background”}, \text{“foreground”}\}$. The pixel labels $\mathbf{C} = \{\mathbf{c}_1, \dots, \mathbf{c}_N\}$ are binary masks, such that $\mathbf{c}_i(\mathbf{x}) = 1$ indicates foreground (the common object), and $\mathbf{c}_i(\mathbf{x}) = 0$ indicates background (not the object) at location \mathbf{x} of image I_i . We also assume no additional information is given on the images or the object class: $\mathbf{A} = \emptyset$.

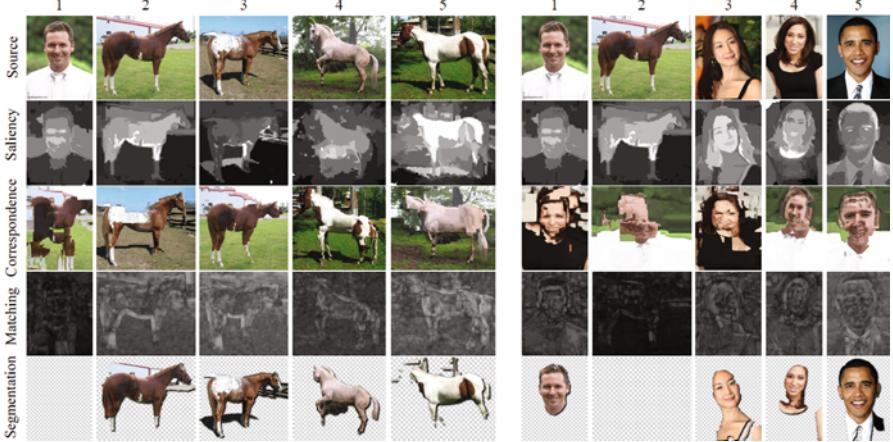


Fig. 13 One of these things is not like the others. An illustration of joint object discovery and segmentation by our algorithm on two small datasets of five images each. The images are shown at the *top row*, with two images common to the two datasets—the face and horse images in columns 1 and 2, respectively. *Left:* when adding to the two common images three images containing horses (columns 3–5), our algorithm successfully identifies *horse* as the common object and *face* as “noise,” resulting in the horses being labeled as foreground and the face being labeled as background (*bottom row*). *Right:* when adding to the two common images three images containing faces, *face* is now recognized as common and *horse* as noise, and the algorithm labels the faces as foreground and the horse as background. For each dataset, the second row shows the saliency maps, colored from *black* (less salient) to *white* (more salient); the *third row* shows the correspondences between images, illustrated by warping the nearest neighbor image to the source image; and the *fourth row* shows the matching scores based on the correspondences, colored from *black* (worse matching) to *white* (better matching)

5.1.1 Image Graph

We construct the image graph slightly differently than the way it was constructed for annotating images in the previous section. We again use SIFT flow [31] as the core algorithm to compute pixel correspondences, however instead of establishing the correspondence between all pixels in a pair of images as done before, we solve and update the correspondences based on our estimation of the foreground regions. This helps in ignoring background clutter and ultimately improves the correspondence between foreground pixels (Fig. 14), which is a key factor in separating the common object from the background and visual noise.

Formally, given (binary masks) $\mathbf{c}_i, \mathbf{c}_j$, the SIFT flow objective function becomes

$$\begin{aligned}
 E(\mathbf{w}_{ij}; \mathbf{c}_i, \mathbf{c}_j) = & \sum_{\mathbf{x} \in \Lambda_i} \mathbf{c}_i(\mathbf{x}) \left(\mathbf{c}_j(\mathbf{x} + \mathbf{w}_{ij}(\mathbf{x})) \|S_i(\mathbf{x}) - S_j(\mathbf{x} + \mathbf{w}_{ij}(\mathbf{x}))\|_1 \right. \\
 & \left. + (1 - \mathbf{c}_j(\mathbf{x} + \mathbf{w}_{ij}(\mathbf{x}))) C_0 + \sum_{\mathbf{y} \in \mathcal{N}_x^i} \alpha \|\mathbf{w}_{ij}(\mathbf{x}) - \mathbf{w}_{ij}(\mathbf{y})\|_2 \right), \quad (11)
 \end{aligned}$$

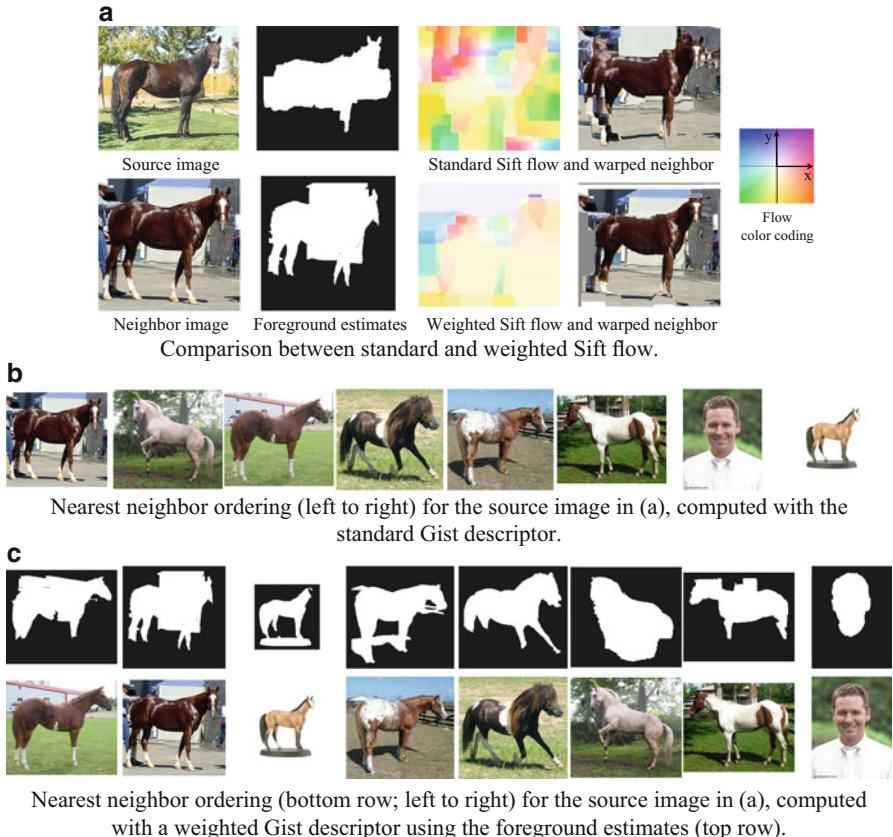


Fig. 14 Weighted Gist and Sift flow for improved image correspondence. We use the foreground mask estimates to remove background clutter when computing correspondences (**a**), and to improve the retrieval of neighbor images (compared to **(b)**, the ordering in **(c)** places right-facing horses first, followed by left-facing horses, with the (noise) image of a person last)

where \mathbf{w}_{ij} is the flow field from image I_i to image I_j , S_i are again the dense SIFT descriptors of image I_i , α weighs the smoothness term, and C_0 is a large constant.

The difference between this objective function and the original SIFT flow [31] is that it encourages matching foreground pixels in image I_i with foreground pixels in image I_j . We also use an L_2 -norm for the smoothness term instead of the truncated L_1 -norm in the original formulation [31] to make the flow more rigid in order to surface mismatches between the images. Figure 14a shows the contribution of this modification for establishing reliable correspondences between similar images.

We use the Gist descriptor [35] to find nearest neighbors, and similarly modify it to account for the foreground estimates by giving lower weight in the descriptor to pixels estimated as background. Figure 14b, c demonstrates that better sorting of the images is achieved when using this *weighted Gist* descriptor, which in turn improves the set of images with which pixel correspondences are computed.

5.1.2 Objective Function Terms

Likelihood

Our implementation is designed based on the assumption that pixels (features) belonging to the common object should be: (a) salient, i.e. dissimilar to other pixels within their image, and (b) sparse, i.e. similar to pixels (features) in other images with respect to smooth transformations between the images (e.g., with possible changes in color, size, and position). The likelihood term is thus defined in terms of the saliency of the pixel, and how well it matches to other pixels in the dataset (Fig. 13)

$$\Phi^i(\mathbf{x}) = \begin{cases} \Phi_{\text{saliency}}^i(\mathbf{x}) + \lambda_{\text{match}} \Phi_{\text{match}}^i(\mathbf{x}), & \mathbf{c}_i(\mathbf{x}) = 1, \\ \beta, & \mathbf{c}_i(\mathbf{x}) = 0, \end{cases} \quad (12)$$

where β is a constant parameter for adjusting the likelihood of background pixels. Decreasing β makes every pixel more likely to belong to the background, thus producing a more conservative estimation of the foreground.

The saliency of a pixel or a region in an image can be defined in numerous ways and extensive research in computer and human vision has been devoted to this topic. In our experiments, we used an off-the-shelf saliency measure—Cheng et al.’s Contrast-based Saliency [5]—that produced sufficiently good saliency estimates for our purposes, but our formulation is not limited to a particular saliency measure and others can be used.

Briefly, Cheng et al. [5] define the saliency of a pixel based on its color contrast to other pixels in the image (how different it is from the other pixels). Since high contrast to surrounding regions is usually a stronger evidence for saliency of a region than high contrast to far away regions, they weigh the contrast by the spatial distances in the image.

Given a saliency map, \widehat{M}_i , for each image I_i , we first compute the dataset-wide normalized saliency, M_i (with values in $[0, 1]$), and define the term

$$\Phi_{\text{saliency}}^i(\mathbf{x}) = -\log M_i(\mathbf{x}). \quad (13)$$

This term will encourage more (resp. less) salient pixels to be labeled foreground (resp. background) later on.

The matching term is defined based on the computed correspondences:

$$\widehat{\Phi}_{\text{match}}^i(\mathbf{x}) = \frac{1}{|N_i|} \sum_{j \in N_i} \|S_i(\mathbf{x}) - S_j(\mathbf{x} + \mathbf{w}_{ij}(\mathbf{x}))\|_1, \quad (14)$$

where smaller values indicate higher similarity to the corresponding pixels. Similarly to the saliency, we compute a dataset-wide normalized term (with values in $[0, 1]$), Φ_{match}^i .

Model Parameters

We additionally learn the color histograms of the background and foreground of image I_i , exactly as done in the previous section for propagating annotations [Eq. (6)]:

$$\Phi_\theta^i(\mathbf{c}_i(\mathbf{x}) = l, \Theta) = -\log \mathbf{h}_c^{i,l}(I_i(\mathbf{x})). \quad (15)$$

Here, the models parameters are comprised of only those color histograms: $\Theta = \{\mathbf{h}_c^{i,0}, \mathbf{h}_c^{i,1}\}, i = 1..N$.

