Journal Subline
LNCS 7870

Anton Konushin
Guest Editor

Transactions on Computational Science XIX

Marina L. Gavrilova · C. J. Kenneth Tan
Editors-in-Chief

Special Issue on Computer Graphics

Springer

Lecture Notes in Computer Science 7870

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Marina L. Gavrilova C.J. Kenneth Tan
Anton Konushin (Eds.)

Transactions on Computational Science XIX

Special Issue on Computer Graphics

Springer

ISSN 0302-9743 (LNCS) e-ISSN 1611-3349 (LNCS)
ISSN 1866-4733 (TCOMPSCIE) e-ISSN 1866-4741 (TCOMPSCIE)
ISBN 978-3-642-39758-5 e-ISBN 978-3-642-39759-2
DOI 10.1007/978-3-642-39759-2
Springer Heidelberg Dordrecht London New York

CR Subject Classification (1998): I.4, I.3, I.2, H.5

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

LNCS Transactions on Computational Science

Computational science, an emerging and increasingly vital field, is now widely recognized as an integral part of scientific and technical investigations, affecting researchers and practitioners in areas ranging from aerospace and automotive research to biochemistry, electronics, geosciences, mathematics, and physics. Computer systems research and the exploitation of applied research naturally complement each other. The increased complexity of many challenges in computational science demands the use of supercomputing, parallel processing, sophisticated algorithms, and advanced system software and architecture. It is therefore invaluable to have input by systems research experts in applied computational science research.

Transactions on Computational Science focuses on original high-quality research in the realm of computational science in parallel and distributed environments, also encompassing the underlying theoretical foundations and the applications of large-scale computation. The journal offers practitioners and researchers the opportunity to share computational techniques and solutions in this area, to identify new issues, and to shape future directions for research, and it enables industrial users to apply leading-edge, large-scale, high-performance computational methods.

In addition to addressing various research and application issues, the journal aims to present material that is validated – crucial to the application and advancement of the research conducted in academic and industrial settings. In this spirit, the journal focuses on publications that present results and computational techniques that are verifiable.

Scope

The scope of the journal includes, but is not limited to, the following computational methods and applications:

- Aeronautics and Aerospace
- Astrophysics
- Bioinformatics
- Climate and Weather Modeling
- Communication and Data Networks
- Compilers and Operating Systems
- Computer Graphics
- Computational Biology
- Computational Chemistry
- Computational Finance and Econometrics
- Computational Fluid Dynamics
- Computational Geometry

- Computational Number Theory
- Computational Physics
- Data Storage
- Data Mining and Data Warehousing
- Geology and Geophysics
- Grid Computing
- Hardware/Software Co-design
- High-Energy Physics
- High-Performance Computing
- Information Retrieval
- Modeling and Simulations
- Numerical and Scientific Computing
- Parallel and Distributed Computing
- Reconfigurable Hardware
- Supercomputing
- System-on-Chip Design and Engineering
- Virtual Reality
- Visualization

Editorial

The Transactions on Computational Science journal is part of the Springer series *Lecture Notes in Computer Science*, and is devoted to the gamut of computational science issues, from theoretical aspects to application-dependent studies and the validation of emerging technologies.

The journal focuses on original high-quality research in the realm of computational science in parallel and distributed environments, encompassing the facilitating theoretical foundations and the applications of large-scale computations and massive data processing. Practitioners and researchers share computational techniques and solutions in the area, identify new issues, and shape future directions for research, as well as enable industrial users to apply the techniques presented.

The current volume is devoted to the topic of computer graphics and is edited by Anton Konushin. It is comprised of 12 excellent papers selected from over 100 submissions to GRAPHICON 2012, held at Lomonosov Moscow State University, Moscow, Russia, in October 2012.

We would like to extend our sincere appreciation to the special issue guest editor Anton Konushin for his effort on this special issue. We would also like to thank all of the authors for submitting their papers to the special issue and the associate editors and referees for their valuable work. We would like to express our gratitude to the LNCS editorial staff of Springer, in particular Alfred Hofmann, Ursula Barth and Anna Kramer, who supported us at every stage of the project.

It is our hope that the fine collection of papers presented in this special issue will be a valuable resource for Transactions on Computational Science readers and will stimulate further research into the vibrant area of computer graphics.

April 2013

Marina L. Gavrilova
C.J. Kenneth Tan

Special Issue on Computer Graphics Guest Editor's Preface

While computer graphics can be regarded as a mature discipline, computer vision and image processing continue to be among the most rapidly evolving areas of computer science. Both computer graphics and vision have enabled a lot of applications, ranging from medical visualization to interactive videogames and gesture-based interfaces.

The international conference GraphiCon 2012 addressed a wide range of research and development topics in the field of computer graphics and vision. Out of 77 registered papers a final selection of 3 invited, 47 full, and 11 short papers comprised the GraphiCon 2012 program. All papers were subjected to a double blind review and were sent to three members of the program committee. The 12 articles appearing in this special issue are revised and extended versions of a selection of papers presented at GraphiCon 2012. The papers were selected based on several criteria, including reviewers' comments, quality of presentation, and feedback of conference participants.

Nowadays computer tomography (CT) is a common medical analysis tool, but each scanning involves radiation. Lowering radiation doses leads to increased noise in CT images, which in turn can make processing erroneous. New methods, described in the paper "2.5D Extension of Neighborhood Filters for Noise Reduction in 3D Medical CT Images", are a step forward in solving this important problem.

Real-time photorealistic rendering with global illumination is a long-term goal of computer graphics. The improvement of Graphics Processing Units (GPUs) during the last decade has resulted in a great advancement of this topic. However, most papers on ray tracing and global illumination with GPU acceleration focus on interactive frame rates and do not pay enough attention to the quality. "Implementing Irradiance Cache in a GPU Realistic Renderer" provides a balanced approach to implementing widely used irradiance cache techniques in a GPU. We have a second paper in our selection dedicated to implementation of ray tracing techniques in a GPU – "GPU Ray Tracing – Comparative Study on Ray-Triangle Intersection Algorithms", with a title that speaks for itself.

3D images and 3D cinema are a booming and highly-marketed technology. "Adaptive Generation of Color Anaglyph" deals with problems arising during anaglyph-image preparations on consumer printing hardware for education and entertainment.

"Audio-adaptive Animation from Still Image" is devoted to another consumer application – creating animated images from still photos, which is a fun way to improve the visual quality of a presentation. Three effects – Flashing Light, Soap Bubbles, and Sunlight Spot – are described in details.

"Auto-calibration for Image Mosaicing and Stereo Vision" explores the problem of camera calibration, which is a necessary step during panoramic image stitching or 3D reconstruction. Existing algorithms are extended to get a robust method that computes internal camera parameters given a series of distant-object images. Experiments show that accurate calibration without patterns is possible if the quality of input images is sufficient.

Spectral methods constitute a popular technique for image segmentation and matting. However, careful parameter selection is required to obtain optimal performance. In "Learning Graph Laplacian for Image Segmentation" a new method is presented, which allows for unsupervised learning of graph Laplacian parameters individually for each image without using any prior information.

Virtual Reality is a very popular subject in science fiction, figuring mainly as a feature of entertainment systems. However, currently it is still too expensive, has significant limitations, and is used mainly for research, e.g., psychological studies. "Virtual Reality Technology for the Visual Perception Study" assesses effects of 2D vs. 3D displays on lightness perception using the CAVE system.

Machine vision industrial applications usually require a special setup, including digital camera and lighting system, e.g., several LEDs as point light sources. In "Locally Adapted Detection and Correction of Unnatural Purple Colors in Images of Refractive Objects Taken by Digital Still Camera" one particular setup is examined. It was observed that images captured with a digital still camera occasionally exhibit bright purple (almost pink) colors, which do not correspond to any monochromatic color. A set of algorithms is proposed to identify image areas to be corrected and to map all unnatural colors to the natural ones in those areas.

Personal robotics will greatly improve quality of life for many people. A lot of problems still should be solved in order to create a robot that will be fully capable of performing such mundane duties as washing plates and dishes. One particular problem is the detection and relative 3D location estimation of transparent objects, like glass bottles and cups. "Pose Refinement of Transparent Rigid Objects with a Stereo Camera" explores this problem and proposes a new method for it.

Finally, two papers in our selection are both devoted to scientific visualization. In "Some Theoretical Issues of Scientific Visualization as a Method of Data Analysis" characteristics of scientific visualization as a modern computer-based method of scientific data analysis are given as observed by the authors from the generalization of practical experience. "Analysis of Space-Time Flow Structures by Optimization and Visualization Methods" considers a specific problem of space-time structures analysis for unsteady problems in CFD (computational fluid dynamics). A new approximate approach is proposed, based on the solution of optimization problems combined with methods of data visual presentation.

Thanks and appreciation go to the authors, the reviewers, and the staff working on the Transactions of Computational Science.

March 2013 Anton Konushin

LNCS Transactions on Computational Science – Editorial Board

Table of Contents

2.5D Extension of Neighborhood Filters for Noise Reduction in 3D Medical CT Images*

Maria Storozhilova[1], Alexey Lukin[1], Dmitry Yurin[1], and Valentin Sinitsyn[2]

[1] Laboratory of Mathematical Methods of Image Processing,
Faculty of Computational Mathematics and Cybernetics,
Lomonosov Moscow State University, Moscow, Russia
http://imaging.cs.msu.ru
[2] Radiology Department at Federal Center of Medicine and Rehabilitation,
Moscow, Russia

Abstract. Noise in 3D computer tomography (CT) images is close to white and becomes large when patient radiation doses are reduced. We propose two methods for noise reduction in CT images: 3D extension of fast rank algorithms (Rank-2.5D) and 3D extension of a non-local means algorithm (NLM-2.5D). We call both our algorithms "2.5D" because the extended NLM algorithm is slightly asymmetric by slice axes, while our Rank algorithms, being fully symmetric mathematically and by results, have some implementation asymmetry. A comparison of the methods is presented. It is shown that NLM-2.5D method has the best quality, but is computationally expensive: its complexity quickly rises as a function of the neighborhood size, while Rank-2.5D only shows linear growth. Another contribution of this paper is a modified multiscale histogram representation in memory with a tree-like structure. This dramatically reduces memory requirements and makes it possible to process 16-bit DICOM data with full accuracy. Artificial test sequences are used for signal-to-noise performance measurements, while real CT scans are used for visual assessment of results. We also propose two new measures for no-reference denoising quality assessment based on the autocorrelation coefficient and entropy: both measures analyze randomness of the difference between noisy and filtered images.

Keywords: medical imaging, CT, DICOM, filtering, enhancement, noise reduction, denoising, 3D image processing, image quality assessment.

1 Introduction

The problem of noise reduction in digital images has a long history. The first algorithms were linear filters [1], such as convolutions with a low-pass window function (rectangular or Gaussian), frequency-domain filtering, Wiener filtering. The problem with linear methods is inevitable loss of image quality: loss of sharpness, blurring of edges, ringing effects.

* The research is done under support of the Russian Foundation for Basic Research, project no. 13-07-00584_a.

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XIX, LNCS 7870, pp. 1–16, 2013.

The next wide class of image filtering methods has been introduced in [2], they are called rank algorithms. The most well-known of these methods is the median filter, which has fast computational algorithms [3], [4], [5]. Median filter preserves sharp edges, but rounds the corners in the image. Recently we have proposed fast algorithms for other types of range filters [6], [7]. Before their existence, practical applications of such filters were limited. Unlike median filtering and linear methods, many rank algorithms, such as averaging with ϵ_V or KNV neighborhood, reduce the contrast of edges without blurring or changing their shape. A well-known bilateral filtering algorithm [8] can be roughly considered as being a rank filter too. If the existing Gaussian spatial and range kernels in [8] are replaced with rectangular kernels, a so-called ϵ_V averaging algorithm is produced [2]. The only fast algorithms known for the bilateral filter are approximating algorithms.

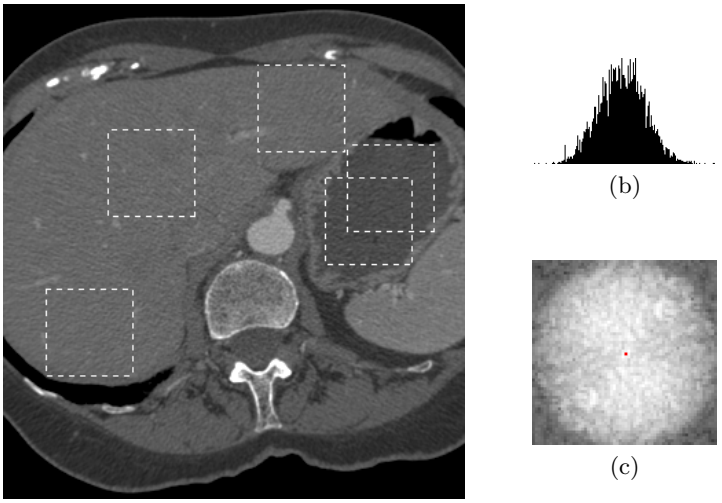
Most recent works in noise reduction show the advantage of methods which average pixels depending on their neighborhood statistics, not just pixel values [9], [10]. The first method, known as non-local means, calculates averaging weights using similarity of pixel neighborhoods (patches). The second one is more complex and consists of two stages. On the first stage a rough frequency-domain filtering is performed to facilitate the search of similar patches. The second stage is the joint filtering of groups of similar patches from the source image. In [10] there are comparisons of the proposed method with 5 other algorithms.

This paper focuses on noise reduction in computer tomography (CT) scans. CT images capture the density of a sequence of slices of a human body. These slices are obtained with a small fixed stride, which is perpendicular to the slice plane and is comparable with the pixel size in each slice. Together they represent a 3D set of data, so traditional 2D methods of image denoising are less than optimal because they fail to exploit a high degree of data correlation between slices. Independent noise reduction in each slice may cause difference in color and position of edges of objects between slices. Our main objective is to reduce noise and avoid the loss of small low-contrast image areas. Such areas may contain symptoms of an illness and it is important to maintain sharp edges and prevent the loss of information for them. This requirement limits the use of median filtering and other algorithms based on the image blurring as they can shift the edges or make them indistinguishable.

Noise is always present in CT images. Interestingly, as the noise reduction methods advance over time, one can expect even noisier raw data to be coming from the scanner, because it means lower radiation dose for the patient. Specialized algorithms for denoising of CT images are an area of active research [11], [12].

