# Deformable Spatial Pyramid Matching for Fast Dense Correspondences

Jaechul Kim[1] Ce Liu[2] Fei Sha[3] Kristen Grauman[1]
Univ. of Texas at Austin[1] Microsoft Research New England[2] Univ. of Southern California[3]
{jaechul,grauman}@cs.utexas.edu celiu@microsoft.com feisha@usc.edu

## Abstract

*We introduce a fast deformable spatial pyramid (DSP) matching algorithm for computing dense pixel correspondences. Dense matching methods typically enforce both appearance agreement between matched pixels as well as geometric smoothness between neighboring pixels. Whereas the prevailing approaches operate at the pixel level, we propose a pyramid graph model that simultaneously regularizes match consistency at multiple spatial extents—ranging from an entire image, to coarse grid cells, to every single pixel. This novel regularization substantially improves pixel-level matching in the face of challenging image variations, while the "deformable" aspect of our model overcomes the strict rigidity of traditional spatial pyramids. Results on LabelMe and Caltech show our approach outperforms state-of-the-art methods (SIFT Flow [15] and PatchMatch [2]), both in terms of accuracy and run time.*

## 1. Introduction

Matching all the pixels between two images is a long-standing research problem in computer vision. Traditional dense matching problems—such as stereo or optical flow—deal with the "instance matching" scenario, in which the two input images contain different viewpoints of the same scene or object. More recently, researchers have pushed the boundaries of dense matching to estimate correspondences between images with *different* scenes or objects. This advance beyond instance matching leads to many interesting new applications, such as semantic image segmentation [15], image completion [2], image classification [11], and video depth estimation [10].

There are two major challenges when matching generic images: image variation and computational cost. Compared to instances, different scenes and objects undergo much more severe variations in appearance, shape, and background clutter. These variations can easily confuse low-level matching functions. At the same time, the search space is much larger, since generic image matching permits no clean geometric constraints. Without any prior knowledge on the images' spatial layout, in principle we must search every pixel to find the correct match.

To address these challenges, existing methods have largely focused on imposing geometric regularization on the matching problem. Typically, this entails a smoothness constraint preferring that nearby pixels in one image get matched to nearby locations in the second image; such constraints help resolve ambiguities that are common if matching with pixel appearance alone. If enforced in a naive way, however, they become overly costly to compute. Thus, researchers have explored various computationally efficient solutions, including hierarchical optimization [15], randomized search [2], 1D approximations of 2D layout [11], spectral relaxations [13], and approximate graph matching [5].

Despite the variety in the details of prior dense matching methods, we see that their underlying models are surprisingly similar: minimize the appearance matching cost of individual pixels while imposing geometric smoothness between paired pixels. That is, existing matching objectives center around *pixels*. While sufficient for instances (e.g., MRF stereo matching [17]), the locality of pixels is problematic for generic image matching; pixels simply lack the discriminating power to resolve matching ambiguity in the face of visual variations. Moreover, the computational cost for dense pixels remains a bottleneck for scalability.

To address these limitations, we introduce a *deformable spatial pyramid* (DSP) model for fast dense matching. Rather than reason with pixels alone, the proposed model regularizes match consistency at multiple spatial extents—ranging from an entire image, to coarse grid cells, to every single pixel. A key idea behind our approach is to strike a balance between robustness to image variations on the one hand, and accurate localization of pixel correspondences on the other. We achieve this balance through a pyramid graph: larger spatial nodes offer greater regularization when appearance matches are ambiguous, while smaller spatial nodes help localize matches with fine detail. Furthermore, our model naturally leads to an efficient hierarchical optimization procedure.

To validate our idea, we compare against state-of-the-art methods on two datasets, reporting results for pixel label transfer and semantic segmentation tasks. Compared to today's strongest and most widely used methods, SIFT Flow [15] and PatchMatch [2]—both of which rely on a pixel-based model—our method achieves substantial gains in matching accuracy. At the same time, it is noticeably faster, thanks to our coarse-to-fine optimization and other implementation choices.

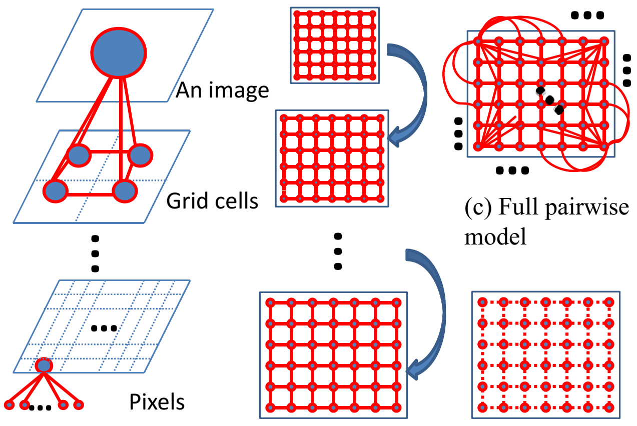
## 2. Background and Related Work

We review related work on dense matching, and explain how prior objectives differ from ours.

Traditional matching approaches aim to estimate very accurate pixel correspondences (e.g., sub-pixel error for stereo matching), given two images of the same scene with slight viewpoint changes. For such accurate localization, most methods define the matching cost on pixels. In particular, the pixel-level Markov random field (MRF) model, combined with powerful optimization techniques like graph-cut or belief propagation, has become the *de facto* standard. It casts matching as a graph optimization problem, where pixels are nodes, and edges between neighboring nodes reflect the existence of spatial constraints between them [4]. The objective consists of a data term for each pixel's matching cost and a smoothness term for the neighbors' locations.