Regularization

The regularization terms too are similar to the ones we defined for image annotation [Eqs. (7), (8)]. Namely, the intra-image compatibility between a pixel, \mathbf{x} , and its spatial neighbor, $\mathbf{y} \in \mathcal{N}_{\mathbf{x}}^i$, is given by

$$\Psi_{\text{int}}^i(\mathbf{x}, \mathbf{y}) = [\mathbf{c}_i(\mathbf{x}) \neq \mathbf{c}_i(\mathbf{y})] \exp \left(- \|I_i(\mathbf{x}) - I_i(\mathbf{y})\|_2^2 \right), \quad (16)$$

and the *inter-image* compatibility is defined as

$$\Psi_{\text{ext}}^{ij}(\mathbf{x}, \mathbf{z}) = [\mathbf{c}_i(\mathbf{x}) \neq \mathbf{c}_j(\mathbf{z})] \exp \left(- \|S_i(\mathbf{x}) - S_j(\mathbf{z})\|_1 \right), \quad (17)$$

where $\mathbf{z} = \mathbf{x} + \mathbf{w}_{ij}(\mathbf{x})$ in the pixel corresponding to \mathbf{x} in image I_j .

5.2 Optimization

The state space in this problem contains only two possible labels for each node: background (0) and foreground (1), which is significantly smaller than the state space for propagating annotations, whose size was in the hundreds. We can therefore optimize Eq. (1) more efficiently, and also accommodate updating the graph structure within reasonable computational time.

Hence, we alternate between optimizing the correspondences \mathbf{W} [Eq. (11)], and the binary masks \mathbf{C} [Eq. (1)]. Instead of optimizing Eq.(1) jointly over all the dataset images, we use coordinate descent that already produces good results. More specifically, at each step we optimize for a single image by fixing the segmentation masks for the rest of the images. After propagating labels from other images, we optimize each image using a Grabcut-like [36] alternation between optimizing Eq. (1) and estimating the color models $\{\mathbf{h}_c^{i,0}, \mathbf{h}_c^{i,1}\}$, as before. The algorithm then rebuilds the image graph by recomputing the neighboring images and pixel correspondences based on the current foreground estimates, and the process is repeated for a few iterations until convergence (we typically used 5–10 iterations).

5.3 Results

We conducted extensive experiments to verify our approach, both on standard co-segmentation datasets (Appendix 5.4) and image collections downloaded from the Internet (Sect. 5.5). We tuned the algorithm’s parameters manually on a small subset of the Internet images, and vary β to control the performance. Unless mentioned otherwise, we used the following parameter settings: $\lambda_{\text{match}} = 4$, $\lambda_{\text{int}} = 15$, $\lambda_{\text{ext}} = 1$, $\lambda_{\text{color}} = 2$, $\alpha = 2$, $K = 16$. Our implementation of the algorithm is comprised of distributed Matlab and C++ code, which we ran on a small cluster with 36 cores.

We present both qualitative and quantitative results, as well as comparisons with state-of-the-art co-segmentation methods on both types of datasets. Quantitative evaluation is performed against manual foreground-background segmentations that are considered as “ground truth”. We use two performance metrics: precision, P (the ratio of correctly labeled pixels, both foreground and background), and Jaccard similarity, J (the intersection over union of the result and ground truth segmentations). Both measures are commonly used for evaluation in image segmentation research. We show a sample of the results and comparisons in the paper, and refer the interested reader to many more results that we provide in the supplementary material.

5.4 Results on Co-segmentation Datasets

We report results for the MSRC dataset [42] (14 object classes; about 30 images per class) and iCoseg dataset [3] (30 classes; varying number of images per class), which have been widely used by previous work to evaluate co-segmentation performance. Both datasets include human-given segmentations that are used for the quantitative evaluation.

We ran our method on these datasets both with and without the inter-image components in our objective function (i.e., when using the parameters above, and when setting $\lambda_{\text{match}} = \lambda_{\text{ext}} = 0$, respectively), where the latter effectively reduces the method to segmenting every image independently using its saliency map and spatial regularization (combined in a Grabcut-style iterative optimization). Interestingly, we noticed that using the inter-image terms had negligible effect on the results for these datasets. Moreover, this simple algorithm—an off-the-shelf, low-level saliency measure combined with spatial regularization—which does not use co-segmentation, is sufficient to produce accurate results (and outperforms recent techniques; see below) on the standard co-segmentation datasets!

The reason is twofold: (a) all images in each visual category in those datasets contain the object of interest, and (b) for most of the images the foreground is quite easily separated from the background based on its relative saliency alone. A similar observation was recently made by Vicente et al. [50], who noticed that their *single image* classifier outperformed recent co-segmentation methods on these datasets, a

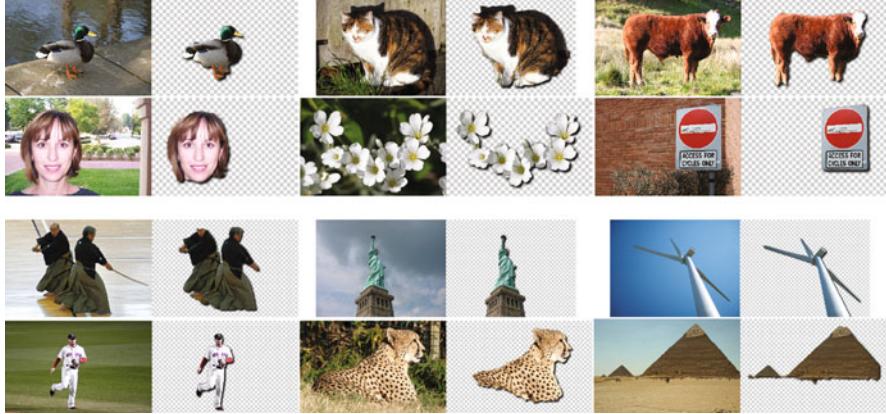


Fig. 15 Sample results on MSRC (*top two rows*) and iCoseg (*bottom two rows*). For each image we show a pair of the original (*left*) and our segmentation result (*right*). More results and qualitative comparisons with state-of-the-art are available on the project web page

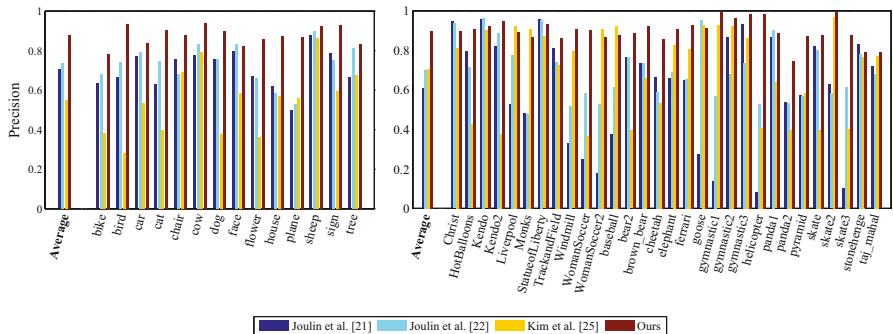


Fig. 16 Segmentation accuracy on MSRC (*left*) and iCoseg (*right*), measured as the ratio of correctly labeled pixels (both foreground and background), and compared to state-of-the-art co-segmentation methods (we performed a separate comparison with Object Cosegmentation [50]; see the text and Table 3). Each plot shows the average per-class precision on the left, followed by a breakdown of the precision for each class in the dataset

finding that is reinforced by our experiments. We thus report the results when the inter-image components are disabled. Representative results from a sample of the classes of each dataset are shown in Fig. 15.

5.4.1 Comparison with Co-segmentation Methods

We compared our results with the same three state of the art co-segmentation methods as in Sect. 5.5. The per-class precision is shown in Fig. 16 and the Jaccard similarities are available in the supplemental material on the project web page. Our

Table 3 Comparison with Object Cosegmentation [50] on MSRC and iCoseg

Method	MSRC		iCoseg	
	\bar{P}	\bar{J}	\bar{P}	\bar{J}
Vicente et al. [50]	90.2	70.6	85.34	62.04
Ours	92.16	74.7	89.6	67.63

\bar{P} and \bar{J} denote the average precision and Jaccard similarity, respectively. The per-class performance and visual results are available on the project web page

overall precision (87.66 % MSRC, 89.84 % iCoseg) shows significant improvement over [20] (73.61 % MSRC, 70.21 % iCoseg) and [25] (54.65 % MSRC, 70.41 % iCoseg).

5.4.2 Comparison with Object Cosegmentation [50]

Vicente et al.’s method [50] is currently considered state-of-the-art on these datasets.² Their code is not publicly available, however they provided us with the segmentation masks for the subsets of MSRC and iCoseg they used in their paper. We performed a separate comparison with their method using only the subset of images they used. Our method outperforms theirs on all classes in MSRC and 9/16 of the classes in iCoseg (see supplementary material), and our average precision and Jaccard similarity are slightly better than theirs (Table 3). We note that despite the incremental improvement over their method on these datasets, our results in this case were produced by segmenting each image separately using generic, low-level image cues, while their method segments the images jointly and requires training.

5.5 Results on Internet Datasets

Using the Bing API, we automatically downloaded images for three queries with query expansion through Wikipedia: *car* (4347 images), *horse* (6381 images), and *airplane* (4542 images). With $K = 16$ nearest neighbors, it took 10 h on average for the algorithm to process each dataset.

Some discovery results are shown in Fig. 17. Overall, our algorithm is able to discover visual objects despite large variation in style, color, texture, pose, scale, position, and viewing-angle. For the objects under a uniform background or with distinct colors, our method is able to output nearly perfect segmentation. Many

²Concurrently to releasing our paper, Kuettel et al. [27] managed to improve the state-of-the-art precision on the iCoseg dataset (91.4 %).