The noise spectrum in our images is close to white: slightly low-pass, but without evident directionality (Fig. 1c). The amplitude p.d.f. is close to Gaussian (Fig. 1b). This justifies the application of standard noise reduction methods, most of which have been formulated for additive white Gaussian noise. Another study of noise in CT images is given in [13], [14].

This paper proposes two approaches to noise reduction. A higher-quality algorithm is based on a non-local means [9] adapted to a 3D image data. We are using a fast, but not quite symmetric variant (we call it NLM-2.5D) developed in



(a) (b) (c)

Fig. 1. Analysis of noise in our CT images: (a) Collected noise patches; (b) Noise histogram; (c) Noise spectrum

our previous work for video processing [15]. The second approach (Rank-2.5D) is using the adaptation of fast rank filters which do not blur or shift the edges of the image [6], [7]. The proposed algorithm is fully symmetric in three dimensions and comprises the direct extension of ϵ_V averaging [2] on a 3D space, with an adaptive choice of ϵ in every point of the image. This ϵ is calculated using another rank algorithm. Every slice is processed using information from a *collection* of adjacent slices. The implementation of the algorithm is not symmetric: it is based on sequential processing of 2D slices and 1D filtering across slices. This allows considerable savings in memory, but slows down the computation. The processing time becomes proportional to the number of slices in the collection (neighborhood).

In our previous works we used array-like histogram representation [16] (and recently [6], [7]) and 8-bit images, while full accuracy DICOM images are 16-bit. Direct extension of our fast rank algorithms to 16-bit medical imaging results in unacceptable memory consumption (more than 4 Gb), so in this work we switch to tree-like structures and significantly renew the algorithms.

A comparison of speed and PSNR quality is carried out on phantom images with an artificially generated noise. Visual results on real medical images are also provided.

When using any denoising methods we face the problem of the algorithm quality assessment. According to the availability of the reference image, the image quality assessment (IQA) methods can be classified into three kinds: Full Reference (FR), No Reference (NR) and Reduced Reference (RR). Medical images can be referred to NR group as we have no ability to receive an ideal CT image

in real-world conditions. In this paper we also propose a new autocorrelation-based measure of IQA for medical images that were denoised with 2.5D Rank and 2.5D NLM algorithms.

We compare original noisy and filtered images and analyze their difference. The proposed measure based on the autocorrelation coefficient estimates randomness of the difference image. Thus if filtering procedure removes or changes some significant image details together with noise, some regular structures, like small spots and lines appear on difference image, and autocorrelation coefficient increases, signaling about excessive filtering.

2 ϵ_V Filtering

We start with some necessary definitions, following [2]. Consider the current v_0 and some neighborhood S of pixel v_0 that contains N pixels. Frequently S-neighborhood has a square, round or octagonal shape [2], [5].

Definition 1. *A rank series* $\{v(r)\}$ *is a one-dimensional sequence of* N *pixels from* S *whose elements are sorted in an ascending order with respect to their values:* $\{v(r) \leq v(r+1), r = 0..N-1\}$.

Definition 2. *Pixel's* v_R *rank* R *is the position of the* v_R *element in the rank series.* $R = rank(v_R)$.

Let's consider some selected pixel $v_c \in S$; for example we can consider $v_c = v_0$, $v_c = med\{v(r)\}$, or $v_c = mean\{v(r)\}$ as a pivot. It should be noted that under such definition the pivot pixel do not always satisfy conditions $v_c \in S$ or $v_c \in \{v(r)\}$. Then

Definition 3. ϵ_V *(or EV) neighborhood is a subset of pixels set* $\{v(r)\}$ *whose values deviate from the value of the pivot pixel at most by a predetermined quantity* ϵ*:*

$$\epsilon_V(v_c) = \{v(r) : |v(r) - v_c| \leq \epsilon\} \tag{1}$$

As it can be seen from the definition, ϵ_V neighborhood average can be treated as a simplified bilateral filter [8], where bilateral filter parts that depend both on distance and on pixel brightness, are represented by rectangular functions instead of Gaussians. For this reason ϵ_V filtering should keep the edges of the objects sharp, assuming the parameter ϵ is properly chosen.

3 2.5D Rank Filtering Algorithm

It is proposed to use a three-dimensional ϵ_V filtering algorithm with an adaptive search of parameter ϵ for denoising of CT images. Fast rank algorithms [6] are based on multiscale histogram approach. Either fast rank algorithms or a lazy calculations technique remain the same in a 3D case, though some difficulties occur with column histogram maintaining [5], [7]. Thus if a two-dimensional

image with the size of 512×512 pixels requires maintaining of 512+1 multiscale histograms, then in a three-dimensional case we will have to maintain $512 \times 512 + 1 = 262145$ multiscale histograms for a sequence of images with the same size. Taking into account the large size of multiscale histograms with the specific additional information [6], we may see that the number above is at the limit of memory size of 32-bit computers. We propose to use a separable approach for reduction of complexity by processing 2D images. At first, ϵ_V neighborhood average will be calculated separately with the same pivot for all N slices that are used for denoising of the current image. Then the 1D variant of ϵ_V filtering will be applied to the result. Really, if the inequality (1) is true for a number of subsets it is true for the union of these subsets too. ϵ_V-averaging means calculation of the ratio of sum of values to number of elements, so it is sufficient to sum up the sums for each subsets and the number elements in each subsets and than calculate their ration once.

For the correct choice of the ϵ parameter we propose to calculate the intensity variance in a square neighborhood of small radius R_{disp} for each pixel of the image. This step allows determining the uniformity of an area of a specified radius around the pixel. For a flat area with insignificantly varying color the intensity variance value will provide the noise variance.

The choice of R_{disp} is based on the fact that the noise in CT images is almost non-correlated (white). So the neighborhood can have an arbitrary radius. Since our main goal is to preserve small low-contrast objects, it is essential to choose R_{disp} in accordance with the size of the object that needs to be preserved. Since the variance will be high on the boundaries of areas with different colors, it is proposed to apply a some kind of minimum filter to the variance image, using the neighborhood of a larger radius. This allows propagating correct values of noise variation to regions with excessive variance.

So the next step is the min-filtering of variance by averaging pixels with the smallest rank of the neighborhood of a larger radius. To completely suppress high-variance values on boundaries it is proposed to use the neighborhood of a higher radius $3R_{disp}$. Let us consider possible cases for location of a boundary within the neighborhood. Since the radius of the neighborhood is small enough, we assume that the boundary between the objects inside the sliding window is a straight line.

In the easiest case Fig. 2a the amount of pixels that represent variance outside the edge area is about 30% of the bigger neighborhood. In the worst case Fig. 2b the amount of pixels for the variance of the outside-edge area is about 10%. These values can be easily proved by simple geometrical calculations.

Variance minimized in such a way corresponds to adaptive ϵ value. In order to control the degree of noise reduction it is proposed to use a multiplier parameter M. In general the coefficient M is user-defined, but it is essential to use the range of $[2 \ldots 4]$ in order to simultaneously suppress the majority of noise pixels and to avoid blurring of the edges.

Then ϵ_V algorithm with the ϵ value that was obtained on the previous step and the $R_{eps} = R_{disp} + 1$ neighborhood radius is performed for each slice. It is

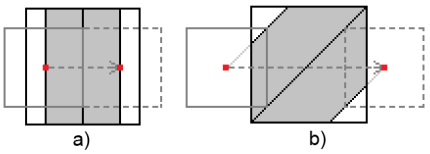


Fig. 2. The area with high variance value for (a) the vertical edge; (b) the diagonal edge

important to emphasize that we take not the central intensity value of the current slice but the intensity value of a source image with corresponding coordinates as a central pixel for ϵ_V neighborhood average algorithm. This condition is important because we apply the algorithm not to a 2D neighborhood, but to the volume region.

To extend the algorithm to a 3D space, it is proposed to calculate the total sum and the total number of elements involved in the averaging for the current pixel of the current slice. As a result, for each pixel of each slice we receive the structure that contains the total sum and the amount of the ϵ_V neighborhood elements.

The last step is ϵ_V filtering for slices. It is proposed to use the minimized variance for the source image multiplied by M as the value of ϵ for each pixel of the source image. The averaging is performed with only one structure from each slice (i.e. if we use N slices, only N elements will take part in averaging). For the pixel $I(x, y)$ of the source image the necessary structures will be located at the same coordinates in the corresponding slice. The number of slices for processing is defined automatically in accordance with the distance between slices.

Let N be the number of images (slices) for denoising of one image, I_{src} be the source image, I_i be the i-th image from the array of image slices, R_{disp} be the radius of the variance calculation window, $D(I_i)$ be the variance of the i-th image, M be the variance multiplier, R_{eps} be the radius for a 2D spatial ϵ_V filtering. The following algorithm is proposed for CT image denoising.

Algorithm 1. 2.5D rank denoising algorithm.

1. **for** $i := 0$, $i < N$ **do**
2. **for** each pixel $I_i(x, y)$ of I_i **do**
3. Compute the variance of $I_i(x, y)$ with the window radius R_{disp};
4. Minimize variance $D(I_i(x, y))$ for I_i by averaging K elements with the smallest rank using the window radius of $3 \cdot Rdips$;
5. Perform ϵ_V filtering around the current pixel's $I_i(x, y)$ neighborhood to obtain the total sum and total count of elements for $I_i(x, y)$. $\epsilon(x, y) = M \cdot D(I_i(x, y))$, the window radius is R_{eps} and $I_{src}(x, y)$ is taken as the central pixel of the neighborhood;
6. **endfor**
7. **endfor**
8. **for** each pixel $I_{src}(x, y)$ of I_src **do**

9. Perform 1D ϵ_V filtering with $\epsilon(x, y) = M \cdot D(I_src(x, y))$ for N corresponding elements $I_i(x, y), i = 1..N$;
10. **endfor**

4 16-bit Extension of Rank Algorithm

The previous algorithm was mainly designed for 8-bit grayscale images. But medical DICOM images originally consist of 16-bit grayscale data. For this case multiscale histogram approach should be slightly modified. In case of 8-bit image we store $w+1$ histograms of 256 elements. Using this approach for a 16-bit image we will need to store $w+1$ histograms of 65536 elements each, which means that the total memory usage will be around 4 GB for a single image denoising. And rather big amount of these elements will have 0 counts.

To avoid the inefficient use of memory and time for zero filling it is proposed to use a tree representation of multiscale histograms instead of an array. Tree nodes are allocated from the previously reserved pool. The maximal number of elements for level L of a multiscale histogram can be counted as a minimum from the neighborhood area and the total number of elements on each level. The total number of elements n for one multiscale histogram for a $L = 16$-bit image is:

$$n = \sum_{i=0}^{L} n_i, \quad n_i = \min\{2^i,\ (2r+1)^2\} \tag{2}$$

5 2.5D Non-local Means Filtering

Bilateral filtering algorithm is well known in image processing for its simplicity and edge-preserving properties [8]. The output pixel value $I_{out}(x, y)$ is formed as a weighted sum of pixel values from the neighborhood Ω:

$$I_{out}(x, y) = \frac{1}{\sum_{i,j\in\Omega} W} \cdot \sum_{i,j\in\Omega} W(x, y, j, i) \cdot I(x + j, y + i) \tag{3}$$

The weights W depend on geometric distance and color difference between pixels (x, y) and $(x+j, y+i)$ in order to facilitate averaging of pixels with similar values:

$$W(x, y, j, i) = exp\left\{-\frac{j^2 + i^2}{2\sigma^2}\right\} \cdot exp\left\{-\frac{(I(x + j, y + i) - I(x, y))^2}{2\rho^2}\right\} \tag{4}$$

Non-local means is a relatively novel method of image filtering that builds upon a bilateral algorithm. Formula (4) for pixel similarity in bilateral filtering considers colors and spatial coordinates of two pixels. In the non-local means algorithm, this formula instead considers *the context* of two pixels [9]. Specifically, instead of comparing values of two pixels, the algorithm compares the content of *image patches* v around two pixels:

$$W(x,y,j,i) = exp\left\{-\frac{||v(x+j,y+i)-v(x,y)||_2^2}{2\rho^2}\right\} \tag{5}$$

The squared norm of pixel-wise patch differences in formula (5) ensures that only pixels with a similar surrounding content are averaged together.

The extension of this method on a three-dimensional space is straightforward. For the standard 2D image processing, the neighborhood Ω from formula (3) is a circle or a square around the central pixel. For the 3D filtering, we extend the neighborhood to be a sphere or cube in the 3D image space: it includes several image slices that are adjacent to the processed pixel. A two-dimensional summation in formula (3) turns into a three-dimensional summation and calculation of weights in formula (5) is adjusted accordingly: similar patches are searched among the array of several adjacent slices. Since the compared patches are still two-dimensional, we call this method NLM-2.5D.

For improved speed of calculations we employ an optimization from [15] for sparse update of weights W.

6 Synthetic Phantom Generation

For testing the noise reduction capability of our algorithms we have generated a synthetic phantom image. We do not use real phantom images because they always contain noise due to CT image generation process. An attempt to shoot a phantom without noise results in high-dose shooting which is radically different from real medical imaging conditions. Our synthetic phantom construction (see Fig. 3) is described below. The phantom body is cylindrical. It has two cylindrical "organs" with different radioparency. These two "organs" are connected by two thin truncated cones ("processes" or "ligaments"), with radioparency smoothly changing from one organ to another. Each "organ" has one truncated conic "vessel" with different radioparency. These truncated cone shapes are used for thin objects in order to detect the extent of damage to the phantom along the slice axis in the process of filtering (if any).

We model all "body", "organs", "ligaments", and "vessels" with color c_j and density function

$$\rho(r) = exp\left\{-\left(\frac{r-r_0}{R}\right)^{2n}\right\}, \quad r=(x,y) \tag{6}$$

which is used smooth object boundaries, mainly to avoid pixelated (aliased) shapes. The center of small objects ("ligaments" and "vessels") depends on a slice position z: $\boldsymbol{r_0}$, $R(z)$, so density function depends on all 3 coordinates $\rho(\boldsymbol{r}) = \rho(x,y,z)$. The resulting slice image set is obtained recursively as

$$I_0(x,y,z) = \rho_{body}(r)\cdot c_{body} \tag{7}$$

$$I_j(x,y,z) = \rho_j(\boldsymbol{r})\cdot c_j + (1-\rho_j(\boldsymbol{r}))\cdot I_{j-1}(x,y,z) \tag{8}$$

where large objects are added first. Typical simulated phantom images are presented in Fig. 3.