Unlike traditional instance matching, recent work attempts to densely match images containing different scenes. In this setting, the intra-class variation across images is often problematic (e.g., imagine computing dense matches between a sedan and a convertible). Stronger geometric regularization is one way to overcome the matching ambiguity—for example, by enforcing geometric smoothness on all pairs of pixels, not just neighbors [3, 13] (see Fig. 1(c)). However, the increased number of pairwise connections makes them too costly for dense pixel-level correspondences, and more importantly, they lack the multi-scale regularization we propose.

The SIFT Flow algorithm pioneered the idea of dense correspondences across different scenes [15]. For efficiency, it uses a multi-resolution image pyramid together with a hierarchical optimization technique inspired by classic flow algorithms. At first glance it might look similar to our spatial pyramid, but in fact its objective is quite different. SIFT Flow relies on the conventional pixel-level MRF model: each pixel defines a node, and graphs from different resolutions are treated independently. That is, no graph edges span between pyramid levels. Although pixels from different resolution levels cover different spatial extents, they still span sub-image (local) regions even at the coarsest resolution. In contrast, our model explicitly addresses both global (e.g., an entire image) and local (e.g., a pixel) spatial extents, and nodes are linked between pyramid levels in the graph. Compare Figures 1(a) and (b). In



(a) DSP (Ours) (b) SIFT Flow (d) PatchMatch

Figure 1. Graph representations of different matching models. A circle denotes a graph node and its size represents its spatial extent. Edges denote geometric distortion terms. **(a)** *Deformable spatial pyramid* (proposed): uses spatial support at various extents. **(b)** *Hierarchical pixel model* [15]: the matching result from a lower resolution image guides the matching in the next resolution. **(c)** *Full pairwise model* [3, 13]: every pair of nodes is linked for strong geometric regularization (though limited to sparse nodes). **(d)** *Pixel model with implicit smoothness* [2]: geometric smoothness is enforced in an indirect manner via a spatially-constrained correspondence search (dotted lines denote no explicit links). Aside from the proposed model (a), all graphs are defined on a pixel grid.

addition, SIFT Flow defines the matching cost at each pixel node by a single SIFT descriptor at a given (downsampled) resolution, which risks losing useful visual detail. In contrast, we define the matching cost of each node using *multiple* descriptors computed at the image's *original* resolution, thus preserving richer visual information.

The PatchMatch algorithm computes fast dense correspondences using a randomized search technique [2]. For efficiency, it abandons the usual global optimization that enforces explicit smoothness on neighboring pixels. Instead, it progressively searches for correspondences; a reliable match at one pixel subsequently guides the matching locations of its nearby pixels, thus implicitly enforcing geometric smoothness. See Figure 1(d).

Despite the variations in graph connectivity, computation techniques, and/or problem domains, all of the above approaches share a common basis: a flat, pixel-level objective. The appearance matching cost is defined at each pixel, and geometric smoothness is imposed between paired pixels. In contrast, the proposed deformable spatial pyramid model considers both matching costs and geometric regularization within multiple spatial extents. We show that this substantial structure change has dramatic impact on the accuracy and speed of dense matching.

Rigid spatial pyramids are well-known in image classification, where histograms of visual words are often compared using a series of successively coarser grid cells at fixed locations in the images [12, 20]. Aside from our focus on dense matching (vs. recognition), our work differs substantially from the familiar spatial pyramid, since we model geometric distortions between and across pyramid levels in the matching objective. In that sense, our matching relates to deformable part models in object detection [7] and scene classification [16]. Whereas all these models use a few tens of patches/parts and target object recognition, our model handles millions of pixels and targets dense pixel matching.

The use of local and global spatial support for image alignment has also been explored for mosaics [18] or layered stereo [1]. For such instance matching problems, however, it does not provide a clear win over pixel models in practice. In contrast, we show it yields substantial gains when matching generic images of different scenes, and our regular pyramid structure enables an efficient solution.

## 3. Approach

We first define our deformable spatial pyramid (DSP) graph for dense pixel matching (Sec. 3.1). Then, we define the matching objective we will optimize on that pyramid (Sec. 3.2). Finally, we discuss technical issues, focusing on efficient computation (Sec. 3.3).

### 3.1. Pyramid Graph Model

To build our spatial pyramid, we start from the entire image and divide it into four rectangular grid cells and keep dividing until we reach the predefined number of pyramid levels (we use 3). This is a conventional spatial pyramid as seen in previous work. However, in addition to those three levels, we further add one more layer, a pixel-level layer, such that the finest cells are one pixel in width.

Then, we represent the pyramid with a graph. See Figures 1 (a) and 2. Each grid cell and pixel is a node, and edges link all neighboring nodes within the same level, as well as parent-child nodes across adjacent levels. For the pixel level, however, we do not link neighboring pixels; each pixel is linked only to its parent cell. This saves us a lot of edge connections that would otherwise dominate run-time during optimization.

### 3.2. Matching Objective

Now, we define our matching objective for the proposed pyramid graph. We start with a basic formulation for matching images at a single fixed scale, and then extend it to multi-scale matching.