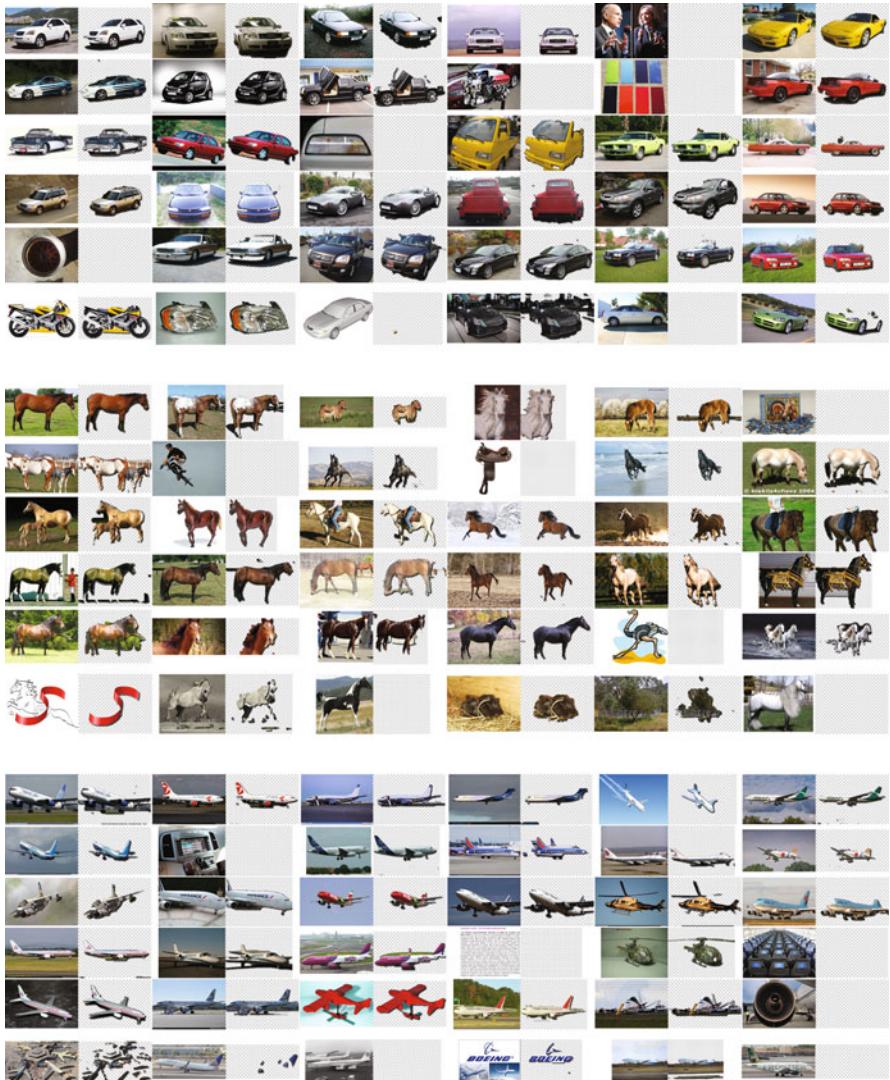


Fig. 17 Automatic discovery of cars, horses and airplanes downloaded from the Internet, containing 4347, 6381 and 4542 images, respectively. For each image, we show a pair of the original (*left*) and the segmentation result (*right*). Notice how images that do not contain the object are labeled as background. The *last row* of each dataset shows some failure cases where no object was discovered or where the discovery is wrong or incomplete. Quantitative results are available in Table 4, and more visual results can be found on the project web page

Table 4 Segmentation accuracy on the Internet datasets, with and without utilizing image correspondences

Method	Car (7.5 %)		Horse (7.8 %)		Airplane (16 %)	
	P	J	P	J	P	J
Without corr.	72.25	46.10	74.88	50.06	80.53	51.18
With corr.	83.38	63.36	83.69	53.89	86.14	55.62

Next to the name of each dataset is its percentage of noisy images (images that do not contain the object). P denotes precision and J denotes Jaccard similarity. Qualitative results for these datasets are shown in Fig. 17

objects are not very distinctive from the background in terms of color, but they were still successfully discovered due to good correspondences to other images. For *car*, some car parts are occasionally missing as they may be less salient within their image or not well aligned to other images. Similarly, for *horse*, the body of horses gets consistently discovered but sometimes legs are missing. More flexible transforms might be needed for establishing correspondences between horses. For *airplane*, saliency plays a more important role as the uniform skies always match best regardless of the transform. However the algorithm manages to correctly segment out airplanes even when they are less salient, and identifies noise images, such as that of plane cabins and jet engines, as background, since those have an overall worse matching to other images in the dataset.

For qualitative evaluation, we collected partial human labels for each dataset using the LabelMe annotation toolbox [41] and a combination of volunteers and Mechanical Turk workers, resulting in 1306 car, 879 horse, and 561 airplane images labeled. All labels were manually inspected and refined.

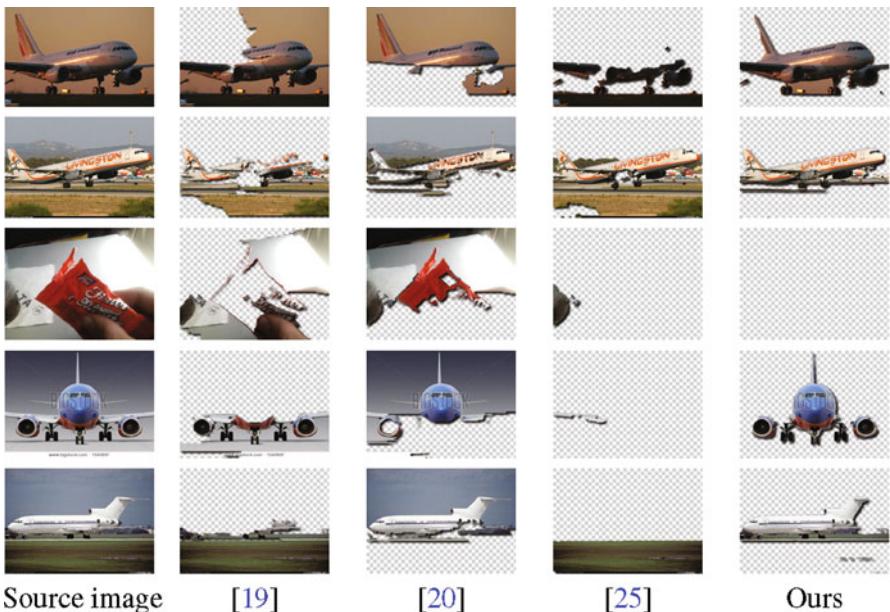
In Table 4 we show the precision and Jaccard similarity of our method on each dataset, with and without using image correspondences. The performance on airplane is slightly better than horse and car as in many of the images the airplane can be easily segmented out from the uniform sky background. Image correspondences helped the most on the *car* dataset (+11 % precision, +17 % Jaccard similarity), probably because in many of the images the cars are not that salient, while they can be matched reliably to similar car images to be segmented correctly.

5.5.1 Comparison with Co-segmentation Methods

We compare our results with three previously proposed methods [19, 20, 25]. For all three methods we used the original implementations by the authors that are publicly available, and verified we are able to reproduce the results reported in their papers when running their code. Since the competing methods do not scale to large datasets, we randomly selected 100 of the images with available ground truth labels from each dataset. We re-ran our method on these smaller datasets for

Table 5 Comparison with previous co-segmentation methods on the Internet datasets

Method	Car (11 %)		Horse (7 %)		Airplane (18 %)	
	P	J	P	J	P	J
Baseline 1	68.91	0	81.54	0	87.48	0
Baseline 2	31.09	34.93	18.46	19.85	12.52	15.26
Joulin et al. [19]	58.7	37.15	63.84	30.16	49.25	15.36
Joulin et al. [20]	59.2	35.15	64.22	29.53	47.48	11.72
Kim et al. [25]	68.85	0.04	75.12	6.43	80.2	7.9
Ours	85.38	64.42	82.81	51.65	88.04	55.81

**Fig. 18** Comparison with state-of-the-art co-segmentation methods on the *airplane* Internet dataset. More comparisons can be found on the project web page

a fair comparison. We also compared to two baselines, one where all the pixels are classified as background (“Baseline 1”), and one where all pixels are classified as foreground (“Baseline 2”). Table 5 summarizes this comparison, showing again that our method produces much better results according to both performance metrics (ours results are not exactly the same as in Table 4 bottom row, since only subsets of the full datasets are used here). The largest gain in precision by our method is on the airplane dataset, which has the highest noise level of these three datasets. Some visual comparisons are shown in Fig. 18 and more are available in the supplementary material.

5.6 Limitations

Some failures of the algorithm are shown in Fig. 17 (last row of each dataset). False positives include a motorcycle and a headlight in the *car* dataset, and a tree in the *horse* dataset. This indicates that although matching image structures often leads to object-level correspondence, exceptions occur especially when context is not taken into account.

The algorithm also fails occasionally to discover objects with unique views or background. This is because Gist is a global image descriptor, and unique view and background make it difficult to retrieve similar objects in the dataset.

Finally, our algorithm makes the implicit assumption of non-structured dataset noise. That is, repeating visual patterns are assumed to be part of some “common” object. For example, had a dataset of 100 *car* images contained 80 images of cars and 20 images of car wheels, then using $K = 16$ neighbor images by our algorithm may result in intra-group connections, relating images of cars to other images of cars and images of wheels with others alike. In such case the algorithm may not be able to infer that one category is more common than the other, and both cars and wheels would be segmented as foreground. Fortunately, the fixed setting of K we used seems to perform well in practice, however in the general case K needs to be set according to what the user considers as “common.”

6 Conclusion

We described an approach for correspondence-driven (a.k.a. example-based) computer vision under sparse training data. We utilize dense image correspondences to construct a large-scale graphical model spanning both labeled and unlabeled images, and solve it to infer pixel labels jointly in all the images by gradually propagating information from labeled images to unlabeled samples.

Our method differs from previous work in three important aspects. First, we use dense image correspondences to explicitly enforce coherent labeling across images, allowing to resolve visual ambiguities due to similar visual patterns. Second, we define an energy function over the entire dataset, and optimize it *jointly* for all the images. Third, unlike previous approaches which rely on large training sets of densely labeled images, our method can make do with significant less data by efficiently leveraging regularities and structures in the dataset.

We showed how our approach can be applied to two different computer vision problems—image annotation, and automatic visual object discovery and segmentation from the Internet—each of which can be casted as a particular configuration of our general joint inference framework. The machinery we used for solving these problems is very similar, and only requires adapting the objective function terms to the problem at hand (Table 6). We believe our framework can be similarly applied to infer depth, geometry, and edge labels, in large datasets with sparse labeled data.