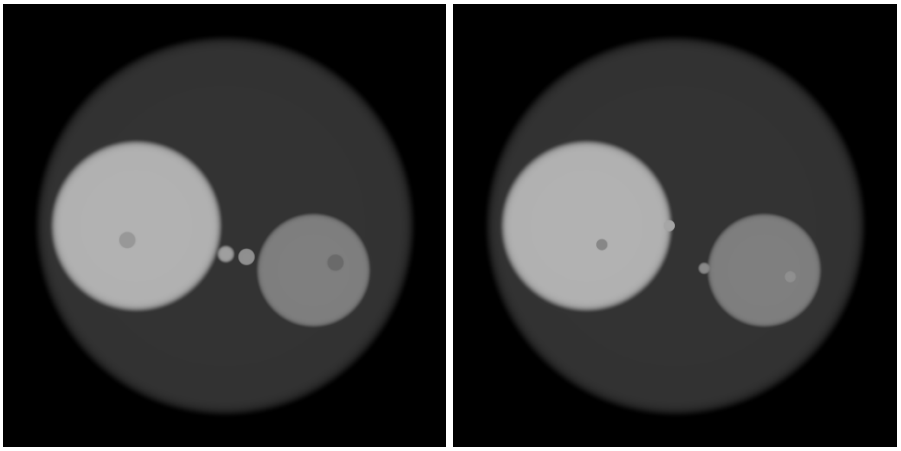


Fig. 3. Simulated noise-free phantom images

7 Analysis of Residual Images

It absence of the noise-free ground truth data, it is common to evaluate the results of noise reduction algorithms by informal analysis of the *residual image*. The residual image is formed as a difference between the original noisy image and the image after noise reduction, i.e. it shows the change happening during noise reduction.

$$I_r(x, y) = I_{noisy}(x, y) - I_{filtered}(x, y) \tag{9}$$

Several examples of residual images can be found in [9]. The desirable feature of the residual image is randomness: lack of apparent correlation with the original image. Simple methods, like linear filtering, suppress high-frequency image details, which becomes apparent in the residual image. More complex algorithms, like non-local means, are more successful in separating random noise from image features (Fig. 7b).

7.1 Correlation Measure

Here we propose a formal measure of randomness of the residual image and show that it can be used for automatic selection of noise reduction algorithm parameters. A simple measure of signal randomness is the *autocorrelation coefficient*, defined as autocorrelation function of a zero-mean signal normalized by the signal variance. While it is possible to compute autocorrelation function for the whole image, we are more interested in a local behavior of autocorrelation, because it reveals information about local image features. So, we compute autocorrelation in overlapping image blocks. The block size is chosen according to the size of features that we are interested in. For our 512×512 CT scans, we choose the block size of 32×32 pixels, with an overlap factor of 75% in each

dimension. Blocks of the difference image are windowed by the Hann window and the autocorrelation function $R_{xx}(i,j)$ is computed via FFT. The values of autocorrelation are then divided by the signal variance $R_{xx}(0,0)$. The resulting autocorrelation coefficients are within $[-1,1]$.

To analyze the presence of structure in the residual image, we propose searching for the maximal autocorrelation coefficient with a nonzero lag: $R^{max} = \max_{i,j\in\Omega} R_{xx}(i,j)$, where Ω contains all lags satisfying: $i^2+j^2 \geq 2^2$. This excludes lags smaller than 2 pixels to prevent snapping to the global maximum near lag 0, which may have a nonzero width due to a low-pass nature of CT reconstruction filters. When R^{max} is close to 0, the residual image is close to white noise, i.e. has no correlation. When R^{max} is high, there is a significant correlation, which indicates the presence of unwanted structure in the residual image. Thus the proposed coefficient can be used as a measure of image quality degradation.

The resulting correlation coefficients R_k^{max} from each block k are averaged over all image blocks to form the overall measure $\mathcal{R}$:

$$\mathcal{R} = \frac{\sum_k w_k R_k^{max}}{\sum_k w_k}, \tag{10}$$

where blocks with higher variance are assigned higher weights: $w_k = \sqrt[4]{R_{xx}(0,0)}$.

The resulting measure $\mathcal{R}$ indicates whether the residual image is "contaminated" with structures (such as edges) from the original image. If the residual image is a synthetic white noise, our simulations show that $\mathcal{R} \approx 0.14$. In practice, $\mathcal{R}$ on real-world CT image residuals is higher than 0.14 because the CT noise is slightly correlated (see Fig. 1c) and because no noise reduction algorithm is perfect.

To evaluate the proposed correlation measure, we have studied the dependence of $\mathcal{R}$ on parameters of noise reduction algorithms for real-world CT scans. The parameter of NLM-2.5D algorithm that directly affects filtering strength is ρ: it affects pixel averaging weights (Eq. 5). For Rank-2.5D this is M: it affects the number of averaged pixels. The results are summarized in Fig. 4.

The graph for NLM-2.5D algorithm (Fig. 4a,b) exhibits bending points near $\log\rho = 8$ and 13. Below these points the correlation measure $\mathcal{R}$ rises slowly. But above these points it rises faster, indicating that the noise reduction algorithm starts to blur essential image details. The change in the derivative can be explained by the fact that noise in CT images has a Gaussian distribution of contrast and low variance, while important image structures have higher variance, but more compact spatial distribution.

Above $\log\rho = 20$ the graph starts to flatten out, indicating that the NLM-2.5D algorithm is degenerating toward a linear blur operator. Our visual assessment indicates that the best value of $\log\rho$ for the tested images is between 10 and 12. This trade-off between amounts of noise reduction and blurring of details corresponds to $\mathcal{R} \approx 0.28 \ldots 0.29$.

The graph for Rank-2.5D algorithm (Fig. 4c,d) does not exhibit this bending point: it steadily rises throughout the usable range of M, indicating that the loss

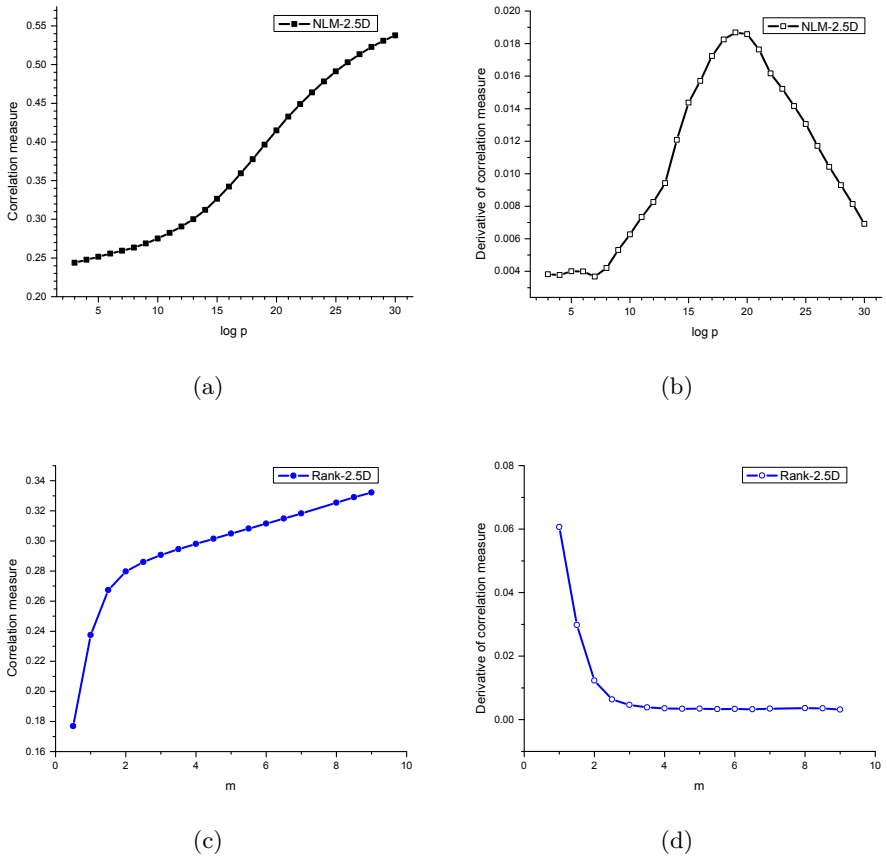


Fig. 4. Dependence of the correlation measure $\mathcal{R}$ and its derivative on filtering strength parameters ρ and M

of structural details happens gradually. Visually best values of M are between 3.5 and 4.5. This corresponds to $\mathcal{R} \approx 0.29 \ldots 0.30$, which is a good match to the "sweet spot" of NLM-2.5D algorithm.

7.2 Entropy Measure

Another possible approach to image filtering quality assessment can be constructed based on entropy as a measure of randomness. Let us consider the residual image $I_r(x, y)$ again. We normalize and quantize $I_r(x, y)$ image (9) as follows:

$$I_N(x, y) = \lfloor \frac{8}{\pi} \cdot \arctan\left(\frac{I_r(x, y) - m}{\alpha\sigma}\right) \rfloor \tag{11}$$

where m and σ are image $I_r(x, y)$ mean intensity and standard deviation respectively and α is a parameter (we use $\alpha = 1.2$ in our experiments). So, the

output image $I_N(x, y)$ consists of no more than 9 different values: $I_N(x, y) = -4, -3, ..., +4$. For low-amplitude residual images there can be less than 9 values, but for large residuals such images can only differ by statistical properties, independently of the residual amplitude. Taking into account that CT images have a typical correlation radius of about $1.5 \dots 2$ pixels we consider the following directional differences:

$$\begin{aligned} D_{0^\circ}(x, y) &= I_N(x, y) - I_N(x, y + 2) \\ D_{45^\circ}(x, y) &= I_N(x, y) - I_N(x + 1, y + 1) \\ D_{90^\circ}(x, y) &= I_N(x, y) - I_N(x + 2, y) \\ D_{135^\circ}(x, y) &= I_N(x, y) - I_N(x - 1, y + 1) \end{aligned} \tag{12}$$

If the residual image $I_r(x, y)$ consists of pure Gaussian white noise (Fig. 1), the probability distribution of differences in (12) is close to uniform and has a range between -8 and 8 (see (11)). But when important details appear in the residual image, such as vessels or organ boundaries, small difference values will appear more frequently in (12). To derive a measure reflecting these properties we consider a joint histogram $H_N(\nu)$ of differences (12) normalized as a probability distribution function and calculate the entropy with a formula:

$$\mathcal{E} = -\sum_{\nu} H_N(\nu) \log_2 H_N(\nu) \tag{13}$$

Since entropy is a measure of chaos, we can expect that when the residual image is low-amplitude the entropy (13) is small too; when more noise gets suppressed, the entropy grows; and finally, when structural details appear in the residual image, the entropy decreases again because correlation of adjacent pixels results in dominance of small difference values in (12).

The results of application of this approach to CT images (Fig. 7) are shown in Fig. 5. The curves for both Rank-2.5D and NLM-2.5D algorithms have a distinct maximum which can be considered as the algorithm parameter value related to the optimal amount of noise reduction. In our experiments the noise reduction strength estimated as maximum of $\mathcal{E}$ was very close to subjectively good values of parameters.

Comparing with the autocorrelation-based approach the advantages of the entropy-based method are:

- similar graph behavior for both algorithms (compare Fig. 4 and Fig. 5);
- visible effect without block-wise weighted averaging (10);

the drawback of the entropy-based approach, in our view, is that the maximal value differs from the surrounding values by only 5% (see Fig. 5) while for the correlation-based approach these differences are more substantial (Fig. 4).

8 Results

Fig. 6a shows the results of a phantom CT image denoising. The Gaussian noise which corresponds to the real noise distribution on CT images was added to the

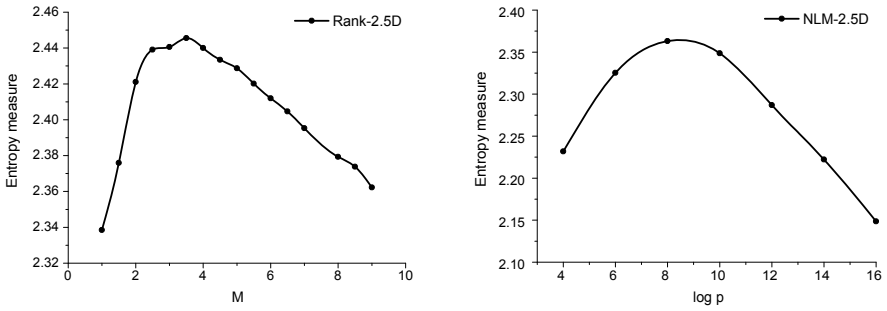
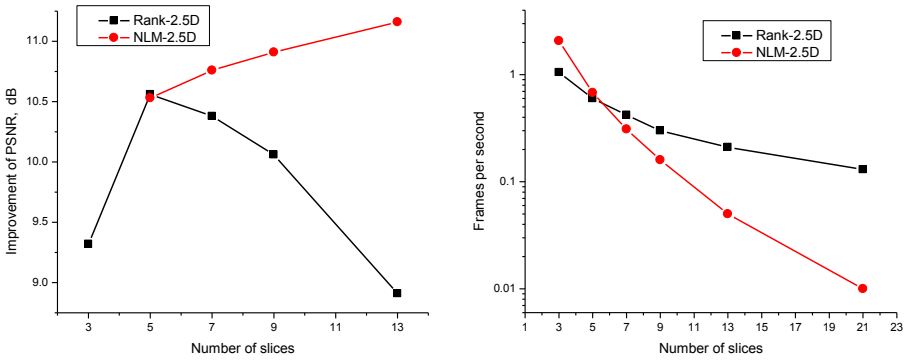


Fig. 5. Dependence of entropy-based measure $\mathcal{E}$ on filtering strength parameters ρ and M



(a) (b)

Fig. 6. (a) Improvement of PSNR for the phantom image filtered by different algorithms, depending on the neighborhood size N; (b) Speed of our C++ implementation of the proposed algorithms on a 2.8-GHz desktop

synthesized phantom slices. The measured value is the improvement of PSNR in decibels between processed images and the ground-truth noise-free image. Each algorithm has been run with optimal parameters which maximize PSNR for each size of the neighborhood. The optimized parameter for Rank-2.5D was M, while the optimized parameter for NLM-2.5D was ρ. The patch size in NLM has been set to 8×8 pixels, while the "pixel" size has been set to 2×2 pixels.