**Fixed-Scale Matching Objective** Let $\mathbf{p}_i = (x_i, y_i)$ denote the location of node $i$ in the pyramid graph, which is given by the node's center coordinate. Let $\mathbf{t}_i = (u_i, v_i)$ be the translation of node $i$ from the first to the second image. We want to find the optimal translations of each node in the first image to match it to the second image, by minimizing the energy function:

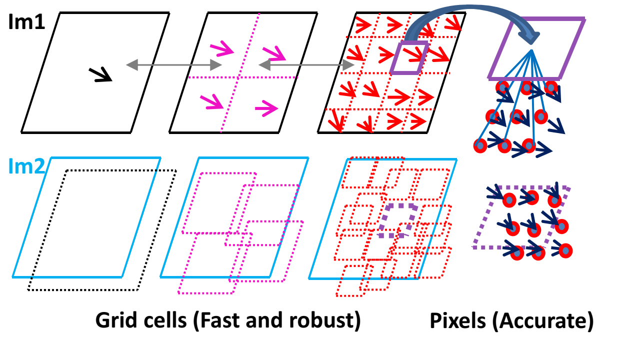


Figure 2. Sketch of our DSP matching method. First row shows image 1's pyramid graph; second row shows the match solution on image 2. Single-sided arrow in a node denotes its flow vector $\mathbf{t}_i$; double-sided arrows between pyramid levels imply parent-child connections between them (intra-level edges are also used but not displayed). We solve the matching problem at different sizes of spatial nodes in two layers. Cells in the grid-layer (left three images) provide reliable (yet fast) initial correspondences that are robust to image variations due to their larger spatial support. Guided by the grid-layer initial solution, we efficiently find accurate pixel-level correspondences (rightmost image). Best viewed in color.

$$E(\mathbf{t}) = \sum_i D_i(\mathbf{t}_i) + \alpha \sum_{i,j \in \mathcal{N}} V_{ij}(\mathbf{t}_i, \mathbf{t}_j), \qquad (1)$$

where $D_i$ is a data term, $V_{ij}$ is a smoothness term, $\alpha$ is a constant weight, and $\mathcal{N}$ denotes pairs of nodes linked by graph edges. Recall that edges span across pyramid levels, as well as within pyramid levels.

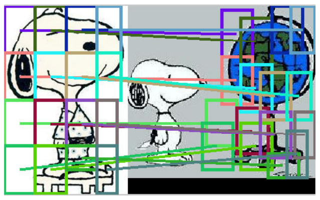
Our data term $D_i$ measures the appearance matching cost of node $i$ at translation $\mathbf{t}_i$. It is defined as the average distance between local descriptors (e.g, SIFT) within node $i$ in the first image to those located within a region of the same scale in the second image after shifting by $\mathbf{t}_i$:

$$D_i(\mathbf{t}_i) = \frac{1}{z} \sum_{\mathbf{q}} \min(\|d_1(\mathbf{q}) - d_2(\mathbf{q} + \mathbf{t}_i)\|_1, \lambda), \qquad (2)$$

where $\mathbf{q}$ denotes pixel coordinates within a node $i$ from which local descriptors were extracted, $z$ is the total number of descriptors, and $d_1$ and $d_2$ are descriptors extracted at the locations $\mathbf{q}$ and $\mathbf{q} + \mathbf{t}_i$ in the first and second image, respectively. For robustness to outliers, we use a truncated L1 norm for descriptor distance with a threshold $\lambda$. Note that $z = 1$ at the pixel layer, where $\mathbf{q}$ contains a single point.

The smoothness term $V_{ij}$ regularizes the solution by penalizing large discrepancies in the matching locations of neighboring nodes: $V_{ij} = \min(\|\mathbf{t}_i - \mathbf{t}_j\|_1, \gamma)$. We again use a truncated L1 norm with a threshold $\gamma$.

How does our objective differ from the conventional pixel-wise model? There are three main factors. First of all,


(a) Fixed-scale match

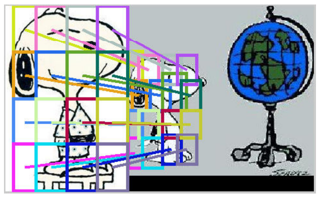

(b) Multi-scale match

Figure 3. Comparing our fixed- and multi-scale matches. For visibility, we show matches only at a single level in the pyramid. In (a), the match for a node in the first image remains at the same fixed scale in the second image. In (b), the multi-scale objective allows the size of each node to optimally adjust when matched.

graph nodes in our model are defined by cells of varying spatial extents, whereas in prior models they are restricted to pixels. This allows us to overcome appearance match ambiguities without committing to a single spatial scale. Second, our data term aggregates many local SIFT matches within each node, as opposed to using a single match at each individual pixel. This greatly enhances robustness to image variations. Third, we explicitly link the nodes of different spatial extents to impose smoothness, striking a balance between strong regularization by the larger nodes and accurate localization by the finer nodes.

We minimize the main objective function (Eq. 1) using loopy belief propagation to find the optimal correspondence of each node (see Sec. 3.3 for details). Note that the resulting matching is asymmetric, mapping all of the nodes in the first image to some (possibly subset of) positions in the second image. Furthermore, while our method returns matches for all nodes in all levels of the pyramid, we are generally interested in the final dense matches at the pixel level. They are what we will use in the results.