Table 6 The main differences in the implementation of image annotation and object discovery using our joint inference framework

	Semantic segmentation (Sect. 4)	Object discovery (Sect. 5)
Input image annotations	Sparse tags and pixel labels	—
Vocabulary ($ V $)	Words (hundreds)	Foreground, Background (2)
Likelihood	Appearance model	Saliency + matching
Correspondences used for	Regularization	Likelihood + regularization
Image graph	Static (computed once)	Dynamic (updated iteratively)
Parameters	Per-image color model, per-class spatial dist., class co-occurrence	Per-image color model

For image annotation, our experiments show that the proposed system produces reasonable semantic labeling and tagging, and outperforms state-of-the-art methods on several large-scale image dataset while requiring significantly less human annotations. For object discovery, our method is able to naturally handle the visual variation and noise in Internet image collections, and improves upon existing co-segmentation techniques on standard co-segmentation datasets and several challenging Internet datasets.

References

1. Bagon, S., Brostovski, O., Galun, M., Irani, M.: Detecting and sketching the common. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 33–40. IEEE, San Francisco (2010)
2. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.: Patchmatch: a randomized correspondence algorithm for structural image editing. ACM Trans. Grap. **28**(3), 24 (2009)
3. Batra, D., Kowdle, A., Parikh, D., Luo, J., Chen, T.: icoseg: interactive co-segmentation with intelligent scribble guidance. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3169–3176 (2010)
4. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
5. Cheng, M.M., Zhang, G.X., Mitra, N.J., Huang, X., Hu, S.M.: Global contrast based salient region detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 409–416 (2011)
6. Collins, M.D., Xu, J., Grady, L., Singh, V.: Random walks based multi-image segmentation: Quasiconvexity results and gpu-based solutions. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1656–1663 (2012)
7. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 886–893 (2005)
8. Delong, A., Gorelick, L., Schmidt, F.R., Veksler, O., Boykov, Y.: Interactive segmentation with super-labels. In: Energy Minimization Methods in Computer Vision and Pattern Recognition, pp. 147–162 (2011)

9. Faktor, A., Irani, M.: Clustering by composition–unsupervised discovery of image categories. In: European Conference on Computer Vision (ECCV), pp. 474–487 (2012)
10. Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2008)
11. Feng, S., Manmatha, R., Lavrenko, V.: Multiple bernoulli relevance models for image and video annotation. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. II–1002 (2004)
12. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. II–264. IEEE, Madison (2003)
13. Freeman, W.T., Pasztor, E.C., Carmichael, O.T.: Learning low-level vision. *Int. J. Comput. Vis.* **40**(1), 25–47 (2000)
14. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. *Mach. Learn.* **63**(1), 3–42 (2006)
15. Grubinger, M., Clough, P., Müller, H., Deselaers, T.: The iapr tc-12 benchmark: a new evaluation resource for visual information systems. In: International Conference on Language Resources and Evaluation, pp. 13–23 (2006)
16. Heath, K., Gelfand, N., Ovsjanikov, M., Aanjaneya, M., Guibas, L.J.: Image webs: computing and exploiting connectivity in image collections. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3432–3439 (2010)
17. Hochbaum, D.S., Singh, V.: An efficient algorithm for co-segmentation. In: IEEE International Conference on Computer Vision (ICCV), pp. 269–276 (2009)
18. Jing, Y., Baluja, S.: Visualrank: applying pagerank to large-scale image search. *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(11), 1877–1890 (2008)
19. Joulin, A., Bach, F., Ponce, J.: Discriminative clustering for image co-segmentation. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, pp. 1943–1950 (2010)
20. Joulin, A., Bach, F., Ponce, J.: Multi-class cosegmentation. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 542–549 (2012)
21. Karsch, K., Liu, C., Kang, S.B.: Depth extraction from video using non-parametric sampling. In: European Conference on Computer Vision (ECCV), pp. 775–788 (2012)
22. Kim, G., Torralba, A.: Unsupervised detection of regions of interest using iterative link analysis. In: Advances in Neural Information Processing Systems (NIPS), pp. 961–969 (2009)
23. Kim, G., Xing, E.P.: On multiple foreground cosegmentation. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 837–844. IEEE, Providence (2012)
24. Kim, G., Xing, E.P.: Jointly aligning and segmenting multiple web photo streams for the inference of collective photo storylines. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 620–627 (2013)
25. Kim, G., Xing, E.P., Fei-Fei, L., Kanade, T.: Distributed cosegmentation via submodular optimization on anisotropic diffusion. In: IEEE International Conference on Computer Vision (ICCV), pp. 169–176 (2011)
26. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected crfs with gaussian edge potentials. arXiv preprint (2012) [arXiv:12105644]
27. Kuettel, D., Guillaumin, M., Ferrari, V.: Segmentation propagation in imagenet. In: European Conference on Computer Vision (ECCV), pp. 459–473 (2012)
28. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 2169–2178. IEEE, New York (2006)
29. Liang, L., Liu, C., Xu, Y.Q., Guo, B., Shum, H.Y.: Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph.* **20**(3), 127–150 (2001)
30. Liu, C., Yuen, J., Torralba, A.: Nonparametric scene parsing via label transfer. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(12), 2368–2382 (2011)

31. Liu, C., Yuen, J., Torralba, A.: Sift flow: dense correspondence across scenes and its applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(5), 978–994 (2011)
32. Liu, S., Yan, S., Zhang, T., Xu, C., Liu, J., Lu, H.: Weakly supervised graph propagation towards collective image parsing. *IEEE Trans. Multimedia* **14**(2), 361–373 (2012)
33. Makadia, A., Pavlovic, V., Kumar, S.: Baselines for image annotation. *Int. J. Comput. Vis.* **90**(1), 88–105 (2010)
34. Mukherjee, L., Singh, V., Dyer, C.R.: Half-integrality based algorithms for cosegmentation of images. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2028–2035. IEEE, Miami Beach (2009)
35. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. *Int. J. Comput. Vis.* **42**(3), 145–175 (2001)
36. Rother, C., Kolmogorov, V., Blake, A.: Grabcut: interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.* **23**(3), 309–314 (2004)
37. Rother, C., Minka, T., Blake, A., Kolmogorov, V.: Cosegmentation of image pairs by histogram matching-incorporating a global constraint into mrf. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 993–1000 (2006)
38. Rubinstein, M., Liu, C., Freeman, W.T.: Annotation propagation in large image databases via dense image correspondence. In: *European Conference on Computer Vision (ECCV)*, pp. 85–99 (2012)
39. Rubinstein, M., Joulin, A., Kopf, J., Liu, C.: Unsupervised joint object discovery and segmentation in internet images. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1939–1946 (2013)
40. Russell, B.C., Freeman, W.T., Efros, A.A., Sivic, J., Zisserman, A.: Using multiple segmentations to discover objects and their extent in image collections. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 1605–1614 (2006)
41. Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T.: Labelme: a database and web-based tool for image annotation. *Int. J. Comput. Vis.* **77**(1–3), 157–173 (2008)
42. Shotton, J., Winn, J., Rother, C., Criminisi, A.: Textonboost: joint appearance, shape and context modeling for multi-class object recognition and segmentation. In: *European Conference on Computer Vision (ECCV)*, pp. 1–15 (2006)
43. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image categorization and segmentation. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8 (2008)
44. Sivic, J., Russell, B.C., Efros, A.A., Zisserman, A., Freeman, W.T.: Discovering objects and their location in images. In: *IEEE International Conference on Computer Vision (ICCV)*, vol. 1, pp. 370–377 (2005)
45. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.* **25**(3), 835–846 (2006)
46. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE Trans. Pattern Anal. Mach. Intell.* **30**(6), 1068–1080 (2008)
47. Tappen, M.F., Liu, C.: A bayesian approach to alignment-based image hallucination. In: *European Conference on Computer Vision (ECCV)*, pp. 236–249 (2012)
48. Tighe, J., Lazebnik, S.: Superparsing: scalable nonparametric image parsing with superpixels. In: *European Conference on Computer Vision (ECCV)*, pp. 352–365 (2010)
49. Tompkin, J., Kim, K.I., Kautz, J., Theobalt, C.: Videoscapes: exploring sparse, unstructured video collections. *ACM Trans. Graph.* **31**(4):68 (2012)
50. Vicente, S., Rother, C., Kolmogorov, V.: Object cosegmentation. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2217–2224 (2011)
51. Vijayanarasimhan, S., Grauman, K.: Cost-sensitive active visual category learning. *Int. J. Comput. Vis.* **91**(1), 24–44 (2011)

52. Von Ahn, L., Dabbish, L.: Labeling images with a computer game. In: ACM Conference on Human Factors in Computing Systems (Proc. SIGCHI), pp. 319–326 (2004)
53. Wang, X.J., Zhang, L., Liu, M., Li, Y., Ma, W.Y.: Arista-image search to annotation on billions of web photos. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2987–2994 (2010)
54. Winn, J., Jojic, N.: Locus: learning object classes with unsupervised segmentation. In: IEEE International Conference on Computer Vision (ICCV), vol. 1, pp. 756–763 (2005)
55. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: large-scale scene recognition from abbey to zoo. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3485–3492 (2010)
56. Zhu, S.C., Wu, Y., Mumford, D.: Filters, random fields and maximum entropy (frame): towards a unified theory for texture modeling. *Int. J. Comput. Vis.* **27**(2), 107–126 (1998)
57. Zoran, D., Weiss, Y.: Natural images, gaussian mixtures and dead leaves. In: Advances in Neural Information Processing Systems (NIPS), pp. 1736–1744 (2012)

Dense Correspondences and Ancient Texts

**Tal Hassner, Lior Wolf, Nachum Dershowitz, Gil Sadeh,
and Daniel Stökl Ben-Ezra**

Abstract This chapter concerns applications of dense correspondences to images of a very different nature than those considered in previous chapters. Rather than images of natural or man-made scenes and objects, here, we deal with images of texts. We present a novel, dense correspondence-based approach to text image analysis instead of the more traditional approach of analysis at the character level (e.g., existing optical character recognition methods) or word level (the so called *word spotting approach*). We focus on the challenging domain of historical text image analysis. Such texts are handwritten and are often severely corrupted by noise and degradation, making them difficult to handle with existing methods. Our system is designed for the particular task of aligning such manuscript images to their transcripts. Our proposed alternative to performing this task manually is a system which directly matches the historical text image with a synthetic image rendered from the transcript. These matches are performed at the pixel level, by using SIFT flow applied to a novel per pixel representation. Our pipeline is robust to document degradation, variations between script styles and nonlinear image transformations. More importantly, this per pixel matching approach does not require prior learning of the particular script used in the documents being processed, and so can easily be applied to manuscripts of widely varying origins, languages, and characteristics.