The maximum in the curve for Rank-2.5D algorithm occurs when the neighborhood size is of order of smallest significant object size ("ligaments" and "vessels"). When the neighborhood becomes larger, small objects are suppressed as noise.

It can be seen that the algorithms are able to exploit high degree of correlation between image slices, which is reflected in PSNR and visual quality.

Fig. 6b compares the speed of the proposed algorithms on a CT scan with 515×512-pixel slices. It shows that the Rank-2.5D algorithm has a linear complexity growth depending on the neighborhood size, which makes it suitable for future high-resolution CT scanners. The complexity of NLM-2.5D algorithm grows much faster.

Fig. 7 shows the result of real CT image denoising with Rank-2.5D and NLM-2.5D algorithms operating on a neighborhood of $7 \times 7 \times 5$ pixels (the last '5' being the number of slices). The value of parameter M in Rank-2.5D method has been set to 3.5 for good visual results. This is lower than the value of $M = 5$ which optimized PSNR in our experiments.

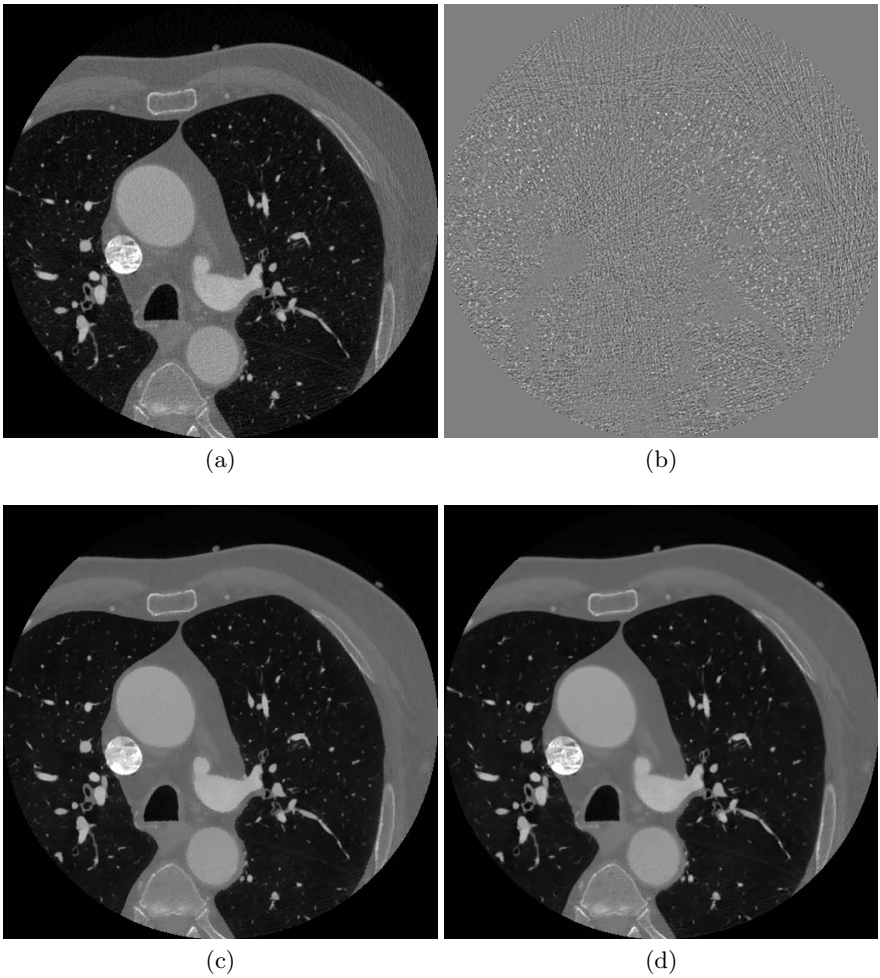


Fig. 7. (a) Original noisy CT image; (b) Residual image for NLM-2.5D algorithm; (c) Result of Rank-2.5D algorithm; (d) Result of NLM-2.5D algorithm

Subjectively, NLM-2.5D approach provides more accurate and clear image while Rank-2.5D leaves some blurred patches in high-contrast areas.

9 Conclusion

Two methods for three-dimensional noise reduction in CT images have been presented. The evaluation shows that they are able to effectively exploit the existing correlation between CT slices for improvement of the resulting image quality. The presented Rank-2.5D shows only moderate growth in computational complexity depending on the size of neighborhood. It is easy to see from Fig. 6b that when the neighborhood size in slices is larger than 6, Rank-2.5D becomes faster than NLM-2.5D. The maximum quality of Rank-2.5D is achieved when the neighborhood size is approximately equal to the minimal useful object size in the image.

Two new methods for no-reference quality analysis of denoised images are proposed and tested on a set of real DICOM images. The first method is based on weighted block-wise autocorrelation coefficient (10). The second one is based on entropy of directional differences (13). Both measures enable automatic selection of algorithm parameters that balance the amount of noise reduction and loss of image details.

References

1. Pratt, W.: Digital Image Processing: PIKS Scientific Inside. Wiley (2007)
2. Yaroslavsky, L., Kim, V.: Rank algorithms for picture processing. ACM Transactions on Graphics (TOG) 35, 234–258 (1986)
3. Huang, T., Yang, G., Tang, G.: A fast two-dimensional median filtering algorithm. IEEE Trans. Acoust., Speech, Signal Proc. 27(1), 13–18 (1979)
4. Weiss, B.: Fast median and bilateral filtering. ACM Transactions on Graphics (TOG) 25(3), 519–526 (2006)
5. Perreault, S., Hebert, P.: Median filtering in constant time. IEEE Transactions on Image Processing 16, 2389–2394 (2007)
6. Storozhilova, M., Yurin, D.: Fast rank algorithms based on multiscale histograms. In: 21st International Conference on Computer Graphics GraphiCon 2011, Moscow, Russia, pp. 132–135 (September 2011)
7. Storozhilova, M., Yurin, D.: Fast rank algorithms with multiscale histograms lazy updating. In: 8th Open German-Russian Workshop "Pattern Recognition and Image Understanding" (OGRW-8-2011), Lobachevsky State University of Nizhny Novgorod, pp. 380–383 (November 2011)
8. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proceedings of the IEEE Sixth International Conference on Computer Vision (ICCV 1998), pp. 839–846 (1998)
9. Buades, A., Morel, J.: A non-local algorithm for image denoising. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 60–65 (2005)
10. Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K.: Image denoising by sparse 3D transform-domain collaborative filtering. IEEE Trans. Image Process. 16(8), 2080–2095 (2007)

11. Reiter, M., Zauner, G.: Denoising of computed tomography images using multiresolution based methods. In: European Conference on Non-Destructive Testing poster (2006)
12. Trinh, D., Luong, M., Rocchisani, J.M., Pham, C., Pham, H., Dibos, F.: An optimal weight method for CT image denoising. Journal of Electronic Science and Technology 10(2), 124–129 (2012)
13. Kijewski, M., Judy, P.: The noise power spectrum of CT images. Phys. Med. Biol. 32(5), 565–575 (1987)
14. Newton, T., Potts, D.: Radiology of the Skull and Brain: Technical Aspects of Computed Tomography. Radiology of the Skull and Brain. Mosby (1981)
15. Putilin, S., Lukin, A.: Non-local means method modification for noise suppression in video. In: 17th International Conference on Computer Graphics, GraphiCon 2007, pp. 257–259 (June 2007) (in Russian)
16. Storozhilova, M., Lukin, A., Yurin, D., Sinitsyn, V.: Two approaches for noise filtering in 3D medical CT-images. In: 22nd International Conference on Computer Graphics, GraphiCon 2012, Moscow, Russia, pp. 68–72 (October 2012)

Implementing Irradiance Cache in a GPU Realistic Renderer

Vladimir Frolov[1,2], Konstantin Vostryakov[2],
Alexander Kharlamov[2], and Vladimir Galaktionov[1]

[1] Keldysh Institute of Applied Mathematics (Russian Academy of Sciences), Moscow, Russia
[2] Nvidia

Abstract. This work presents an approach to integrating irradiance caching (IC) technique in a complete GPU photorealistic renderer. This work proposes a GPU friendly IC solution, where performance critical parts of an irradiance cache algorithm are done completely on the GPU. The modified algorithm for the GPU is different from a traditional implementation in 2 ways. The first distinction is a predictive nature of our algorithm that allows us to insert a large record set at once instead of inserting records one by one, as in traditional approaches. The second distinction is a new heuristic for validity radius computations. We also consider some low-level details and provide performance analysis of our solution.

Keywords: Ray Tracing, GPU Global illumination, Irradiance cache, realistic rendering.

1 Introduction

For the last decade Graphics Processing Units (GPUs) have made a great advance in performance and have become fully programmable processors. Several commercial GPU photorealistic renderers are available today. Most of them use unbiased path tracing methods due to its simplicity for GPU implementation and scalability. This results in tracing up to ten times more rays than biased alternatives. Moreover, path tracing of complex scenes suffers from highly irregular workload and memory access tends to be random. These issues lead to inefficient hardware utilization. On the other hand, although biased approaches have lower complexity they are more difficult to implement on the GPU.

Our paper illustrates the research that we have performed for GPU accelerated biased rendering via irradiance caching and path tracing techniques. The key results of this research are:

A new GPU friendly IC generation algorithm, performed before final render pass. Introduction of a new validity radius computation heuristic while inserting records into octree for fast irradiance interpolation on the GPU.

Our main contribution is a high quality IC solution that provides from 4 to 14 times acceleration (with an average PSNR of 40-45 compared to a GPU accelerated path tracing.

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XIX, LNCS 7870, pp. 17–32, 2013.

2 Related Work

The majority of modern papers on ray tracing and global illumination and either irradiance caching are focused on interactive frame rates and fast computations and are not paying enough attention to the quality of their solution. We discuss some of modern papers in the related field further but we would like to underline that we can't provide a quantity comparison of these papers to our work because most of them does not provide a numerical error analysis in contrast to our solution. So, we provide only quantity comparison between proposed IC approach and our own optimized path tracing implementation.

2.1 GPU Ray-Tracing

Although fast GPU ray tracing for complex scenes is still a challenge we do not focus on ray tracing acceleration in this paper. Aila and Laine work [1] provides comprehensive performance analysis of ray tracing on the GPU. Our ray tracing implementation has approximately the same performance on diffuse rays although it's of 1.5x factor slower for coherent frustum tracing. For "Conference Room", "Sibenik" and other commonly used test scenes our ray tracer achieves ~130M rays per sec for primary rays and ~60M rays per sec for diffuse rays on average (measured on a GTX 560 HW). However we found that depending on the scene, path tracing rays after several bounces path tracing rays can be a factor of 2x-4x slower than diffuse rays – this is the case of poor HW utilization due to random memory access and non-uniform workload.

2.2 Irradiance Cache

Irradiance caching decreases the overall cost of indirect illumination computation by performing full hemisphere sampling (or final gathering) only at selected points in the scene, caching the results, and reusing the cached indirect illumination values between those points through interpolation. It was introduced in [2]. The algorithm can be summarized as follows:

```
if interpolation is possible then
  reuse cached values through interpolation;
else
  compute new value;
  store it in the cache;
end if
```

The number of irradiance cache points is usually of 1 or 2 magnitude order less than the number of pixels – so the irradiance cache is quite efficient and it can greatly speed up the whole rendering. However, irradiance cache is a challenging algorithm, even on a CPU. It has a lot of issues and heuristic approaches that suppress its

artifacts [3]. A well-known irradiance cache algorithm [2,3] cannot be implemented on a GPU in a straightforward way because of its serial nature:

- Trace one ray
- Evaluate and insert one record at given “transaction”.

2.3 Parallel Irradiance Cache

It is difficult to parallelize irradiance cache on multi-core CPUs. To understand what the problem is, consider a situation when we have several rendering threads in the simple algorithm discussed above. In this case, each insertion operation will block a part of cache for other threads until the hemisphere sampling is finished. The comprehensive analysis of parallel IC solutions are described at [4] and [5] and a 2 different approaches were introduced.

Mainly, previous researches in parallel IC were focused on the problem of accessing shared memory on SMP systems (or distributed memory on clusters) and achieving efficiency on all processors. Besides the fact that the problem of effective data access and high processors utilization is very important, this is not the only problem when implementing parallel IC. Another challenge is that the IC construction algorithm is based on a variety of heuristics (neighbor clamping and etc) which were used with a serial IC records inserting [3]. When doing parallel implementation and inserting records in batches we have redundant amount of records with strong overlapping of validity areas and because of that poor IC look-up performance. This problem became critical for GPU with the massive number of threads.

PBRT 2.0 [6] also has multithreaded implementation of IC. It does the first pass to compute the cache and the second one to render final image. This approach needs to be refined for the case of massive parallelism. We discuss it further in our paper.

2.4 GPU Irradiance Cache

GPU irradiance caching was introduced in [7] and described in details in [3]. These papers mainly focus on replacing irradiance interpolation via octree lookups with splatting to avoid traversing hierarchical structures on GPUs. The approach used in [7] can be used for primary rays or interactive visualization in computer games, however, it has one serious limitation: only one light bounce can be evaluated either for hemisphere sampling or for final rendering. Thus, it will be hard to have precise photorealistic result with this approach. Besides, it was done mainly for rasterization based engines and cannot be combined directly with a GPU path tracer.

Wang et al. [8] presents an efficient approach for global illumination using photon mapping on the GPU. The key aspect of this work is to use irradiance cache with photon mapping and final gathering [9] to quickly compute smooth indirect illumination. Direct lighting is computed using ray tracing and supports hard shadows from point light sources. In this paper irradiance cache point positions are predicted from the geometry discontinuities. Wang’s approach to build IC was combined with path tracing in [10] to focus on rendering images with glossy reflections and shadows

offline. However, both of these approaches work with geometry term and they use predictive nature without further refinement.