**Multi-Scale Extension** Thus far, we assume the matching is done at a fixed scale: each grid cell is matched to another region of the same size. Now, we extend our objective to allow nodes to be matched across different scales:

$$E(\mathbf{t},\mathbf{s}) = \sum_i D_i(\mathbf{t}_i,\mathbf{s}_i) + \alpha \sum_{i,j\in\mathcal{N}} V_{ij}(\mathbf{t}_i,\mathbf{t}_j) + \beta \sum_{i,j\in\mathcal{N}} W_{ij}(\mathbf{s}_i,\mathbf{s}_j). \tag{3}$$

Eq. 3 is a multi-scale extension of Eq. 1. We add a scale variable $\mathbf{s}_i$ for each node and introduce a scale smoothness term $W_{ij} = \|\mathbf{s}_i - \mathbf{s}_j\|_1$ with an associated weight constant $\beta$. The scale variable is allowed to take discrete values from a specified range of scale variations (to be defined below). The data term is also transformed into a multi-variate function defined as:

$$D_i(\mathbf{t}_i,\mathbf{s}_i) = \frac{1}{z}\sum_{\mathbf{q}} \min(\|d_1(\mathbf{q}) - d_2(\mathbf{s}_i(\mathbf{q}+\mathbf{t}_i))\|_1, \lambda), \tag{4}$$

where we see the corresponding location of descriptor $d_2$ for a descriptor $d_1$ is now determined by a translation $\mathbf{t}_i$ followed by a scaling $\mathbf{s}_i$.

Note that we allow each node to take its own optimal scale, rather than determine the best global scale between two images. This is beneficial when an image includes both foreground and background objects of different scales, or when individual objects have different sizes. See Figure 3.

Dense correspondence for generic image matching is often treated at a fixed scale, though there are some multi-scale implementations in related work. PatchMatch has a multi-scale extension that expands the correspondence search range according to the scale of the previously found match [2]. As in the fixed-scale case, our method has the advantage of modeling geometric distortion and match consistency across multiple spatial extents. While we handle scale adaptation through the matching objective, one can alternatively consider representing each pixel with a set of SIFTs at multiple scales [9]; that feature could potentially be plugged into any matching method, including ours, though its extraction time is far higher than typical fixed-scale features. Our multi-scale matching is efficient and works even with fixed-scale features.

### 3.3. Efficient Computation

For dense matching, computation time is naturally a big concern for scalability. Here we explain how we maintain efficiency both through our problem design and some technical implementation details.

There are two major components that take most of the time: (1) computing descriptor distances at every possible translation and (2) optimization via belief propagation (BP). For the descriptor distances, the complexity is $O(mlk)$, where $m$ is the number of descriptors extracted in the first image, $l$ is the number of possible translations, and $k$ is the descriptor dimension. For BP, we use a generalized distance transform technique, which reduces the cost of message passing between nodes from $O(l^2)$ to $O(l)$ [8]. Even so, BP's overall run-time is $O(nl)$, where $n$ is the number of nodes in the graph. Thus, the total cost of our method is $O(mlk+nl)$ time. Note that $n$, $m$, and $l$ are all on the order of the number of pixels (i.e., $\sim 10^5 - 10^6$); if solving the problem at once, it is far from efficient.

Therefore, we use a hierarchical approach to improve efficiency. We initialize the solution by running BP for a graph built on all the nodes except the pixel-level ones (which we will call first-layer), and then refine it at the pixel nodes (which we will call second-layer). In Figure 2, the first three images on the left comprise the first layer, and the fourth depicts the second (pixel) layer.

Compared to SIFT Flow's hierarchical variant [15], ours runs an order of magnitude faster, as we will show in the results. The key reason is the two methods' differing matching objectives: ours is on a pyramid, theirs is a pixel model. Hierarchical SIFT Flow solves BP on the *pixel grids* in the image pyramid; starting from a downsampled image, it progressively narrows down the possible solution space as it moves to the finer images, reducing the number of possible translations $l$. However, $n$ and $m$ are still on the order of the number of pixels. In contrast, the number of nodes in our first-layer BP is just tens. Moreover, we observe that sparse descriptor sampling is enough for the first-layer BP: as long as a grid cell includes $\sim$100s of local descriptors within it, its average descriptor distance for the data term (Eq. 2) provides a reliable matching cost. Thus, we don't need dense descriptors in the first-layer BP, substantially reducing $m$.

In addition, our decision not to link edges between pixels (i.e., no loopy graph at the pixel layer) means the second-layer solution can be computed very efficiently in a non-iterative manner. Once we run the first-layer BP, the optimal translation $\mathbf{t}_i$ at a pixel-level node $i$ is simply determined by: $\mathbf{t}_i = \arg\min_{\mathbf{t}}(D_i(\mathbf{t}) + \alpha V_{ij}(\mathbf{t}, \mathbf{t}_j))$, where a node $j$ is a parent grid cell of a pixel node $i$, and $\mathbf{t}_j$ is a fixed value obtained from the first-layer BP.