T. Hassner (✉)

Department of Mathematics and Computer Science, The Open University of Israel, Raanana, Israel

e-mail: hassner@openu.ac.il

L. Wolf • N. Dershowitz • G. Sadeh

Tel Aviv University, Tel Aviv, Israel

e-mail: wolf@cs.tau.ac.il; nachum@cs.tau.ac.il; gils@cs.tau.ac.il

D. Stökl Ben-Ezra

École Pratique des Hautes Études, Paris, France

e-mail: daniel.stoekl@ephe.sorbonne.fr

1 Introduction

Recent large scale digitization and preservation efforts are producing vast image collections of historical manuscripts. Such manuscripts provide important records and artifacts from our shared cultural heritage. Taking European history as an example, close to one million manuscript books along with countless archival documents have survived to modern times from a period stretching back for over a millennium. Taken together, these documents serve as valuable sources of history, literature, philosophy, and scientific and medical literature, as well as art history (considering illuminations often present in such manuscripts). They are additionally important subjects of scientific inquiry in their own right, as they reflect scribal and monastic culture, the history and development of writing systems, languages and dialects, and the evolution of texts over time. Although efforts to digitally store these manuscripts provide new means for both safeguarding the information buried in them and making them widely accessible, searching through such manuscript image archives remains a challenge.

Manuscript images, unlike printed text images, can be extremely difficult to read by anyone other than experts skilled with a specific script or language type. They are often written in old languages and hence training computer systems to recognize or process them is challenging due to limited access to relevant training data. Further limiting their accessibility are abbreviations and scribal signs, commonly used by scribes. Many manuscripts have been severely degraded in quality over the years and are often corrupted by dirt and moisture. The text itself often fades, requiring specialized, sensitive imaging systems to capture. In other cases, ink may bleed through from one side of a parchment to another, making it difficult to differentiate between different texts on the same page. These and other concerns have made optical character recognition (OCR) of historical documents notoriously difficult [10].

All these challenges, as well as many more, are reflected in some of the most valuable manuscript collections, recently digitized and made available online. Some examples of these include the Dead Sea Scrolls (www.deadseascrolls.org.il), Greek papyri (www.papyrology.ox.ac.uk/Ancient_Lives), Codex Sinaiticus (codexsinaiticus.org), some of the Cairo Genizah documents (www.genizah.org), much of the Tibetan Buddhist Canon (www.tbrc.org; idp.bl.uk), Taiwanese deeds and court papers in the Taiwan History Digital Library (thdl.ntu.edu.tw), medieval Latin and English manuscripts (scriptorium.english.cam.ac.uk/manuscripts), the Early English Laws project (www.earlyenglishlaws.ac.uk), and the Papers of George Washington (rotunda.upress.virginia.edu-founders/GEWN.html) and many others. Our goal is to facilitate searching such manuscript archives, by proposing a system for the task of determining letter-by-letter mappings between transcription texts and their matching image regions in scanned manuscripts. By doing so, we provide access on a character level to these manuscript images. To our knowledge, no fully automatic method has previously been described for this task.

In previous chapters of this book, dense correspondences were used to transfer semantic information (e.g., depth, segmentation, scene labels) from reference examples to query images. Here, the same approach is used to transfer character labels from a synthetically rendered reference image to the manuscript image. We describe a system which is general in the sense that it does not attempt to learn how to identify graphemes in the manuscript. By using a robust “optical-flow” technique to directly match the pixels of the historical image with a synthetic image created from the text, it also avoids the assumption that one is provided with letter segmentation, a challenge in itself, particularly in images of historical texts. Instead, by transferring the (known) letter labels of pixels in the reference image to those in the historical document image, the extents of each letter (i.e., its segmentation) are obtained as part of our output.

The capabilities of our system were tested on images of a number of manuscripts from a range of scripts, writing directions, writing styles, and languages. As a possible extension to our method, we discuss how manual corrections of false correspondences can be used to improve the correspondence estimation quality from one line to the next. Beyond the specific application discussed here, our method provides an idea of the potential capabilities of a per pixel, correspondence-based approach in handling even extremely challenging text images.

This chapter is based on work which has previously been published in [11, 30].

2 Previous Work

Although previous work exists on the subject of matching text with images of the same text [13, 34], this problem has received very little attention compared to related problems of automatic text processing. In broad terms, existing systems take one of the following general approaches to this problem. OCR can be applied to the manuscript image in order to automatically extract the text appearing in it. Following character recognition, alignment of text, and image is a straightforward task. Although a number of effective, commercial quality OCR systems exist, applying them to manuscript images is well known to be extremely challenging (see, e.g., [3, 5, 28, 36] and many others).

One of the problems faced by OCR systems when applied to historical documents is the challenge of segmenting individual letters in texts. To avoid this problem, some have proposed systems which learn to recognize entire words rather than single characters. Some such examples include the systems of [15, 22] and more recently [2]. These systems all involve training on a collection of example word images. These are not available in many cases, particularly when examples from the same script and scribal hand do not exist.

Like us, some attempt to align text lines in their entirety. In [16, 19], as well as in other works, a sequence of word images and the transcript are treated as time series. Dynamic time warping (DTW) is used to align the sequence of word images

and its transcript. Alternatively, hidden Markov models (HMM) have been used to represent these sequences and then align them in [7, 29, 44]. These methods too require sufficient examples of similar texts in order to produce models tuned to the manuscripts being processed.

Character sizes, inter-character gaps, and other text punctuation characteristics have been modeled using geometric models, in order to improve the quality of the text segmentation, thereby providing better alignment quality. This approach has been applied, for example, on Japanese texts in [43] and Chinese texts in [40]. These methods are typically designed for a specific language and character properties, and it is not clear how to generalize them from one language to another, where these properties change.

The methods above are all fully automatic. A different approach which has been gaining in popularity is to offer manuscript scholars convenient software systems for transcript alignment. Several such tools include the text–image linking environment (TILE) [33], the UVic image markup tool project [14], and the TextGrid system [21]. Though these software tools can be of immense help towards cataloging the text in manuscript images, applying them to the huge collections of digitized manuscript images may be too labor intensive.

In contrast to the methods mentioned above, ours does not use expert knowledge of the properties of the language or typeface used by its characters. Furthermore, our method does not require training data to adapt to a manuscript type. Instead, we use direct, image-to-image, per pixel matching techniques between the manuscript image and a rendered image produced using a font sufficiently similar to the one used in the text. Though the quality of our alignments can depend on the suitability of this user-selected font, our system is designed to be robust even in cases when it is only a rough approximation to the typeface used by the scribe when writing the manuscript. Finally, compared to the software tools described above, our system is designed to be applied automatically, though it can be extended to allow for manual corrections. Moreover, when processing long manuscripts, images, manual corrections can be used to learn how to better estimate correspondences and improve alignment from one manuscript line to the next.

3 Method Overview

Our pipeline is illustrated in Fig. 1. For an input manuscript image and a text file containing its line-by-line transcript, our system performs the following steps. We begin processing the manuscript image by applying the line detection method of Wolf et al. [39] in order to obtain individual lines of text in the image. This method first binarizes the manuscript image and then projects the binary values onto the vertical axis. Peaks are then detected on these projected values in order to localize each line. Lines are then individually trimmed at their horizontal boundaries, detected by projecting the binary values of each line onto the horizontal axis (Fig. 1,

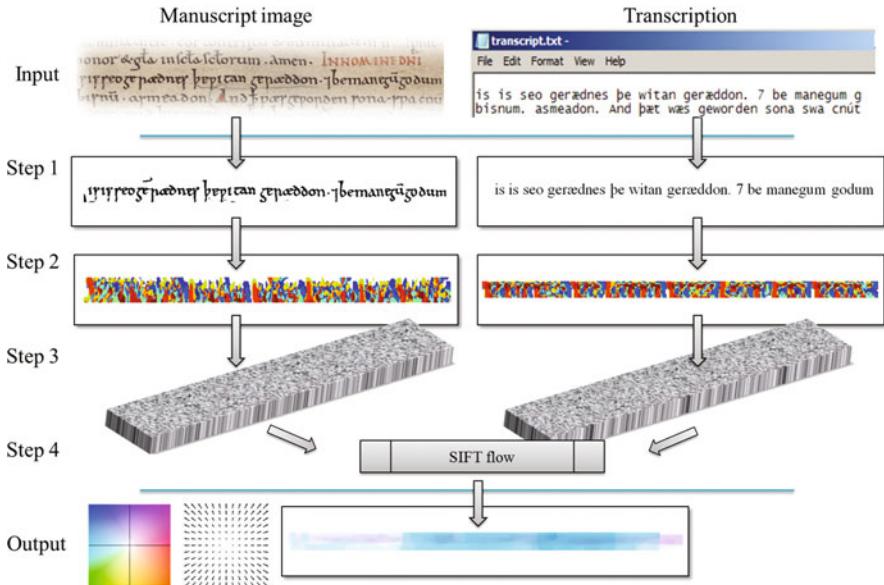


Fig. 1 The stages of our transcript alignment method. *Top row*: Our input is a historical manuscript image (*left*), along with a line-by-line transcript of the text in the image (*right*). Step 1, *left*: Each manuscript line is horizontally projected and trimmed; *right*, each matching line of the transcript is rendered using a suitable typeface, producing a matching reference image. Step 2: FPLBP codes [37] are computed for the pixels of each image (codes color coded). Step 3: Each pixel is represented by the frequency histogram of the codes in its elongated neighborhood. Step 4: Dense correspondences are established between these two histogram matrices, using SIFT flow [23]. *Bottom row*: the output flow (shown here color coded) matches each pixel in the manuscript line with a pixel in the rendered, reference image, providing access to its known character label

Step 1, left). The output of this step can then be manually adjusted using a graphical user interface designed for this purpose.

Once lines are isolated, our problem becomes that of assigning transcript character labels to each of the pixels of ink and of inter-word spaces in the line. To this end we render the matching transcript text line, producing a synthetic, reference image of the text (Fig. 1, Step 1, right). This is performed using a suitable font chosen for the purpose. Font selection is performed in order to produce a reference image in the same script and style (e.g., cursive, and print). In practice, the reference image fonts are not expected to be identical to those in the manuscript and, as we later show, can often be very different.

We maintain the spatial location of each character, as it is rendered onto the reference image. We therefore have a per pixel character label for all the reference image pixels. These labels are illustrated in Fig. 2 where a zoomed-in view of the reference image is shown above the pixel labels, visualized as colors. The transition from colder colors to warmer colors reflects the indices to the letters of the transcript; colder colors represent lower indices and warmer colors, higher indices.