The radiance hints method introduces in [11] is a stable (for animation) and a fast solution for diffuse global illumination. The method is based on grid based radiance caching with reflective shadow maps and can handle multiple light bounces. This method works for interactive rendering with view-independent algorithm so it can't control image error that is required for photorealistic rendering. Besides, using regular grid will not allow one to have high precision with reasonable memory consumption.

3 Suggested Approach

Similar to PBRT 2.0 our algorithm consists of 2 main phases. The first phase is "irradiance cache creation" and the second phase – "final rendering". The goal of the first phase is to generate a set of irradiance cache points that will completely cover the space where future samples can occur. This separates computing irradiance cache from using it and thus we compute the only part of 3D irradiance signal (via computing IC) that is essential for the final image.

The key novelty points of our IC construction are:

1. Usage of 2 different algorithms for directly and indirectly visible surfaces.
2. Image processing algorithms to predict where more IC records should be placed and where we can have only several of them.
3. Z-curve clusterization of irradiance evaluation queries.
4. Adjusting IC records validity radiuses using a new clamping heuristic to get an faster irradiance interpolation.

3.1 Creation of Irradiance Cache

IC creation process consists of multiple passes (10-20 passes, the maximum number is user controlled). It can be summarized in the following pseudo code:

```
procedure Create_IC(ic : out Irradiance_Cache) is

  geomDiscMap  : Image;
  irradDiscMap : Image;
  temp         : Image;
  discMap      : Texture2D;
  candidates   : array of IC_Record;
  smallGroup   : array of IC_Record;
  candGroups   : array of (array of IC_Record);
  iterNum      : Integer;

  -- user controlled
  MAX_PASS_NUMBER   : Integer := 20;
  MIN_CAND_TRESHOLD : Integer := 100;
```

```
begin

  -- the very first pass
  --
  geomDiscMap := CreateGeometryDiscMap();
  discMap     := Build2DMipMapChain(geomDiscMap);
  candidates  := Dithering(discMap);

  ic.Insert(candidates);
  iterNum := 0;
  candidates.resize(MIN_CAND_TRESHOLD+1);

  -- multipass construction
  --

  while(candidates.size() >= MIN_CAND_TRESHOLD and
        iterNum < MAX_PASS_NUMBER):

    -- screen space stage
    --
    temp       := CreateIrradianceDiscMap();
    discMap    := Build2DMipMap(temp);
    candidates := Dithering(discMap);

    -- insert candidates except for pixels
    -- for which we already have records
    --
    ic.Insert({candidates} \ {ic.records});

    -- world space stage
    --
    candidates := SelectIfInterpErrorIsLarge();
    candidates := SortWithZCurve(candidates);
    candGroups := GroupRecords(candidates);

    candidates := []
    for group in candGroups:
      smallGroup := SelectSeveralCands(group);
      candidates.append(smallGroup);
    end for;

    ic.Insert(candidates);
    iterNum := iterNum + 1;
  end while;
end Create_IC;
```

The ‘Insert’ procedure also evaluates irradiance for each records. We will discuss its implementation later. Each pass consists of 2 independent stages. The first stage works only for visible points. The second stage works for visible and for points that are not directly visible from the eye. During each pass and within each stage new irradiance cache records are inserted into the cache. The very first pass is different from the others and works with geometry discontinuity like [8] and [10] do. The important aspect of the irradiance cache generation process is that a large set of points (several hundred or even thousand points) are selected at once, irradiance for these points are computed on the GPU and these points are added to the cache structure in one transaction.

3.2 The Very First Pass (Coarse Screen Space Pass)

In the very beginning of the process of irradiance cache creation we have no information about the scene at all. Owing to this fact, the goal of this pass is to create first approximation of irradiance cache that will be used as a starting point for further passes. Our algorithm tries to predict complexity of different screen parts, using geometry discontinuity maps and image processing. It attempts to place more records in areas where more of them are required.

First, we trace rays from the eye position and store hit positions and normals in separate full screen textures. A mip-map pyramid for each of these textures is built. Then, for each mip-level a surface discontinuity texture map is evaluated according to this formula:

```
surfDisc := k*normDiff + worldPosDiff;
```

Where worldPosDiff and normDiff is a maximum difference between positions (and normals accordingly) within neighboring pixels (in an appropriate mip-level). And k is a parameter that depends on the world scale. Having a discontinuity map, we blend all of up-scaled mip levels and perform the dithering algorithm on the resulting image with a binary quantization (i.e. each pixel in resulting image can have a value equal to 1 or 0). The result of this dithering is a binary image - a set of initial points; this initial set is our first approximation of irradiance cache.

The main idea behind that binary dithering is that it allows us to represent discontinuity maps (both geometry and irradiance) in terms of sparse point set – potential IC records.

Dithering Implementation.

Here is our dithering implementation in pseudo code:

```
-- in parallel for each screen pixel
--
d0 := tex2DLod(discontinuty, x, y, 0);
d1 := tex2DLod(discontinuty, x, y, 1);
```

```
d2 := tex2DLod(discontinuty, x, y, 2);
...
deterministicSelect := F1(d0,d1,d2,...,x,y);
stochasticSelect    := F2(d0,d1,d2,...,x,y);

if deterministicSelect or stochasticSelect then
  PutRecordInThisPixel(x,y);
end if;
```

For each pixel we read discontinuity value from each mip level of the discontinuity map - (call these values d0,d1,d2,d3,...) and then rely on some functions (F1 and F2) - we decide whether we put record in the target screen pixel or not. We believe the F1 and F2 implementation could be different. We used a set of thresholds for F1 and a stochastic selection with fixed threshold for F2.

Screen Space Stage (1).
The same operation but for irradiance discontinuity is performed in subsequent passes– calculate irradiance by fetching it from the cache, build difference image, create mip-map pyramid, perform dithering and insert newly obtained points into irradiance cache. This procedure is repeated several times.

World Space Stage (2).
The presented screen space algorithm works on primary visible smooth surfaces. However it cannot be used for indirectly visible surfaces and it tends to miss tiny geometric details. The red triangle in Fig. 1 represents viewers' frustum. Red points are not visible from the camera, yet secondary rays can still reach such regions.

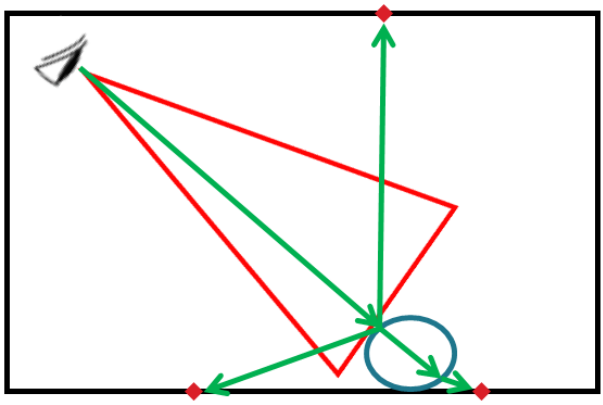


Fig. 1. Rays that are traced from the eye may reach points that are not visible directly

Our goal is, to generate a set of irradiance cache points that completely cover the space where rays can hit a surface during path tracing process. We used an idea similar to the clustering approach that had been used in [12].

Rays are traced from the eye and all hit points are saved on each bounce if interpolation has unacceptable error estimation from geometric considerations. All such points are stored in separate buffer during ray tracing using CUDA 'atomicAdd'.

However, a very large set of points is produced and we have to select a subset of the best candidates from it. We believe several approaches can be applied to form clusters, we used a simple technique that can be easily ported to GPU. At first, we sort candidates according to 3D Z-Curve [13]. After that, we start inserting points into a cluster (around the first point in the sorted array). However we want to keep the bounding box of the current cluster within certain limits. If after inserting a point the bounding box of the current cluster exceeds the maximum bounding box size, we create a new cluster and continue inserting candidates into it. Our motivation to use clustering was to select candidates with maximal interpolation error and this way IC records will be more regularly distributed.

Once all the points are clustered, it is possible to select a single point from each cluster with maximum error however it is not the best approach. Let us consider a cluster that was formed around the corner of the Cornell Box. The corner consists of 3 walls and if our cluster contains points on every wall, we have to select at least one point on each wall, otherwise we lose useful candidates, because we know that radiance difference usually corresponds to rapid changes of the normal field. So, from each cluster we select several candidates with unique normals and thus, deal with corner cases.

We terminate the creation process when the maximum number of passes is reached or when 'candidates.size()' becomes small enough (this parameter is user defined). Due to the stochastic nature of our IC creation process (world space stage uses random 'path tracing style' rays) on some complex scenes there can be a regions that were not covered by IC records and candidates are still produced. However, if we stop IC creation process in that case, it will not introduce a significant error in the final image because the probability that rays hit such regions tends to be zero.

3.3 Final Rendering

After we have irradiance cache computed we do adaptive path tracing as described in [10] with fetching indirect smooth lighting from the irradiance cache. Thus, for fast and smooth indirect lighting we use irradiance caching technique and we use path tracing for other effects, such as soft shadows, glossy reflections and refractions, depth of field and motion blur.

3.4 Implementation and Results

Our implementation is done using CUDA and C++. All performance critical parts of the algorithm are done in CUDA. However, such things as tree construction and some other algorithms are implemented in C++ on the host.

3.5 Hemisphere Sampling

For irradiance computations we use the progressive evaluation algorithm with Monte Carlo path tracing. All irradiance cache records are placed in 'active records list'. For each active record we use a sequence of randomly distributed (but coherent) hemisphere samples – 1024, 4096, 16384, 65536 and etc. At first we use 1024 rays for all records in the list. If estimated error for a record is small enough we discard that record from 'list of active records' and process remaining records with 16384 rays (the next value in the sequence). We repeat this process until all records are evaluated or the maximum number of samples per irradiance record is achieved. Using the sequence of pre-generated samples instead of simple random rays is important because we can save rays coherency at least for the first bounce and have valuable speed-up (~ 2-4 times) for ray tracing on GPUs.

To evaluate convergence for a record we use the approach described in [14] accumulating odd and even partial sums of lighting integral. We used 'Hammersley' sampling technique described in [15] to cover hemisphere with samples. To have more coherent groups of rays we used stratification (subdivide the hemisphere into sectors and generate 32*k rays for each sector where k >= 1). We group 'Hammersley' points in groups of size 32*k. We do that once on the CPU in tangent space. On the GPU we transform directions from tangent to object space to get correct hemisphere sampling. During the hemisphere sampling, we also calculate initial validity radius for each irradiance cache point using harmonic mean distance [Ward et all 88].

3.6 Insertion Records in Octree

The insertion process is done on the CPU. Our implementation inserts a set of records in one transaction, and we modified the original insertion algorithm, described in [3] and [6].

```
procedure Insert ( self    : inout Irradiance_Cache;
                   records : in array of IC_Record ) is
begin
  EvaluateIrradiance(records);
  self.auxOctree.Insert(records);
  ValidityRadiusClamping(records, self.auxOctree);
  NeighbourClamping(records, self.auxOctree);
  self.mainOctree.Insert(records);
end Insert;
```

The problem with inserting multiple records is that in several cases, we can find a large set of closely-located records, with overlapping validity radiuses. This is a problem, because in those regions octree leafs will contain a large list of points and interpolation becomes slow. This motivated us to develop a special algorithm for decreasing validity radiuses during insertion. Our insertion consists of 3 phases. First, we consider irradiance cache records as points (not as spheres!) and insert them into an auxiliary octree. This octree will be used to speed-up location of k-nearest points (irradiance cache records).

We call the second step ‘validity radius clamping’. It treats irradiance cache records as spheres. The goal of this step is to decrease validity radius for each point. For each irradiance cache record it locates k nearest neighbors (k is 4-7) in ‘different directions’ and if the validity radius of the current point is greater than the distance to the farthest point, the validity radius is clamped to this distance.

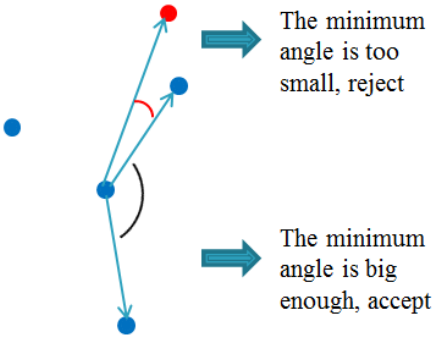


Fig. 2. Angular criterion of filtering nearest neighbors

By ‘different directions’ we mean that while we look for neighboring points we calculate the angle between a new candidate and all the points that we have already found (Fig. 2). If the angle between the direction to a new point and any direction to a point we already found is too small, we do not consider this point, i.e. we do not add it into the nearest neighbors list.

At last we also consider irradiance cache records as spheres. But validity radiuses of these records were clamped by the previous step. The goal of the last step is to insert all points into the final octree that will be used for fetching irradiance from the cache on the GPU. Validity radius clamping is an important part of the algorithm. Table 1 shows performance improvements gained by introducing our validity clamping approach.

Table 1. The column marked IC1 time presents time (in milliseconds) required to perform one million look-up operations when validity radius clamping is disabled. The column marked IC2 time presents time, required to perform one million look-up operations with enabled validity radius clamping. The last column represents acceleration factor. All measurements were done with GTX560 HW. The original Krivánek’s Neighbor Clamping was performed in all cases - both for IC1 and IC2 columns.

Scene	IC1 time (ms)	IC2 time (ms)	Look-up acceleration
Teapot	30.0 ms	5.8 ms	5.1
Sponza1	59.0 ms	13.0 ms	4.4
Sponza2	44.0 ms	9.1 ms	4.9
Reneissance	20.0 ms	9.6 ms	2.1
Arch-class	53.0 ms	12.0 ms	4.3
Cry-Sponza	45.0 ms	13.0 ms	3.3

Thus, inspired by Krivánek's Neighbor Clamping, we introduce a new validity clamping radius criterion in order to accelerate look-up operation by means of density control. To avoid light leaks near tiny geometry details we use original Neighbor Clamping heuristic, extended to the case of simultaneous multiple record insertion.

3.7 Fast Octree Look-Up

We use interpolation formula proposed by Tabellion and Lamorlette in [16] and stackless octree look-up as described in [3]. We have found that the stackless approach is very efficient on GPUs if only several irradiance cache records are stored in octree leaves (the timings are presented in table 1).