Our multi-scale extension incurs additional cost due to the scale smoothness and multi-variate data terms. The former affects message passing; the latter affects the descriptor distance computation. In a naive implementation, both linearly increase the cost in terms of the number of the scales considered. For the data term, however, we can avoid repeating computation per scale. Once we obtain $D_i(\mathbf{t}_i, \mathbf{s}_i = 1.0)$ by computing the pairwise descriptor distance at $\mathbf{s}_i = 1.0$, it can be re-used for all other scales; the data term $D_i(\mathbf{t}_i, \mathbf{s}_i)$ at scale $\mathbf{s}_i$ maps to $D_i((\mathbf{s}_i - 1)\mathbf{q} + \mathbf{s}_i\mathbf{t}_i, \mathbf{s}_i = 1.0)$ of the reference scale (see supplementary file for details). This significantly reduces computation time, in that SIFT distances dominate the BP optimization since $m$ is much higher than the number of nodes in the first-layer BP.

## 4. Results

The main goals of the experiments are (1) to evaluate raw matching quality (Sec. 4.1), (2) to validate our method applied to sematic segmentation (Sec. 4.2), and (3) to verify the impact of our multi-scale extension (Sec. 4.3).

We compare our deformable spatial pyramid (DSP) approach to state-of-the-art dense pixel matching methods, SIFT Flow [15] (**SF**) and PatchMatch [2] (**PM**), using the authors' publicly available code. We use two datasets: the Caltech-101 and LabelMe Outdoor (LMO) [14].

**Implementation details:** We fix the parameters of our method for all experiments: $\alpha = 0.005$ in Eq. 1, $\gamma = 0.25$, and $\lambda = 500$. For multi-scale, we set $\alpha = 0.005$ and $\beta = 0.005$ in Eq. 3. We extract SIFT descriptors of 16x16 patch size at every pixel using VLFeat [19]. We apply PCA to the extracted SIFT descriptors, reducing the dimension to 20. This reduction saves about 1 second per image match without losing matching accuracy.[1] For multi-scale match, we use seven scales between 0.5 and 2.0—we choose the search scale as an exponent of $2^{\frac{i-4}{3}}$, where $i = 1, ..., 7$.

| Approach | LT-ACC | IOU | LOC-ERR | Time (s) |
|---|---|---|---|---|
| DSP (Ours) | **0.732** | **0.482** | **0.115** | **0.65** |
| SIFT Flow [15] | 0.680 | 0.450 | 0.162 | 12.8 |
| PatchMatch [2] | 0.646 | 0.375 | 0.238 | 1.03 |

Table 1. Object matching on the Caltech-101. We outperform the state-of-the-art methods in both matching accuracy and speed.

| Approach | LT-ACC | Time (s) |
|---|---|---|
| DSP (Ours) | **0.706** | **0.360** |
| SIFT Flow [15] | 0.672 | 11.52 |
| PatchMatch [2] | 0.607 | 0.877 |

Table 2. Scene matching on the LMO dataset. We outperform the current methods in both accuracy and speed.

**Evaluation metrics:** To measure image matching quality, we use label transfer accuracy (LT-ACC) between pixel correspondences [14]. Given a test and an exemplar image, we transfer the annotated class labels of the exemplar pixels to the test ones via pixel correspondences, and count how many pixels in the test image are correctly labeled.

For object matching in Caltech-101 dataset, we also use the intersection over union (IOU) metric [6]. Compared to LT-ACC, this metric allows us to isolate the matching quality for the foreground object, separate from the irrelevant background pixels.

We also evaluate the localization error (LOC-ERR) of corresponding pixel positions. Since there are no available ground-truth pixel matching positions between images, we obtain pixel locations using an object bounding box: pixel locations are given by the normalized coordinates with respect to the box's position and size. For details, please see the supplementary file.

### 4.1. Raw Image Matching Accuracy

In this section, we evaluate raw pixel matching quality in two different tasks: object matching and scene matching.

**Object matching under intra-class variations:** For this experiment, we randomly pick 15 pairs of images for each object class in the Caltech-101 (total 1,515 pairs of images). Each image has ground-truth pixel labels for the foreground object. Table 1 shows the result. Our DSP outperforms SIFT Flow by 5 points in label transfer accuracy, yet is about 25 times faster. We achieve a 9 point gain over PatchMatch, in about half the runtime. Our localization error and IOU scores are also better.

[1]We use the same PCA-SIFT for ours and PatchMatch. For SIFT Flow, however, we use the authors' custom code to extract SIFT; we do so because we observed SIFT Flow loses accuracy when using PCA-SIFT.