Fig. 2 Reference image and matching character labels. Focusing on Fig. 1, Step 1, right: A transcript line is rendered to produce a reference image along with a per pixel index to the characters in the original transcript. Index values are visualized here by *colors*. Note that spaces between words are also assigned with indices, matching them to the spaces appearing in the original transcript. Please see text for more information

Note in particular that spaces and punctuation marks are also labeled with indices to transcript letters.

Step 2 of our pipeline (Fig. 1) converts both the binarized manuscript image and the synthetic reference image to Four-Patch LBP (FPLBP) code images C and C' , respectively. These FPLBP codes [37] are discussed in detail below (Sect. 4). This transformation converts each pixel to an integer value in the range [0..15]. In Step 3 of our pipeline, both code images are used to produce dense, per pixel descriptors for both images. This is accomplished by replacing the code assigned to each image with a histogram of the codes in the immediate neighborhood around the pixel. In order to capture the horizontal ambiguity of horizontal scripts, these histograms are computed using a vertical ellipse as their spatial support.

Finally, in Step 4 we use a robust dense correspondence estimation method, the SIFT flow of Liu et al. [23] in order to match the pixels of the two images. Where SIFT flow originally used Dense-SIFT (DSIFT) descriptors [35] to represent pixels, we replace it with the FPLBP code histograms produced in Step 3. Correspondences computed by SIFT flow from manuscript image to the reference image allow transferring the per pixel character labels of the reference back onto the manuscript, thereby providing the alignment from manuscript pixels to transcript letters. Key steps of our method are detailed in the following sections.

4 Pixel Encoding

4.1 Local Binary Patterns and Their Variants

Local binary patterns (LBP) [25–27] were originally designed as texture descriptors characterized by an invariance to monotonic photometric changes. Recent years have shown these representations to be highly effective in domains other than texture recognition, particularly in face recognition tasks[1]. The success of LBP motivated the development of LBP variants, which were later applied to a range of additional applications, including object localization [42] and action recognition [17]. The original LBP codes are computed at each pixel location by applying a threshold over the 3×3 neighborhood around the pixel, using the central pixel's intensity value as the threshold value. The resulting pattern of eight bits is then treated as a binary string and stored as an 8-bit number. An entire image is then represented by counting occurrences of every LBP code, in nonoverlapping regions of the image.

4.2 Four-Patch LBP Codes

We employ an LBP variant called Four-Patch LBP (FPLBP) [37]. Its design was motivated by a previous LBP variant called the center-symmetric LBP (CSLBP) [12]. CSLBP compares four pairs of intensities arranged in a circle around a central pixel. These four comparisons are represented as binary values reflecting which of the two pixels in each pair has the higher intensity value. FPLBP combines a similar circular sampling strategy, with the patch based approach of another LBP variant, Multi-Block LBP [41], which provides better spatial support for each comparison by sampling square patches, rather than pixels, and comparing their mean values.

Though motivated by these previous methods, FPLBP is a very different representation. Rather than encoding the relation between pixel or pixel-patch intensities, it uses short binary strings to encode local self-similarities [32] of pixel patches. FPLBP and the related Three-Patch LBP codes (TPLBP) [37] have been shown to capture valuable local information which complements the information reflected by the pixel-based descriptors, including the original LBP codes [38].

FPLBP encoding is performed by considering two concentric rings around each pixel. Each ring is sampled at S evenly distributed $w \times w$ pixel patches. Two opposing pairs consisting of an inner ring patch and an outer ring patch, separated by α patches between them, are then compared by evaluating the L2 distances between the patches in each pair. A single bit in the representation is set according to which of these two pairs holds more similar patches (i.e., has a smaller L2 distance between its two patches). The central pixel is then assigned with the $S/2$ bits resulting from these comparisons.

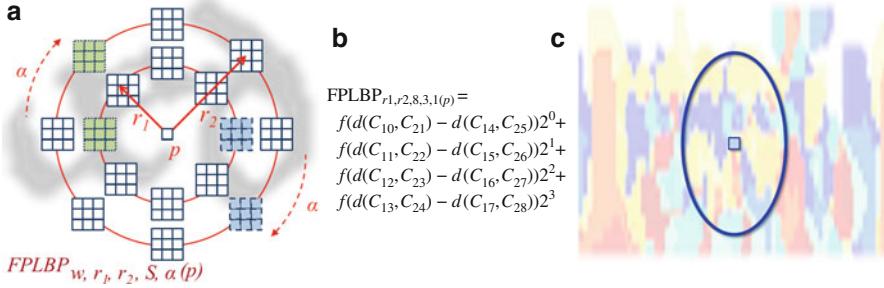


Fig. 3 (a) Visualizing how a single bit in the FPLBP code of pixel p is set (Step 2 of our pipeline, Fig. 1). Two patch pairs are compared: one in *dotted lines* and *green fill* and the other in *dashed lines* and *blue fill*. Patches in each pair are separated by $\alpha = 1$ patches. (b) Formal expression for FPLBP codes with parameters $S = 8$, $w = 3$, and $\alpha = 1$. The C_{ij} denote the various 3×3 patches; the first subscript indicates the ring (inner or outer) and the second denotes their location on the ring, from index 0 at twelve o'clock. (c) Visualizing the vertical ellipse support region used to construct the histogram of FPLBP codes (visualized in *washed out colors*) at each pixel

This entire process is summarized in Fig. 3a, b. It shows the standard case of using $S = 8$ patches, resulting in only four bits per pixel. Despite this small size, these codes have been shown to be extremely effective representations. In [8], for example, FPLBP codes were used for face recognition and were shown to perform nearly as well as the significantly more expensive collections of SIFT [24] descriptors.

4.3 Why FPLBP?

Our choice of using the FPLBP codes here requires justification, especially in comparison to the standard DSIFT [35] typically used with SIFT flow [23] or alternative LBP code variants.

The Three-Patch LBP (TPLBP), presented with the FPLBP codes in [37], is also a patch-based local binary pattern descriptor in which a central patch is compared to a pair of patches in a manner designed to capture local self-similarities. It sets the value of a code bit to 1 following a comparison of the similarity of a central patch to two patches, α patches away from each other on a ring around pixel p . Eight patches are distributed on this ring, resulting in eight comparisons and an eight-bit code or a number in the range [0..255].

Previous work has shown that TPLBP captures more information than FPLBP and produces superior representations when used for face recognition [37] and texture recognition [6]. Our experiments with both these descriptors have shown that this advantage is minor, providing only slightly better performance than SIFT flow using DSIFT. The original LBP codes perform worst than both. This is unsurprising, when considering that LBP is computed by comparing pixel pairs, rather than

patches, making them more sensitive to noise—a major problem when considering degraded manuscript images.

FPLBP, by comparison, has several appealing properties, making it ideal for the task considered here. First, it is substantially smaller than any of the other representations. As we will show next, this property is significant when extracting descriptors on a dense grid; other representations require storage and processing time that can quickly become unreasonable. Second, and more importantly, when computed for document images, the range of codes produced by TPLBP (and also LBP) is only partially represented in the codes computed in practice. This is due to the far more restrictive nature of such images, which results in a smaller local pattern variations. Thus, the comparisons performed by FPLBP of local appearances between pairs in left/right, top/down, and both diagonals suffice to capture meaningful information.

These observations are verified empirically in Fig. 4. It compares the variability of code values computed by TPLBP vs. FPLBP. Evidently, the TPLBP histogram is far more sparse than its FPLBP counterpart. It therefore uses different values more efficiently in order to capture appearance variations.

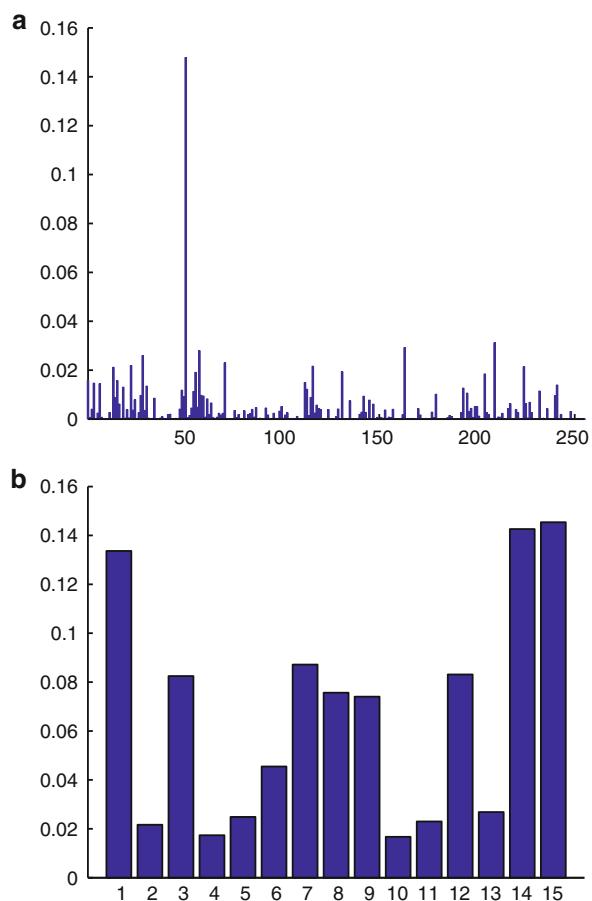
4.4 From Codes to Dense Descriptors

Both manuscript and reference images are converted to code images C and C' respectively (Sect. 4.2). We next convert the codes in these images to dense, per pixel representations (Fig. 1, Step 3). This is performed in a manner similar to the *explode* operator used by the distribution fields representation for object tracking in [31]. Specifically, in order to account for local displacements of pixel codes, each pixel is represented by the histogram of code frequencies in its immediate neighborhood.

When considering horizontal scripts, the uncertainty of code appearances is greater horizontally than vertically. This is because letters tend to preserve their vertical appearance (similar image features would likely appear above or below each other), whereas, the variability of letters appearing before or after an image location implies greater horizontal feature variability. To account for this, our histograms are produced by considering a vertical ellipse support region (Fig. 3c). Of course, when considering vertical scripts, this may conceivably be changed to reflect vertical uncertainty, though we did not perform such tests here. Additional uncertainty in code values is introduced by weighing the influence of codes further away from the central pixel. In practice, we use a 2D Gaussian filter, with sigmas 2.5 (horizontally) and 1 (vertically), for this purpose.