3.8 Results Overview

The results of our renderer are presented in Table 2. We target high quality images at 1920x1200 resolution and we used world space irradiance interpolation. We perform gamma-correction for the final images and a Tone Mapping for several of them (marked in table).

Table 2. Test setup. All scenes were rendered in 1920x1200 on GTX560 HW. Image difference and square error were computed with 'The Compressonator' tool [17]. The path tracing accuracy was always set to 5%. Note that path tracing accuracy should not correlate to the IC accuracy except for the cases where the lighting is pure diffuse indirect (i.e. interreflected from diffuse surfaces). The naive path tracing accuracy (as reference method) was set to 5%. Maximum diffuse bounces of path tracing was limited to 2.We used accuracy = 10% for IC records evaluation. The column named 'Rays per sec/Samples per sec' show the number of rays per second and samples per second that our renderer performs for naive path tracing. ** Our renderer shoots 4 shadow rays per each hit point if only one area light is present in the scene. So the difference between number of rays per second and number of samples per second is higher for this scene.

Scene	Triangles number	Rays per sec / Samples per sec;	IC records	IC time	Render pass time
Teapot	25 612	82M / 5.8M**	16994	9.0s	133.7s
Sponza1	66 454	62M / 5.4M	213435	157s	20s
Sponza2	66 454	64M / 5.4M	97789	109s	11s
Reneissance	871 786	64M / 4.3M	306326	167s	10s
Arch-Class	1 010 724	45M / 4.1M	302992	285s	20s
Cry-Sponza	262 267	30M / 2.8M	442332	501s	40s

Scene	Total time (with IC)	Naive path tracing time	Accelerati-on (times)	Square error (hdr) / (png)	PSNR (png)
Teapot	142.7s	510s	3.6	2.3 / 2.1	46.5
Sponza1	177s	1554s	8.8	1.3 / 2.1	46.4

Table 2. *(Continued)*

Sponza2	120s	1695s	14.1	1.6 / 3.6	41.6
Reneissance	177s	1830s	10.3	1.7 / 2.9	43.5
Arch-Class	305s	2001s	6.6	2.1 / 4.3	40.2
Cry-Sponza	541s	3347s	6.2	2.9 / 2.5	45.0

Scene	Avg samples per pixel for Naive Path Tracing	Avg samples per IC record	CPU overhead (octree build)	Tone Mapping
Teapot	1242	5069	7%	No
Sponza1	3644	4840	19%	No
Sponza2	3972	5398	5.6%	Yes
Reneissance	3416	3156	16%	No
Arch-Class	3562	6432	8%	No
Cry-Sponza	4068	5328	9%	Yes

We also provide zoomed-in side by side comparisons of our IC and path tracing solutions by taking the time we compute image with IC and see what image we will get with path tracing.

3.9 Bottleneck Analysis

For our current implementation we used Monte-Carlo path tracing to evaluate irradiance. We limit number of diffuse light bounces to 2 in naive path tracing and IC. We start from 1024 hemisphere samples and continue to increase the number of samples with our progressive evaluation algorithm. The average saples per IC record is presented in table 2.

We have measured that during irradiance cache construction ~80% of the time is spent on evaluating records (i.e. hemisphere sampling with Monte-Carlo path tracing) and it takes ~50-90% of the total rendering time. One way to reduce this time is to use fewer samples with one bounce. This will work much faster because on the first bounce we have coherent sets of rays (and the noise is less than for 2 or more bounces). However this will prevent us from computing indirect lighting from multiple diffuse bounces. Another choice is to use photon mapping with final gathering [3] instead of Monte-Carlo path tracing. We believe this idea should give us a performance benefit and we'll investigate this in our future research. We also think that using recursive irradiance cache [3] is a promising idea; it allows tracing only coherent set of rays to transport light from one level of the cache to another (or even use rasterization).

We perform interpolation in the world space and as a result the IC generation algorithm places a lot of records near tiny geometry details. We suppose screen space IC should be used for primary visible points.

The octree construction (insertion of records) is not a bottleneck in our implementation; it usually takes 7-15% of irradiance cache construction time. The remaining 5% of the time was spent on ray tracing during IC construction, building of discontinuity maps (both geometry and irradiance), data transfers between GPU and CPU.

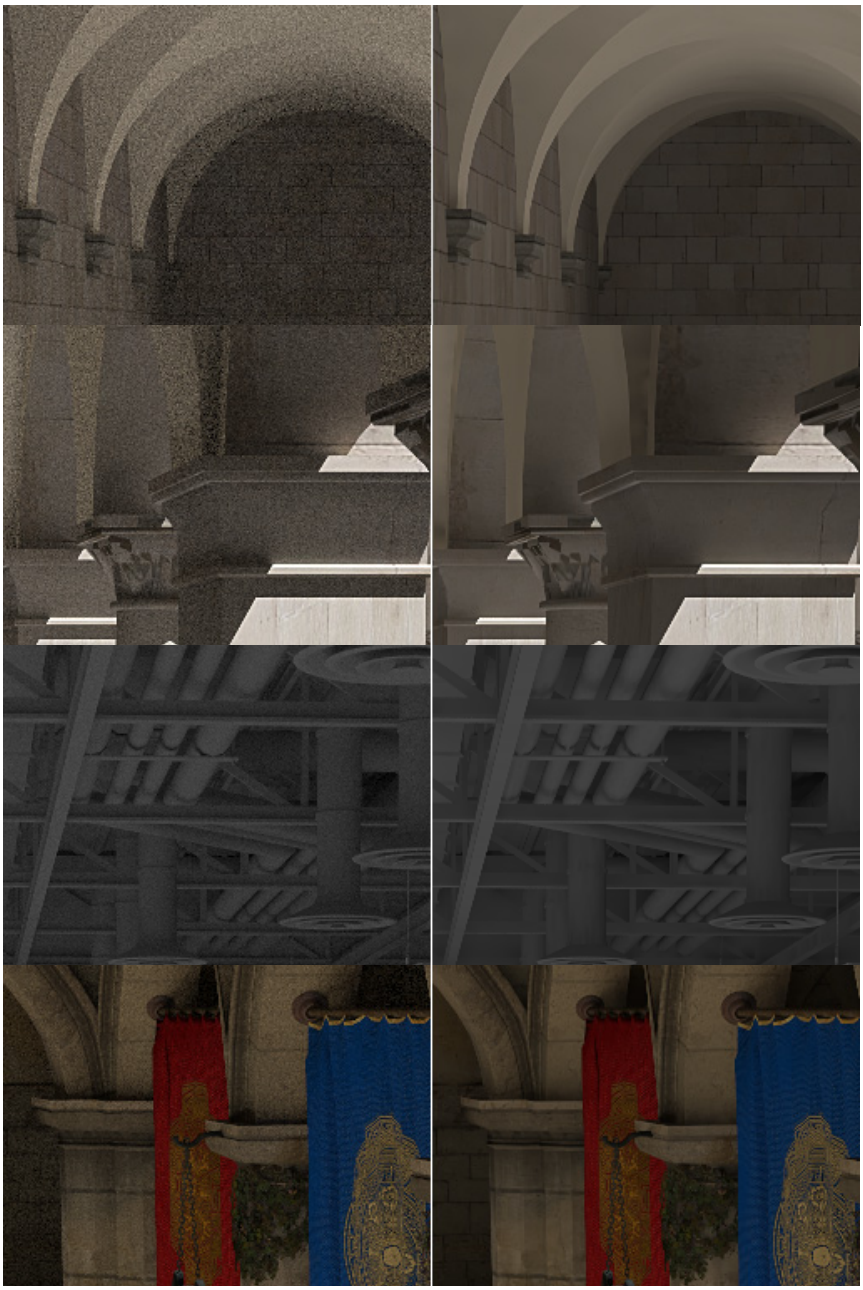


Fig. 3. Zoomed in comparison (256x128 pixels) of the path tracing and irradiance caching within the same time limit

Fig. 4. Test setup. The difference is magnified 16 times

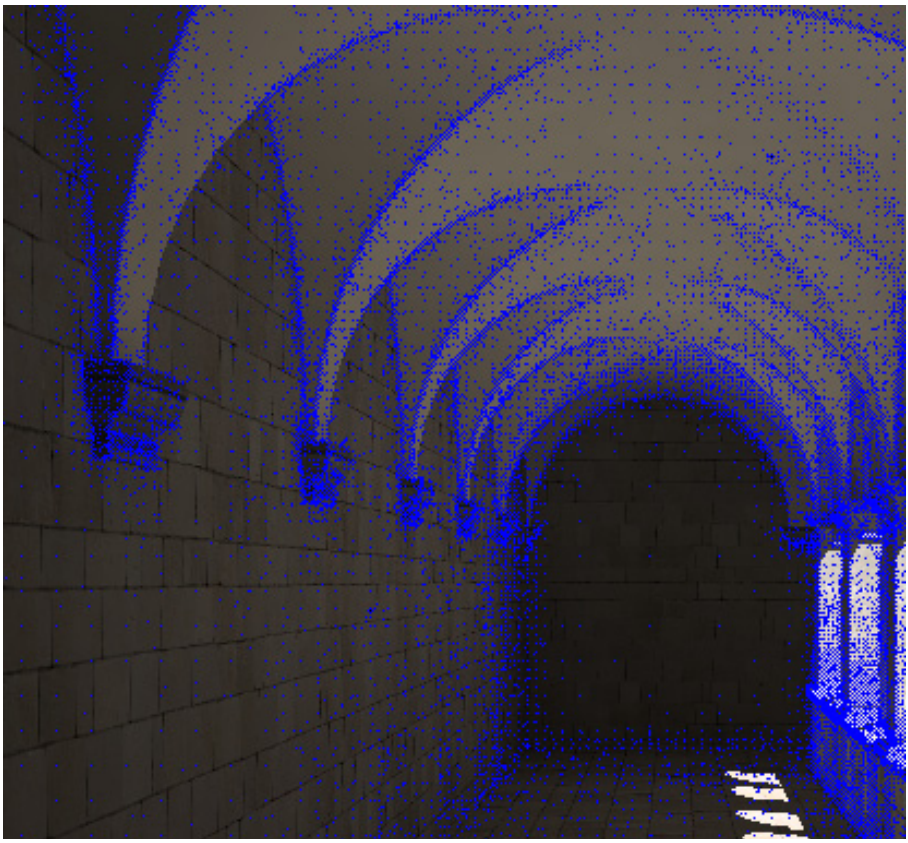


Fig. 5. IC record positions. Image was rendered in 1024x768

Regarding the final rendering, the irradiance cache look-up operation on average takes less than 15% of the total rendering time. The ray tracing (incoherent rays) takes 65-75% of this time and the rest is taken by shading and pipeline overhead.

3.10 Conclusion

In this work we provided an IC solution for a realistic GPU renderers. We achieved from 4 to 14 times acceleration over our own GPU optimized path tracing with an average PSNR 40-45. Instead of inserting records one by one like in traditional IC approach, we used image processing algorithms to predict IC records distribution and thus inserted large sets of IC records in parallel. We used clustering approach to distribute IC records more regularly and put records in places with maximum error. Also, we introduced clamping heuristic for validity radii in order to solve the problem of overlapping validity areas of IC records. Solving that problem was critical for efficient implementation on massive parallel machines like GPUs when records were inserted in large sets. That way we achieved fast IC look-up.

Acknowledgements. This work was sponsored by the RFFI (Russian Foundation for Fundamental Investigations), grants “MOL_A 12-01 31027” and 12-01-00560.

References

1. Aila, T., Laine, S.: Understanding the efficiency of ray traversal on GPUs. In: Proceedings of the Conference on High Performance Graphics 2009, New Orleans, Louisiana, August 1-3. S.N. (2009)
2. Ward, G., Rubinstein, F., Clear, R.: A ray tracing solution for diffuse interreflection. In: SIGGRAPH 1988. Computer Graphics Proceedings (1988)
3. Křivánek, J., Gautron, P., Ward, G., Jensen, H.W., Christensen, P.H., Tabellion, E.: Practical global illumination with irradiance caching. In: ACM SIGGRAPH 2008 Classes, Los Angeles, California, August 11-15, pp. 1–20. ACM, New York (2008), http://doi.acm.org/10.1145/1401132.1401213
4. Debattista, K., Santos, L.P., Chalmers, A.: Accelerating the irradiance cache through parallel component-based rendering. In: EGPGV 2006 - 6th Eurographics Symposium on Parallel Graphics Visualization. Eurographics, pp. 27–34 (May 2006)
5. Dubla, P., Debattista, K., Santos, L.P., Chalmers, A.: A wait-free shared-memory irradiance caching. IEEE Computer Graphics and Applications (2010) (accepted for publication)
6. Pharr, M., Humphreys, G.: Physically Based Rendering: From Theory to Implementation. Morgan Kaufmann (2010)
7. Gautron, P., Křivánek, J., Bouatouch, K., Pattanaik, S.: Radiance cache splatting: A GPU-friendly global illumination algorithm. In: Proceedings of Eurographics Symposium on Rendering (June 2005)
8. Wang, R., Zhou, K., Pan, M., Bao, H.: An efficient GPU-based approach for interactive global illumination. ACM Trans. Graph. 28(3), 1–8 (2009)
9. Jensen, H.W., Suykens, F., Christensen, P.H., Kato, T.: A Practical Guide to Global Illumination using Photon Mapping. In: SIGGRAPH 2002 Course Note #43, San Antonio, USA, July 21-26. ACM (2002)
10. Papaioannou, G.: To be presented at High Performance Graphics 2011, Vancouver, Canada (August 2011)
11. Frolov, V., Kharlamov, A., Ignatenko, A.: Biased solution of integral illumination via irradiance caching and path tracing on GPUs. Programming and Computer Software 37(5), 252–259 (2011), doi:10.1134/S0361768811050021
12. Gassenbauer, V., Křivánek, J., Bouatouch, K., Bouville, C., Ribardière, M.: Improving Performance and Accuracy of Local PCA (November 4, 2011), doi:10.1111/j.1467-8659.2011.02047.x
13. Morton, G.M.: A computer Oriented Geodetic Data Base; and a New Technique in File Sequencing. Technical Report. IBM Ltd., Ottawa (1966)
14. Krivánek, J., Bouatouch, K., Pattanaik, S., Žára, J.: Making Radiance and Irradiance Caching Practical: Adaptive Caching and Neighbor Clamping
15. Suffern, K.: Ray Tracing from the Ground Up. A. K. Peters, Ltd., Natick (2007)
16. Tabellion, E., Lamorlette, A.: An approximate global illumination system for computer-generated films. In: Proceedings of SIGGRAPH (2004), doi:10.1145/1186562.1015748
17. The Compressonator. A tool for compressing textures and creating mip-map levels. Can visualize image difference and calculate square error, http://developer.amd.com/tools/compressonator/pages/default.aspx

Adaptive Generation of Color Anaglyph

Elena Patana, Ilia Safonov, and Michael Rychagov

Samsung Moscow Research Center, 12, bldg. 1, Dvintsev str., Moscow, Russia
{e.patana,ilia.safonov,michael.rychagov}at.samsung.dot.com

Abstract. Nowadays stereophotography is rapidly developing, providing a plenty of sources for stereoimages. The goal of current technology – to provide users with possibility to get high quality 3D anaglyph prints for education and entertainment. To do so it is necessary to agree color characteristics of glasses and printed colors, since errors in color transmission lead to cross-talk interference and ghosting effects. There is no easy way for user to adjust colors of anaglyph in order to coordinate characteristics of glasses and printer.