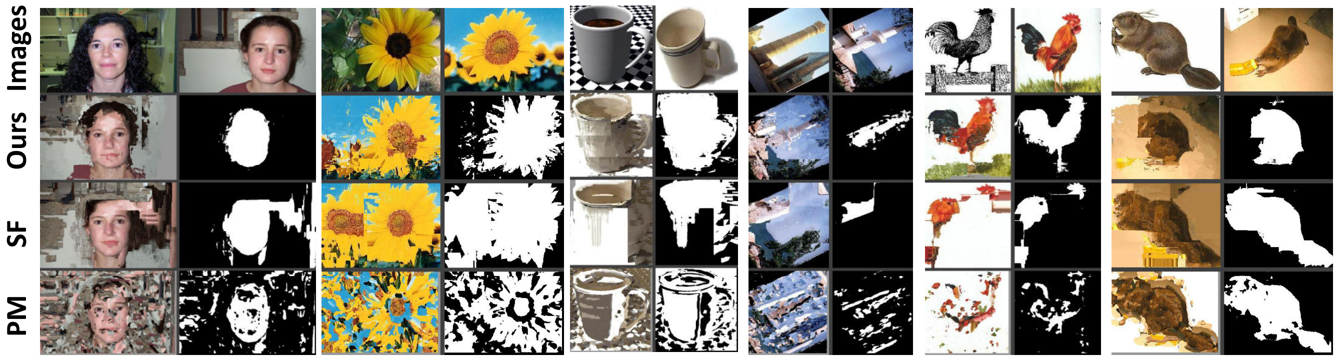


Figure 4. Example object matches per method. In each match example (rows 2-4), the left image shows the result of warping the second image to the first via pixel correspondences, and the right one shows the transferred pixel labels for the first image (white: fg, black: bg). We see that ours works robustly under image variations like background clutter (1st and 2nd examples), appearance change (4th and 5th ones). Further, even when objects lack texture (3rd example), ours finds reliable correspondences, exploiting global object structure. However, the single-scale version of our method fails when objects undergo substantial scale variation (6th example). Best viewed in color.

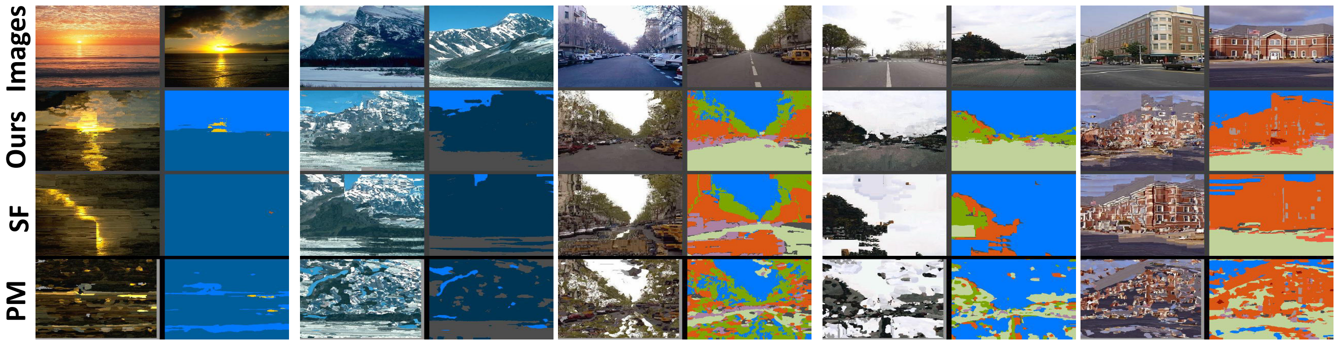


Figure 5. Example scene matches per method. Displayed as in Fig. 4, except here the scenes have multiple labels (not just fg/bg). Pixel labels are marked by colors, denoting one of the 33 classes in the LMO dataset. Best viewed in color.

Figure 4 shows example matches by the different methods. We see that DSP works robustly under image variations like appearance change and background clutter. On the other hand, the two existing methods—both of which rely on only local pixel-level appearance—get lost under the substantial image variations. This shows our spatial pyramid graph successfully imposes geometric regularization from various spatial extents, overcoming the matching ambiguity that can arise if considering local pixels alone. We also can see some differences between the two existing models. PatchMatch abandons explicit geometric smoothness for speed. However, this tends to hurt matching quality—the matching positions of even nearby pixels are quite dithered, making the results noisy. On the other hand, SIFT Flow imposes stronger smoothness by MRF connections between nearby pixels, providing visually more pleasing results. In effect, DSP combines the strengths of the other two. Like PatchMatch, we remove neighbor links in the pixel-level optimization for efficiency. However, we can do this without hurting accuracy since larger spatial nodes in our model enforce a proper smoothness on pixels.

**Scene matching:** Whereas the object matching task is concerned with foreground/background matches, in the scene matching task each pixel in an exemplar is annotated with one of multiple class labels. Here we use the LMO dataset, which annotates pixels as one of 33 class labels (e.g., river, car, grass, building). We randomly split the test and exemplar images in half (1,344 images each). For each test image, we first find the exemplar image that is its nearest neighbor in GIST space. Then, we match pixels between the test image and the selected exemplar. When measuring label transfer accuracy, we only consider the matchable pixels that belong to the classes common to both images. This setting is similar to the one in [14].

Table 2 shows the results.[2] Again, our DSP outperforms the current methods. Figure 5 compares some example scene matches. We see that DSP better preserves the scene structure; for example, the horizons (1st, 3rd, and 4th examples) and skylines (2nd and 5th) are robustly estimated.

[2]The IOU and LOC-ACC metrics assume a figure-ground setting, and hence are not applicable here.

| Approach | LT-ACC | IOU |
|---|---|---|
| DSP (Ours) | **0.868** | **0.694** |
| SIFT Flow [15] | 0.820 | 0.641 |
| PatchMatch [2] | 0.816 | 0.604 |

Table 3. Figure-ground segmentation results in Caltech-101.

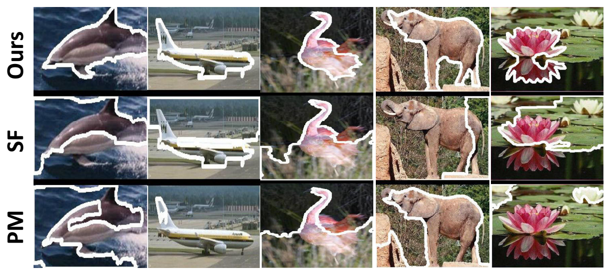


Figure 6. Example figure-ground segmentations on Caltech-101.