We produce the dense, per pixel representation from FPLBP code images as follows. First, each FPLBP code is converted to a 16D binary vector, with the value 1 in the cell indexed by the code itself. We then consider the 16 binary maps, representing each dimension of these binary vectors, taken over the entire image. Each such map is processed by applying the elongated Gaussian filter

Fig. 4 (a) A histogram of TPLBP codes computed for a set of document images. (b) FPLBP code histogram computed for the same images, demonstrating a far more uniform distribution of code values



described above. This is performed efficiently by using the fast, integral image-based approximation to Gaussian filtering, described in [20]. Each pixel is then represented by its values following this filtering.

Given the two histogram fields produced by the process described above, we form dense correspondences between them using SIFT flow as a matching engine [23], applied to our own representation. Previous chapters describe SIFT flow in detail. Unlike other related methods (e.g., PatchMatch [4] and its variants [9, 18]), it explicitly seeks smooth correspondences with small displacements. These requirements reflect a desire to match similar regions of the manuscript image to the reference image and is therefore ideally suited for our purposes. Note that its complexity is dominated by the size of the per-pixel representation. Here, by using the FPLBP, the resulting representation is 16D, substantially smaller than other representations, allowing for faster processing.

5 Experiments

Experiments were performed to evaluate the effectiveness of our approach both qualitatively and quantitatively. Quantitative tests were performed using a synthetic data set produced in order to obtain an exact measure of the accuracy of our method, when applied to a wide range of varying fonts. Qualitative tests were performed on real data—historical documents from a wide range of sources.

5.1 Empirical Tests

We produced a synthetic data set from the Charles Dickens book, *A Tale of Two Cities*. We rendered a page from the book using all 274 standard fonts installed on our MS-Windows PC. This large collection of font types was then used to test the robustness of our method to differences between appearances of text in manuscript and reference images. With each rendered text we produced also the reference, ground truth character labels, similar to those visualized in Fig. 2. We scaled all fonts to the height of 19 pixels and the resulting image was at a resolution of 1140×716 pixels. The average width of a character was about nine pixels.

Our tests used a single reference image, which was the one produced using the Times New Roman font. All other fonts, excluding non-English graphemes such as Webdings, Wingdings, and Euclid Math, were used to represent manuscript images. In total, 269 query images were tested, each one providing 50 lines of text. These were processed automatically as described in Sect. 3.

Accuracy was empirically evaluated by considering the x, y pixel coordinates of the center of each letter. We measure the distance between the center of each manuscript image letter to the position of the corresponding reference letter following warp. We note that this error measure cannot be zero, even when perfect alignment is produced, since the center of mass of each character varies depending on the font. Still, this value is expected to decrease as the alignment quality improves. We report the mean (\pm SD) distance over all letters in a document as our alignment error rate.

Our results are summarized in Table 1. Besides the mean and standard deviation of the per document distances, we provide the median distance as well as the percent of manuscript fonts for which each method obtained the best results (smallest distances). We compare the following methods: A baseline method, in which text lines are linearly stretched or shrunk to match their extreme ends from reference to manuscript; the original SIFT flow using DSIFT as the per pixel representation; finally, our proposed method. Though we have also tested LBP-based and TPLBP-based representations, neither one was competitive with those reported here and so are excluded from the table.

We visualize our empirical tests in Fig. 5. It shows four example lines aligned using our method. All these examples were produced using the same reference font

Table 1 Summary of empirical results

Method	Baseline	SIFT flow	Proposed
Mean error \pm SD	10.21 ± 4.2	8.38 ± 3.8	6.18 ± 3.1
Median error	11.23	7.42	5.27
Percent best error (%)	6	17	77

Each manuscript font is matched to the reference font using three tested methods. Accuracy was measured by considering the mean displacement error between letter centroids. The table reports mean (\pm SD) distance per method over all 269 fonts tested. Also provided are median displacement errors and the percentage of fonts for which each method performs best compared to all others. As a baseline we applied linear stretching or shrinking of the reference text to match the horizontal dimensions of the tested text

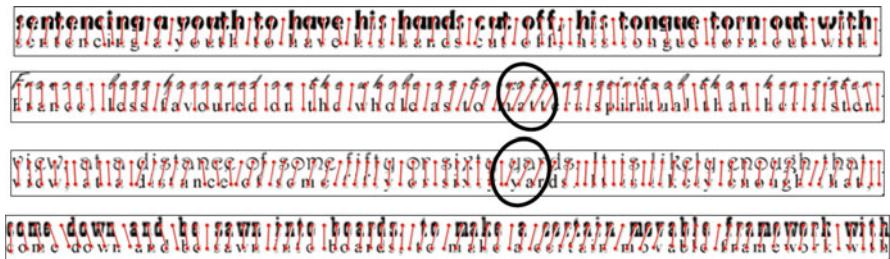


Fig. 5 Empirical tests visualized. Four examples from our empirical tests, produced using our system. Each example shows the synthetic (test) manuscript font on *top* and the reference at the *bottom*. Lines connect the center of each reference font character to its estimated location in the test font. Mistaken alignments are *highlighted* in the two middle examples. Note that by establishing per pixel correspondences, we obtain alignments for the spaces between letters as well as for punctuation marks

(bottom) and different test fonts, representing the manuscript. Lines connect the centers of reference characters to their estimated locations in the test lines of text. Evidently, for the most part, alignment succeeds, despite the differences in fonts used. Noteworthy are the links established between spaces and punctuation marks, and not just the letters themselves. We highlight a few alignment mistakes in the two middle examples.

5.2 Qualitative Results

We used our system to align manuscripts from a variety of sources, representing different languages and writing characteristics. Many such results are reported in [11].

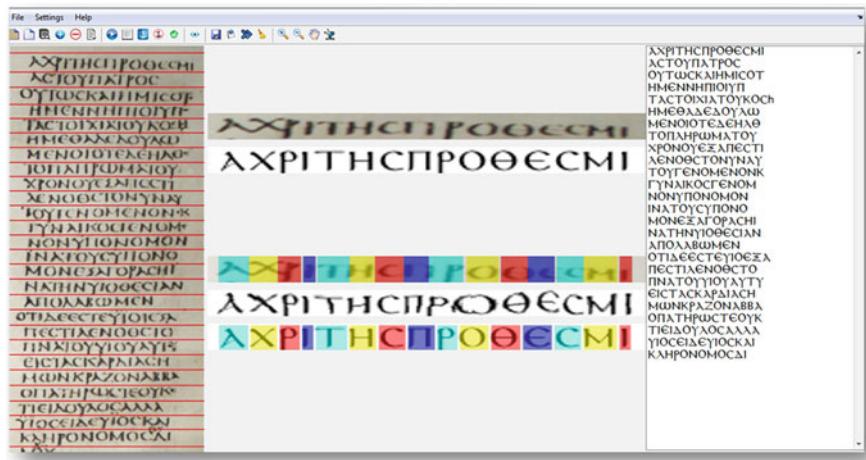


Fig. 6 Graphic user interface for transcript alignment. A page from the Codex Sinaiticus following processing of the first line using our graphical user interface. *Left panel:* The manuscript image. Horizontal lines denote automatically detected text rows. *Middle panel:* Alignment result shown for the current (first) line of text. *From top to bottom* are the current manuscript line; the rendered reference image produced for the manuscript line; the current manuscript line, color coded according to the character labels assigned through the correspondences; the synthetic reference image, warped according to the correspondences from manuscript to reference; finally, the synthetic reference image, color coded according to character labels. *Right panel:* Synthetically produced reference lines for the entire manuscript page

We have developed a graphical user interface in order to provide convenient means of loading manuscript and transcript images, as well as present alignment results. Beyond row-by-row application of the method described in this chapter, the interface allows human operators to correct alignment errors. Our system uses these corrections to train a letter spotting mechanism that provides manuscript-specific cues for alignment, improving the quality of alignment, and reducing the required manual labor, from one line to the next. These novel features are described in depth in [30].

A screen shot of our interface is provided in Fig. 6. It shows an alignment result for the first row of a page from Codex Sinaiticus. Alignment results are visualized in two ways. First, by warping the synthetic reference image, according to the inferred correspondences. Here, good results should warp the reference to match the appearance of the manuscript line, and in particular, reference letters should appear beneath their corresponding manuscript lines. Second, reference character labels are shown, color coded, over their matched manuscript image regions. The final result of aligning all text lines of the same page is provided in Fig. 7. An additional example in Hebrew MS Kaufmann A50 of the Mishnah is presented in Fig. 8.

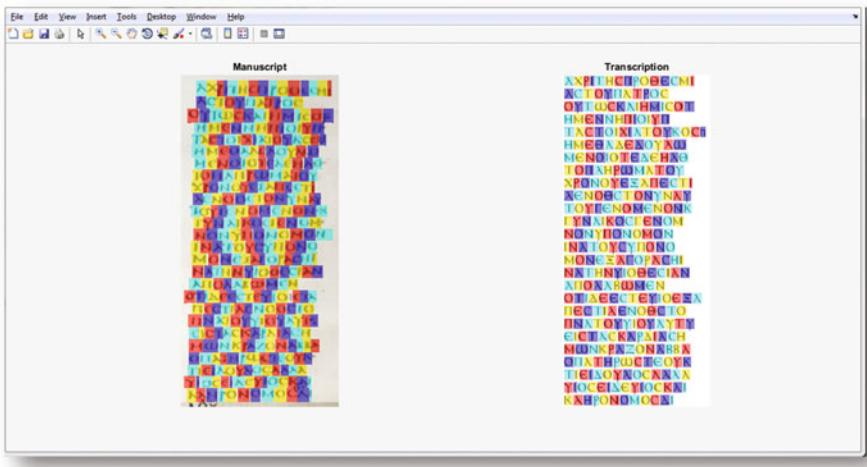


Fig. 7 Alignment result for an entire page from the Codex Sinaiticus. *Left panel:* The manuscript image with per character labels visualized by the graphical user interface as colors overlaid on image regions corresponding to each character. *Right panel:* Synthetically rendered reference rows with their per character labels visualized as colors overlaid on each character. Good results show the same color assigned to a transcript character on the right and to its corresponding manuscript character (image region) on the left

6 Conclusion

This chapter presents a dense correspondence-based solution to the very specific problem of transcript alignment. Despite the ostensibly narrow scope of the problem considered here, the method we present suggests far wider prospects for correspondence-based approaches in processing text images. Specifically, it marks a change from processing text documents on a per character or per word level to a per pixel level. It further demonstrates that with a suitable per pixel representation and a robust correspondence estimation engine, even challenging texts may be automatically processed. It remains to be seen if a similar approach can be applied to more traditional text processing applications. Unlike the settings considered here, these often do not have the privilege of a known transcript to match with the text image. Rather than relying on transcripts, future extensions can instead assume a fixed (possibly very large) vocabulary of known words and abbreviations using them as potential references.