We propose a technique that allows generating anaglyphs with colors adapted to given glasses and printer colors by means of special color pattern analysis. In addition, our approach takes into account the size of the printed anaglyph image. Resulting printed images have a good quality that is confirmed by user opinion survey. The images contain fewer artifacts and look better in comparison to anaglyphs without adaptation, which are generated in existing software applications. The technique utilizes a low amount of memory and has low computational complexity.

Keywords: Anaglyph, stereo printing, crosstalk noise, ghosting reduction.

1 Introduction

At present time, there are a lot of sources of stereo images: 3D cameras, 3D movies, stereo-pairs can be created from two frames captured by conventional 2D camera, several software technologies allow to catch different views of the same 3D object, for example, Google Earth and parallax effect in HTML 5. Examples of stereoimage sources are shown in figure 1.

Presentation of three-dimensional views by anaglyphs is one of the simplest and the most economic methods. However this method has some disadvantages such as loss of color and discomfort for prolonged viewing. In spite of the drawbacks, sometimes consumers want to print 3D color anaglyph pictures for education and entertainment. 3D anaglyph books have been known for several decades, they are viewed through glasses with different color filters, for example blue and red. Recently a famous magazine SLINK (Issue 7, Sep. 2012) has printed all photos as anaglyph images. There are various user groups in Web like Flickr and others [10], where the process of 3D color anaglyphs generation is discussed [5].

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XIX, LNCS 7870, pp. 33–47, 2013.

There are several PC and mobile software applications for generation of anaglyphs, for example StereoPhoto Maker, Anaglyph Maker, Anaglyph, Anaglyph Workshop, Z-Anaglyph. They have some disadvantages: settings for ghosting effect reduction do not take into account size of the printed anaglyph; there is no adaptation for given glasses and colors of printer ink/toner.

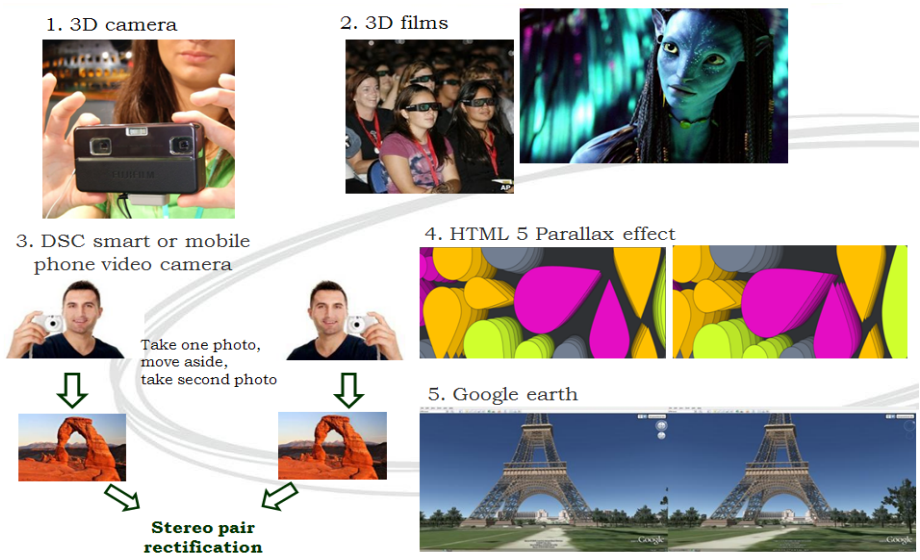


Fig. 1. Examples of stereoimage sources

Usually it is assumed that viewing anaglyph on a display and adjusting of printer color profile is enough to get similarly looking printed anaglyph. However, our experiments show that it is impossible to get similar color perception while looking at a display and color hardcopy produced by a laser or ink-jet printer. From a theoretical point of view it can be easily explained, because gamut of a display and gamut of printing devices are rather different [3]. That is why printing devices manufacturers are interested in rising of printing quality of anaglyph [11].

After performing the measurements of transmission coefficient for several glasses ($\tau_r(\lambda)$ for right filter and $\tau_l(\lambda)$ for left filter) as well as reflection spectrum of printed colors, i.e. magenta $g_M(\lambda)$ and cyan $g_C(\lambda)$, by spectrometer, and analyzing estimated reflection and transmission coefficients, it was concluded that full elimination of cross-talk effect is impossible. Nevertheless, it is possible to reduce cross-talk interference by correct color setting of a printed anaglyph. As it is evident by experiment, the primary printed colors (cyan and magenta) aren't transmitted well; but our estimations show that colors selection according to given glasses allows to decrease ghosting artifacts significantly.

In the paper we propose an adaptive to transmission characteristics of glasses and printed colors algorithm for anaglyph generation and printing.

2 Related Works

Anaglyph generation is a difficult problem. It isn't enough to put one color channel in a left image and another in a right image. Contradictory problems should be solved in anaglyph generation process, because it is needed to code two images on a single view by colors for stereo effect making and to reproduce colors with maximum naturalness. Due to stereo and color conflicts it appears to be impossible to develop an algorithm for anaglyph creation which always produces good color representation, good details, and is permanently free from typical artifacts, such as ghosting, and region merging.

Well-grounded techniques for color anaglyph generation for display is described in [2,4,6,9]. Several techniques have been proposed for the production of anaglyphs for viewing on displays. In [9] three approaches are discussed: Photoshop algorithm (PS) and its variants, the least squares algorithm (LS) proposed by Eric Dubois, that optimizes colors in the CIE space, and the midpoint algorithm (MID) that minimizes the sum of the distances between the anaglyph color and the left and right eye colors in CIE L*a*b*. Linear anaglyph algorithms will always map several different left/right eye colors to the same RGB color. The results show that the MID method produces excellent color and detail for color images but may suffer severe ghosting. Anaglyphs produced by the LS method are normally darker with less detail and require brightening or gamma correction but appear to have no ghosting. The PS method is easy to implement and works well for grayscale images but may also suffer from ghosting and poor color representation.

In [4] several methods for anaglyph enhancement are proposed that rely on stereo image registration, defocusing and nonlinear operations on synthesized depth maps. These enhancements substantially reduce unwanted ghosting artifacts, improve the visual quality of the images, and make comfortable viewing of the same sequence possible in three-dimensional as well as the two-dimensional mode.

The method for computing pixel colors in anaglyph images presented in [6] depends upon knowing the RGB spectral distributions of the display device and the transmission functions of the filters in the viewing glasses. It requires the solving of a nonlinear least-squares problem for each pixel in a stereo pair and is based on minimizing color distances in the CIEL*a*b* uniform color space.

The method proposed in [2] is adapted to the spectral absorption curves of the left and right filters of the anaglyph glasses. A projection technique is used to compute the anaglyph image that yields an image pair (after the glasses) as close as possible to the desired stereo pair. In order to generate anaglyph it is necessary to move into the XYZ space by transition matrix:

$$[C]_{kj} = c_{kj} = \int \bar{p}_k(\lambda)\, d_j(\lambda) d\lambda,$$

where $\bar{p}(\lambda)$ - color-matching function, $d(\lambda)$ - spectrum of standard illuminant, $k = 1,2,3$ – indexes for each color component (red, green, blue), $j = 1,2,3$ – indexes for each color component (red, green, blue), λ – wave length. Reflection light from an

image passes through color filters of an anaglyph glasses and is transformed by two transition matrixes:

$[A_l]_{kj} = a_{lkj} = \int \bar{p}_k(\lambda) d_j(\lambda) f_l(\lambda) d\lambda$ (for left eye filter) and

$[A_r]_{kj} = a_{rkj} = \int \bar{p}_k(\lambda) d_j(\lambda) f_r(\lambda) d\lambda$ (for right eye filter), where $f_l(\lambda)$ and $f_r(\lambda)$ - transmission function of left and right filter.

An anaglyph is generated by the following formula:

$$\widehat{V}_{an}(x) = N(R^T W R)^{-1} R^T W C_2 V(x),$$

where $V(x) = [V_{l1}(x)\ V_{l2}(x)\ V_{l3}(x)\ V_{r1}(x)\ V_{r2}(x)\ V_{r3}(x)]^T$,

where $V_l(x) = [V_{l1}(x)\ V_{l2}(x)\ \ V_{l3}(x)]^T$ – left image of stereo pair in RGB, $V_r(x) = [V_{r1}(x)\ V_{r2}(x)\ \ V_{r3}(x)]^T$ – right image of stereo pair in RGB, N is normalizing matrixes for condition $\widehat{V}_{aj} \in [0,1]$, $N = diag\left(V'_{aj}/\widehat{E}_{aj}\right)$, $j = 1,2,3$ – indexes for each color component (red, green, blue), V'_a - possible anaglyph vector with maximum values which is needed for normalizing of $\widehat{V}_a$, $V'_a = [1\ 1\ 1]^T$, $\widehat{E}_a = (R^T W R)^{-1} R^T \mathrm{W} C_2 E$, W - is weighted matrix, which allows weighting of the Y component more heavily than X and Z to favor reproduction of the correct luminance, $R = \begin{bmatrix} A_l \\ A_r \end{bmatrix}$, $C_2 = \begin{bmatrix} C & 0 \\ 0 & C \end{bmatrix}$.

3 Adaptive Anaglyph Generation

3.1 General workflow

In general we generate anaglyphs by means of the method described in [2], but we propose new approach for adaptation to spectral characteristics of stereo glasses and colors of ink/toner. In additional we propose several procedures to reduce ghosting effect to minimum.

General scheme of an adaptive anaglyph generation is presented on figure 2. Before anaglyph generation a process of transmission function estimation is performed. For this purpose firstly we print the color pattern (figure 2) on a target printer, then we estimate transmission coefficients of given glasses. Detailed description of test color pattern is present in section 3.3; and a sequence of actions for evaluation of transmission function of glasses filters by user is described in section 3.4.

Then it is needed to prepare a stereo pair. Preparation includes geometrical aligning, color correction and enhancement. Stereo pair color correction is performed by well-known method of histogram matching [5]. After that the stereo pair is enhanced to reduce unwanted artifacts.

There are two main ideas for anaglyph enhancement: decreasing of disparity range of a stereo pair and color component defocusing. Detailed description of these ideas is present in section 3.2. Then anaglyph generation is performed by method [2] with adapted transmission functions.

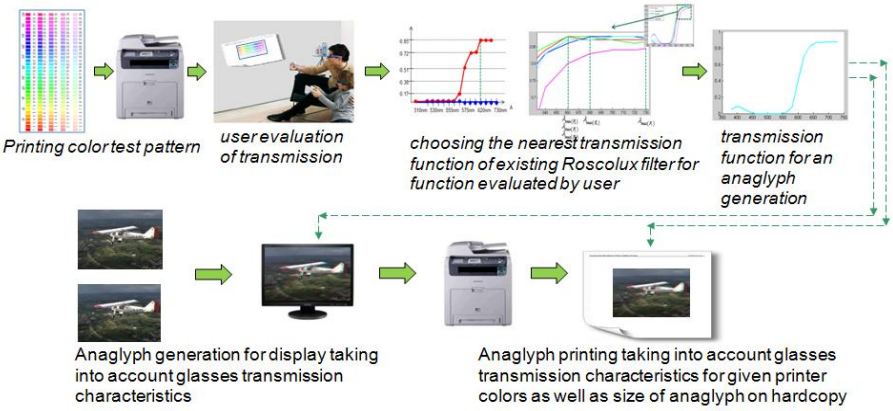


Fig. 2. General scheme of an adaptive anaglyph generation

Also we propose a method of disparity correction for keeping 3D effect and reduction of ghosting effect appearance on an anaglyph hardcopy, because anaglyph size can be different on display and hardcopy. It is presented in section 3.5.

3.2 Anaglyph Enhancement

Decreasing of disparity range of a stereo pair includes estimation of average disparity value on a stereo pair by the following method:

$$arg_{d_i} min \sum_{(x,y)\in W} |I(x,y) - I(x+d_i, y)|,$$

where d_i - disparity with minimal value of SAD (Sum of Absolute Differences) [7] in some region on left and right images. We compute average disparity $\bar{d}$ after estimation of the disparity map. Decreasing of disparity is produced by horizontal shifting of stereo pair proportionally to $\bar{d}$. If average disparity is less than 4 pixels, stereo pair is not changed.