## 4.2. Semantic Segmentation by Matching Pixels

Next, we apply our method to a semantic segmentation task, following the protocol in [14]. To explain briefly, we match a test image to multiple exemplar images, where pixels in the exemplars are annotated by ground-truth class labels. Then, the best matching scores (SIFT descriptor distances) between each test pixel and its corresponding exemplar pixels define the class label likelihood of the test pixel. Using this label likelihood, we use a standard MRF to assign a class label to each pixel. See [14] for details. Building on this common framework, we test how the different matching methods influence segmentation quality.

**Category-specific figure-ground segmentation:** First, we perform binary figure-ground segmentation on Caltech. We randomly select 15/15 test/exemplar images from each class. We match a test image to exemplars from the same class, and perform figure-ground segmentation with an MRF as described above. Table 3 shows the result. Our DSP outperforms the current methods substantially. Figure 6 shows example segmentation results. We see that our method successfully delineates foreground objects from confusing backgrounds.

**Multi-class pixel labeling:** Next, we perform semantic segmentation on the LMO dataset. For each test image, we first retrieve an initial exemplar "shortlist", following [14]. The test image is matched against only the shortlist exemplars to estimate the class likelihoods.[3] We test three different ways to define the shortlist: (1) using the ground truth (GT), (2) using GIST neighbors (GIST), and (3) using an SVM to classify the images into one of the 8 LMO scene categories, and then retrieving GIST neighbors among only exemplars from that scene label (SVM).

[3]Our test/exemplar split, shortlist, and MRF are all identical to those in [14], except we do not exploit any prior knowledge (e.g., likelihood of possible locations of each class in the image) to augment the cost function of the MRF. Instead, we only use match scores in order to most directly compare the impact of the three matching methods.

| Approach | LT-ACC (GT) | LT-ACC (GIST) | LT-ACC (SVM) |
|---|---|---|---|
| DSP (Ours) | **0.868** | 0.745 | **0.761** |
| SIFT Flow [15] | 0.834 | **0.759** | 0.753 |
| PatchMatch [2] | 0.761 | 0.704 | 0.701 |

Table 4. Semantic segmentation results on the LMO dataset.

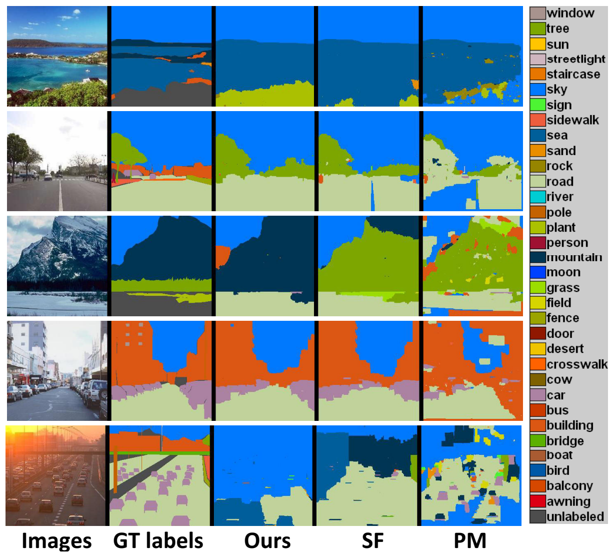


Figure 7. Example semantic segmentations on the LMO dataset. Our DSP and SIFT Flow (SF) both work reasonably on this dataset, though our segmentation is more consistent to the image's scene layout (e.g., the first and the third row). On the other hand, PatchMatch (PM) results are quite noisy. The last row shows our failure case, where we fail to segment small objects (cars).

Table 4 shows the results. The segmentation accuracy depends on the shortlist mechanism, for all methods. When using ground-truth annotations to choose the shortlist, our method clearly outperforms the others. On the other hand, when using automatic methods to generate the shortlist (GIST and SVM), our gain becomes smaller. This is because (1) the shortlist misses reasonable exemplar images that share class labels with the test image and (2) SIFT features may not be strong enough to discriminate confusing classes in a noisy shortlist—some classes (e.g., grass, field, and tree) are too similar to be distinguished by SIFT match scores alone. Again, our method is more efficient; 15-20 times faster than SIFT Flow, and about twice as fast as Patch Match. Figure 7 shows example segmentation results.

## 4.3. Multi-scale Matching

Finally, we show the results of our multi-scale formulation. For this experiment, we test using the same image pairs from Caltech as used in Sec. 4.1. We compare our multi-scale method to various baselines, including all the fixed-scale methods in the previous section and PatchMatch

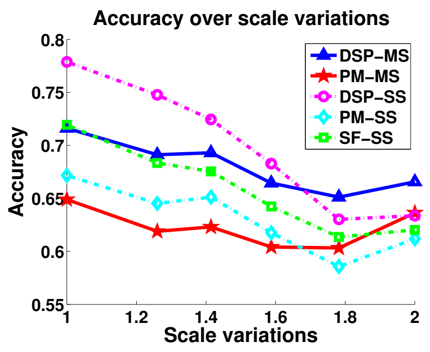


Figure 8. Matching accuracy as a function of scale variations. MS and SS denote multi-scale and single-scale, respectively.