Acknowledgements MS Kaufmann A50 by courtesy of the Oriental Collection of the Library and Information Centre of the Hungarian Academy of Sciences. This research was initiated at the Dagstuhl Perspectives Workshop 12382, “Computation and Palaeography: Potentials and Limits” and seminar 14302 on “Digital Palaeography: New Machines and Old Texts.”

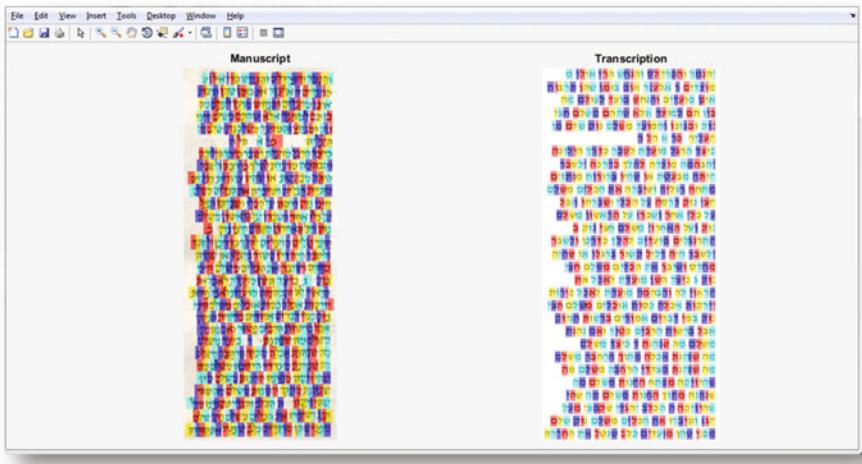


Fig. 8 Alignment result for an entire page from the MS Kaufmann A50 with the manuscript image (*left*) and the synthetically rendered characters of the transcription (*right*). Corresponding characters in the transcript and in the manuscript are labeled with similar color boxes

References

1. Ahonen, T., Hadid, A., Pietikainen, M.: Face description with local binary patterns: application to face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(12), 2037–2041 (2006)
2. Al Azawi, M., Liwicki, M., Breuel, T.M.: WFST-based ground truth alignment for difficult historical documents with text modification and layout variations. In: IS&T/SPIE Electronic Imaging, International Society for Optics and Photonics (2013)
3. Asi, A., Rabaev, I., Kedem, K., El-Sana, J.: User-assisted alignment of arabic historical manuscripts. In: Proceedings of Workshop on Historical Document Imaging and Processing, pp. 22–28. ACM, New York (2011)
4. Barnes, C., Shechtman, E., Goldman, D.B., Finkelstein, A.: The generalized PatchMatch correspondence algorithm. In: Proceedings of ECCV (2010)
5. Dovgalecs, V., Burnett, A., Tranouez, P., Nicolas, S., Heutte, L.: Spot it! Finding words and patterns in historical documents. In: Proceedings of International Conference on Document Analysis and Recognition, pp. 1039–1043. IEEE, New York (2013)
6. Ebert, S., Larlus, D., Schiele, B.: Extracting structures in image collections for object recognition. In: Proceedings of ECCV (2010)
7. Fischer, A., Frinken, V., Fornés, A., Bunke, H.: Transcription alignment of Latin manuscripts using hidden Markov models. In: Proceedings of HIP (2011)
8. Guillaumin, M., Verbeek, J., Schmid, C., Lear, I., Kuntzmann, L.: Is that you? Metric learning approaches for face identification. In: Proceedings of ICCV (2009)
9. HaCohen, Y., Shechtman, E., Goldman, D.B., Lischinski, D.: Non-rigid dense correspondence with applications for image enhancement. *ACM Trans. Graph.* **30**(4), 70:1–70:9 (2011)
10. Hassner, T., Rehbein, M., Stokes, P.A., Wolf, L.: Computation and palaeography: potentials and limits. *Dagstuhl Manifestos* **2**(1), 14–35 (2013)
11. Hassner, T., Wolf, L., Dershowitz, N.: OCR-free transcript alignment. In: Proceedings of International Conference on Document Analysis and Recognition, pp. 1310–1314 (2013)

12. Heikkilä, M., Pietikäinen, M., Schmid, C.: Description of interest regions with center-symmetric local binary patterns. In: Indian Conference Computer Vision, Graphics and Image Processing (2006)
13. Hobby, J.D.: Matching document images with ground truth. *Int. J. Doc. Anal. Recognit.* **1**(1), 52–61 (1998)
14. Holmes, M.: The UVic image markup tool project. Available: http://tapor.uvic.ca/~mholmes/image_markup (2008)
15. Huang, C., Srihari, S.N.: Mapping transcripts to handwritten text. In: Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition, pp. 15–20 (2006)
16. Jose, D., Bhardwaj, A., Govindaraju, V.: Transcript mapping for handwritten English documents. In: Yanikoglu, B.A., Berkner, K. (eds.) DRR, SPIE Proceedings, vol. 6815, SPIE (2008)
17. Kellokumpu, V., Zhao, G., Pietikainen, M.: Human activity recognition using a dynamic texture based method. In: Proceedings of BMVC (2008)
18. Korman, S., Avidan, S.: Coherency sensitive hashing. In: Proceedings of the IEEE International Conference on Computer Vision (2011)
19. Kornfield, E.M., Manmatha, R., Allan, J.: Text alignment with handwritten documents. In: Proceedings of Document Image Analysis for Libraries (DIAL), pp. 195–211. IEEE Computer Society, Cambridge (2004)
20. Kovesi, P.: Fast almost-Gaussian filtering. In: Proceedings of International Conference on Digital Image Computing: Techniques and Applications, pp. 121–125 (2010)
21. Kuster, M., Ludwig, C., Al-Hajj, Y., Selig, T.: Textgrid provenance tools for digital humanities ecosystems. In: Proceedings of Conference on Digital Ecosystems and Technologies Conference, pp. 317–323. IEEE, New York (2011)
22. Lavrenko, V., Rath, T.M., Manmatha, R.: Holistic word recognition for handwritten historical documents. In: Proceedings of Document Image Analysis for Libraries (DIAL), pp. 278–287 (2004)
23. Liu, C., Yuen, J., Torralba, A.: Sift flow: dense correspondence across scenes and its applications. *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(5), 978–994 (2011)
24. Lowe, D.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **60**(2), 91–110 (2004)
25. Ojala, T., Pietikäinen, M., Harwood, D.: A comparative-study of texture measures with classification based on feature distributions. *Pattern Recognition* **29**(1), 51–59 (1996)
26. Ojala, T., Pietikäinen, M., Mäenpää, T.: A generalized local binary pattern operator for multiresolution gray scale and rotation invariant texture classification. In: Proceedings of ICAPR (2001)
27. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Anal. Mach. Intell.* **24**(7), 971–987 (2002)
28. Rabaev, I., Biller, O., El-Sana, J., Kedem, K., Dinstein, I.: Case study in Hebrew character searching. In: Proceedings of International Conference on Document Analysis and Recognition, pp. 1080–1084. IEEE, New York (2011)
29. Rothfeder, J.L., Manmatha, R., Rath, T.M.: Aligning transcripts to automatically segmented handwritten manuscripts. In: Bunke, H., Spitz, A.L. (eds.) Document Analysis Systems. Lecture Notes in Computer Science, vol. 3872, pp. 84–95. Springer, Berlin (2006)
30. Sadeh, G., Wolf, L., Hassner, T., Dershowitz, N., Ben-Ezra, D.S., Ben-Ezra Stökl, D.: Viral transcription alignment. In: Proceedings of International Conference on Document Analysis and Recognition (2015)
31. Sevilla-Lara, L., Learned-Miller, E.: Distribution fields for tracking. In: Proceedings of CVPR (2012)
32. Shechtman, E., Irani, M.: Matching local self-similarities across images and videos. In: Proceedings of CVPR, pp. 1–8 (2007). doi:10.1109/CVPR.2007.383198
33. Terras, M., Cayless, H., Noel, W.: Text-image linking environment (TILE). Available: <http://mith.umd.edu/tile> (2009)

34. Tomai, C.I., Zhang, B., Govindaraju, V.: Transcript mapping for historic handwritten document images. In: *Frontiers in Handwriting Recognition*, pp. 413–418 (2002)
35. Vedaldi, A., Fulkerson, B.: VLFeat: an open and portable library of computer vision algorithms. In: *Proceedings of International Conference on Multimedia*, pp. 1469–1472 (2010)
36. Wei, H., Gao, G.: A keyword retrieval system for historical Mongolian document images. *Int. J. Doc. Anal. Recognit.* **17**(1), 33–45 (2014)
37. Wolf, L., Hassner, T., Taigman, Y.: Descriptor based methods in the wild. In: *Post-ECCV Faces in Real-Life Images Workshop* (2008)
38. Wolf, L., Hassner, T., Taigman, Y.: Effective unconstrained face recognition by combining multiple descriptors and learned background statistics. *Trans. Pattern Anal. Mach. Intell.* **33**(10), 1978–1990 (2011)
39. Wolf, L., Littman, R., Mayer, N., German, T., Dershowitz, N., Shweka, R., Choueka, Y.: Identifying join candidates in the Cairo Genizah. *Int. J. Comput. Vis.* **94**(1), 118–135 (2011)
40. Yin, F., Wang, Q.F., Liu, C.L.: Integrating geometric context for text alignment of handwritten Chinese documents. In: *Proceedings of International Conference on Frontiers in Handwriting Recognition*, pp. 7–12. IEEE, New York (2010)
41. Zhang, L., Chu, R., Xiang, S., Liao, S., Li, S.: Face detection based on multi-block LBP representation. In: *IAPR/IEEE International Conference on Biometrics* (2007)
42. Zhang, J., Huang, K., Yu, Y., Tan, T.: Boosted local structured HOG-LBP for object localization. In: *Proceedings of CVPR*, pp. 1393–1400 (2011)
43. Zhu, B., Nakagawa, M.: Online handwritten Japanese text recognition by improving segmentation quality. In: *Proceedings of 11th International Conference on Frontiers in Handwriting Recognition*, Montreal, pp. 379–384 (2008)
44. Zimmermann, M., Bunke, H.: Automatic segmentation of the IAM off-line database for handwritten English text. In: *Proceedings of ICPR*, vol. 4, pp. 35–39 (2002)