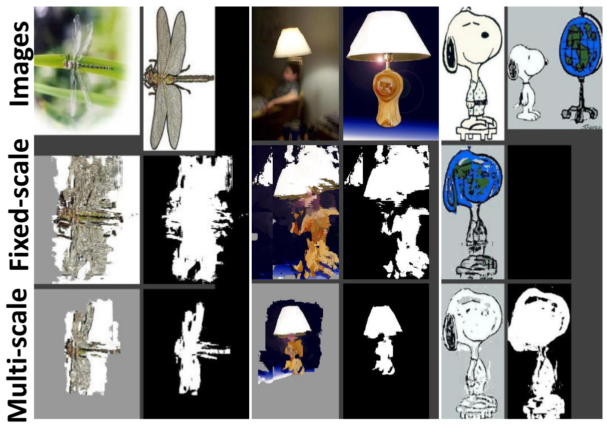


Figure 9. Example matching results by our multi-scale matching. For comparison, we also show results from our fixed-scale version. Our multi-scale method successfully finds the correct scale between objects, providing accurate matching. On the other hand, a single-scale one prefers the fixed size between objects, causing gross errors: e.g., in the 3rd example, Snoopy matches to a globe since they have similar size.

with its multi-scale option on.

Figure 8 plots the matching accuracy as a function of scale variation. The scale variation between two objects is defined by: $\frac{\max(O_1,O_2)}{\min(O_1,O_2)}$, where $O_1$ and $O_2$ are the size of matched objects in the first and the second image respectively. We see that the curves from multi-scale methods (DSP-MS and PM-MS) are flatter than the single-scale ones, verifying their relative scale tolerance. In addition, our multi-scale method (DSP-MS) outperforms multi-scale PatchMatch (PM-MS) by a substantial margin. However, we also see our curve is not perfectly flat across the scale changes. This is because scale is not the only factor that affects the matching. In fact, as scale variation increases, we observe that objects undergo more variations in viewpoint or shapes as well, making the matching more challenging.

Figure 9 shows some matching examples by our multi-scale method, compared to our single-scale counterpart. The examples show that our multi-scale matching is flexible to scale variations.

## 5. Conclusion

We introduced a deformable spatial pyramid model for dense correspondences across different objects or scenes. Through extensive evaluations, we showed that (1) various spatial supports by our spatial pyramid improve matching quality, striking a balance between geometric regularization and accurate localization, (2) our pyramid structure permits efficient hierarchical optimization, enabling fast dense matching, and (3) our model can be extended into a multi-scale setting, working flexibly under scale variations. As such, compared to the existing methods that rely on a flat pixel-based model, we achieve substantial gains in both matching accuracy and runtime. We share our code at http://vision.cs.utexas.edu/projects/dsp.

**Acknowledgements:** This research is supported in part by ONR N00014-12-1-0068.

## References

[1] S. Baker, R. Szeliski, and P. Anandan. A Layered Approach to Stereo Reconstruction. In *CVPR*, 1998.

[2] C. Barnes, E. Shechtman, D. Goldman, and A. Finkelstein. The Generalized PatchMatch Correspondence Algorithm. In *ECCV*, 2010.

[3] A. Berg, T. Berg, and J. Malik. Shape Matching and Object Recognition Low Distortion Correspondences. In *CVPR*, 2005.

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *PAMI*, 23(11), 2001.

[5] O. Duchenne, A. Joulin, and J. Ponce. A Graph-matching Kernel for Object Categorization. In *ICCV*, 2011.

[6] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *IJCV*, 88(2), 2010.

[7] P. Felzenswalb, D. McAllester, and D. Ramanan. A Discriminatively Trained Multiscale Deformable Part Model. In *CVPR*, 2008.

[8] P. Felzenszwalb and D. Huttenlocher. Efficient Belief Propagation for Early Vision. *IJCV*, 70(1), 2006.

[9] T. Hassner, V. Mayzels, and L. Zelnik-Manor. On SIFTs and Their Scales. In *CVPR*, 2012.

[10] K. Karsch, C. Liu, and S. Kang. Depth Extraction from Video Using Non-parametric Sampling. In *ECCV*, 2012.

[11] J. Kim and K. Grauman. Asymmetric Region-to-Image Matching for Comparing Images with Generic Object Categories. In *CVPR*, 2010.

[12] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *CVPR*, 2006.

[13] M. Leordeanu and M. Hebert. A Spectral Technique for Correspondence Problems using Pairwise Constraints. In *ICCV*, 2005.

[14] C. Liu, J. Yuen, and A. Torralba. Nonparametric Scene Parsing via Label Transfer. *PAMI*, 33(12), 2011.

[15] C. Liu, J. Yuen, and A. Torralba. SIFT Flow: Dense Correspondence across Different Scenes and Its Applications. *PAMI*, 33(5), 2011.

[16] M. Pandey and S. Lazebnik. Scene Recognition and Weakly Supervised Object Localization with Deformable Part-Based Models. In *ICCV*, 2011.

[17] D. Scharstein and R. Szeliski. A Taxonomy and Evaluation of Dense Two-frame Stereo Correspondence Algorithms. *IJCV*, 47, 2002.

[18] H. Shum and R. Szeliksi. Construction and Refinement of Panoramic Mosaics with Global and Local Alignment. In *ICCV*, 1998.

[19] VLFeat Open Source Library. *http://www.vlfeat.org/*.

[20] D. Xu, T. Cham, S. Yan, and S. Chang. Near Duplicate Image Identification with Spatially Aligned Pyramid Matching. In *CVPR*, 2008.