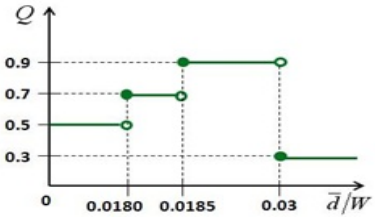
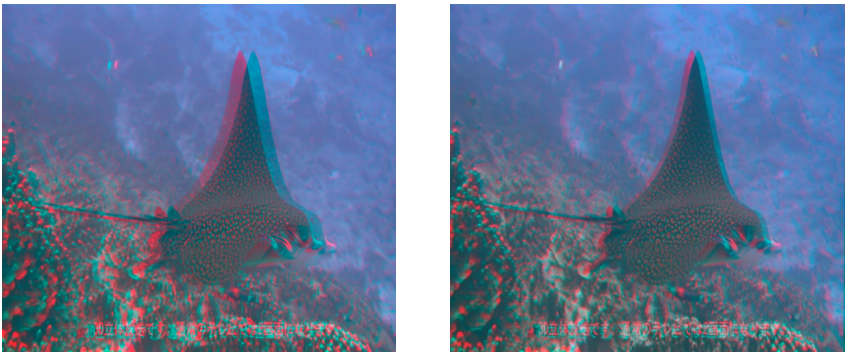


Fig. 3. Dependence of the regularization coefficient from size and disparity of a stereo pair

The main idea is that disparity should not be too large relatively to image size. Therefore, horizontal shift is computed as $\text{Shift} = \text{Q} \cdot \bar{\text{d}}$, where Q is regularization coefficient which depends on ratio of average disparity and image width W in pixels, as presented on figure 3. If the value of the ratio $\bar{\text{d}}$/W is greater than 0.03 it might

inform about erroneous average disparity estimation; and shifting of a stereo pair should be made with a smaller value. All constants of the empirical approach were found during a plenty of visual experiments.

Figure 4a shows the anaglyph without disparity correction. On its hardcopy ghosting effect is present. Figure 4b demonstrates the anaglyph with decreased disparity. For its generation stereo pair was shifted by several pixels. Ghosting effect for this anaglyph hardcopy is almost invisible.



a) Without disparity correction. b) With corrected disparity.

Fig. 4. Effect of disparity correction for anaglyphs

Simple and effective way for decreasing of crosstalk noise is defocusing one color channel for both images of stereo-pair [4]. We carry out red or blue channel or both the channels smoothing by means of low-pass box-filter:

$$\tilde{I}(x,y) = \frac{1}{(2N_x+1)(2N_y+1)} \sum_{n_x=-N_x}^{N_x} \sum_{n_y=-N_y}^{N_y} I(x-n_x, y-n_y),$$

where $2N_x + 1, 2N_y + 1$ are kernel size. Effective algorithm for Box filter calculation is based on summed area table (also well-known as *integral image*) [1]. The value at any point (x, y) in the integral image is just the sum of all the pixels above and to the left of (x, y), inclusive: $I(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x',y')$. Integral image I can be computed efficiently in a single pass over the image i, using the fact that the value in the summed area table at (x, y) is: $I(x,y) = i(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1)$ Sum of the darkest box shown in figure 5 can be calculated from integral image as:

$$\sum_{\substack{A(x)<x' \le C(x) \\ A(y)<y' \le C(y)}} i(x',y') = I(A) + I(C) - I(B) - I(D)$$

Such way provides identical processing time for any box size. There is no necessity to precalculate and store whole integral image. Integral image can be calculated for relatively small band jointly with Box filter.

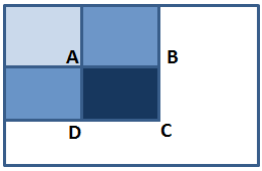


Fig. 5. Illustration of Box-filter calculation via integral image

3.3 Transmission Functions Estimation by User

For achievement of good correspondence between glasses and printed colors, we propose to estimate transmission functions $f_l(\lambda)$ and $f_r(\lambda)$ for given glasses and printer colors. It allows to reduce ghosting effect significantly and to improve quality of printed 3D color anaglyph image. In this case we need to know dependence of digital color components and reflection spectrum of these colors that is dependence of three color components RGB and wave length λ.

For evaluation of $f_l(\lambda)$ and $f_r(\lambda)$ we print the color pattern (figure 6) on target printer. This pattern is a color table including all printable colors in HSL space. This pattern reveals a dependence of digital color components and reflection spectrum of these colors. Hue corresponds to wavelength, lightness is an average between maximum and minimum values of spectrum; saturation is distance from maximum (or minimum) value to lightness. Rows of the table are colors with various saturation (step is 0.2), columns are colors with various hue (step is 10º).

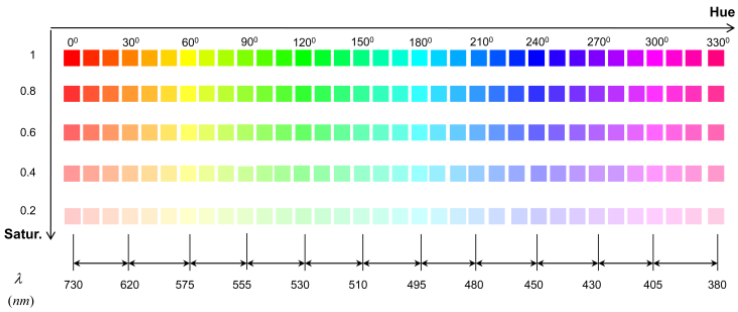


Fig. 6. Test color pattern

For estimation of transmission function of color filters of glasses user should examine the pattern through left and right filters of anaglyph glasses separately. By visibility level of color sample through the filter, the transmission function is evaluated and afterwards is applied for anaglyph generation. If the color sample is invisible, it fully passes through the filter and maximum of transmission function is at that location.

Let's left filter is red and right filter is cyan. Firstly user should examine first row with maximal saturation through red filter. We suppose that a color with hue = 10^0 ± 10^0 corresponds to red color of 700 nm ± 27.50 nm wave length range. If any color

from the band $10^0 \pm 10^0$ is invisible, the maximum of transmission function is in the 700 nm ± 27.50 nm range. If the color is visible, user should choose the row with less saturation (one of the lower rows) and look at corresponding color sample once more. If the color yet is visible, user should choose the row with the least saturation for analysis. Value of maximum of transmission function depends on saturation that can be extracted from the Table 1.

Table 1. Dependence of maximum of transmission value from saturation

Row of color pattern	Saturation	Maximum of transmission (*MaxTrans*)
1	1	0.90 (90%)
2	0.8	0.85 (85%)
3	0.6	0.65 (65%)
4	0.4	0.50 (50%)
5	0.2	0.45 (45%)

For cyan filter construction of the transmission function is similar. We use cyan filter of glasses and suppose cyan color has hue = $180^0 \pm 10^0$ that corresponds to 495 nm ± 3.75 nm range. User should examine the test pattern and define visibility level of each color by grade (0…5) which corresponds to transmission. For each clearly visible point we give a 0 grade. Values of transmission function depend on maximum of transmission associated with row of color pattern and grade evaluated by observer (table 2).

Table 2. Dependence of transmission value from grade

Transmission	Grade	Transmission	Grade
$\boldsymbol{MaxTrans \times 1}$	5	$MaxTrans \times 0.45$	2
$\boldsymbol{MaxTrans \times 0.85}$	4	$MaxTrans \times 0.20$	1
$\boldsymbol{MaxTrans \times 0.60}$	3	$MaxTrans \times 0$	0

User should mark only invisible or semitransparent colors. In this way we obtain rough approximation of transmission functions for particular glasses. Examples of such functions are shown on figure 7.

This approach for transmission estimation can be used for another type of glasses (green, orange and etc.). We suppose orange color as hue = $40^0 \pm 10^0$ that corresponds 609 nm ± 11.25 nm, green color as hue = $120^0 \pm 10^0$ that corresponds 530 nm ± 5.0 nm.

3.4 Algorithm for selection of transmission function of existing filter

Usage of the transmission functions in such “pure” form lead to anaglyph image with wrong colors. We propose to use known transmission functions of Roscolux filters and calculate functions of given glasses by selecting the nearest function to our rough approximation. Roscolux filters are used in professional photography. We’ve chosen

10 filters which have spectrum similar to spectrum of a commercial anaglyph glasses and transmission values of glasses computed by spectrometer.

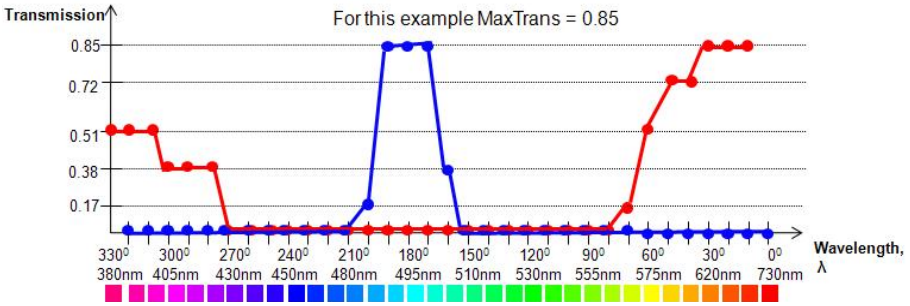


Fig. 7. Example of transmission function evaluated by user

In order to choose appropriate filter or glasses with transmission function F_i most similar to our computed transmission function, we note position of maximum of function and its value. Then, we choose F_i as estimation position of the maximum:

$$\left\|\lambda_{\max(F_i)} - \lambda_{\max(f)}\right\|_i \to \min \tag{1}$$

Then for the maximums with equal wavelength we apply the following condition:

$$\|\max(F_i) - \max(f)\|_i \to min. \tag{2}$$

For example, compare f (users' evaluation for red filter) on figure 8 and F_i (real transmission functions of Roscolux filters) on figure 9. On the top of the figure the transmission functions are presented; in the bottom the enlarged part of the graph with transmission function maximums area is shown. By analyzing wave lengths which correspond to maximums of transmission functions and according to decision rule (1) we select one of the most appropriate filters.

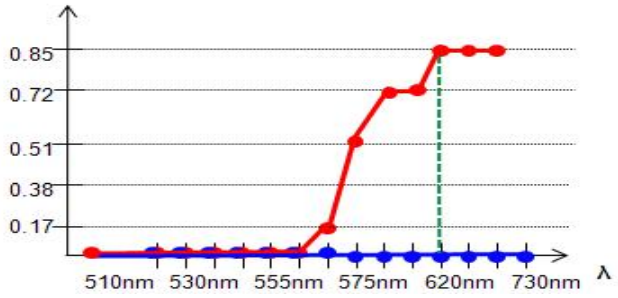


Fig. 8. A part of graph of f

$\lambda_{\max(f)} = 620$; $\lambda_{\max(F_1)} = 730$; $\lambda_{\max(F_2)} = 660$; $\lambda_{\max(F_3)} = 680$; $\lambda_{\max(F_4)} = 700$; $\lambda_{\max(F_5)} = 730$; $\lambda_{\max(F_6)} = 660$. According to (1) we choose $F_2 = 0.88$;

$F_6 = 0.89$; $\lambda_{\max(F_2)} = \lambda_{\max(F_6)} = 660nm$. Then for maximums with equal wavelength we apply the condition of the nearest transmission (2) and choose a Roscolux filter "Fire" with $F_2 = 0.88$.

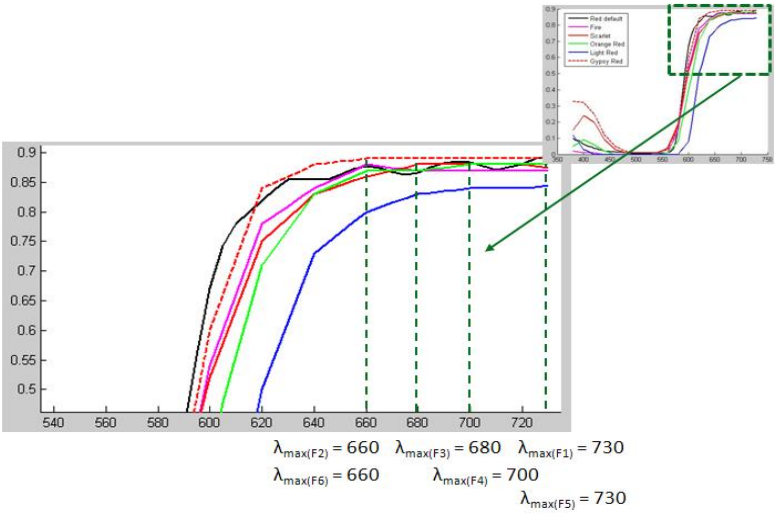


Fig. 9. Choosing transmission function of Roscolux filters

An anaglyph on figure 10a is generated with transmission functions of some common stereo glasses. Its hardcopy printed on Samsung CLP-6240 printer has perceptible ghosting effect for given glasses. An anaglyph on figure 10b is generated with transmission functions adapted for given glasses and printer colors. Ghosting effect on this anaglyph hardcopy is significantly reduced.

a) An anaglyph example for transmission functions of some common stereo glasses.

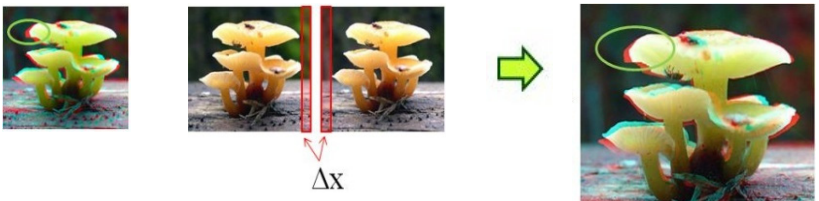
b) An anaglyph example with adaptation for given glasses.

Fig. 10. Anaglyphs examples

3.5 Adaptation to Size of Hardcopy

Anaglyph upsizing while printing leads to large disparity increase and emerging of ghosting effect. Downsizing leads to large disparity decrease and disappearance of stereo effect. We propose to decrease disparity value by shifting stereo pair images relative to each other before enlarging process.

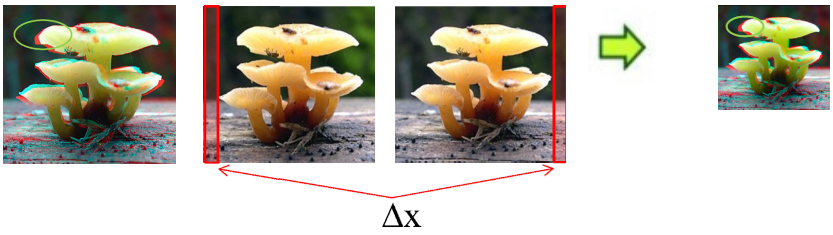
In case of upsizing $\Delta x = -0.3 * \bar{d}$, where $\bar{d}$ is average disparity value computed for whole images of stereo pair, Δx is value of shift of stereo pair images relative to each other for disparity decrease (figure 11). The value of constant 0.3 is chosen as a result of large number of experiments. *Vice versa* in case of downsizing $\Delta x = 0.3 *$ AvrgDisp is a shift for disparity increase (figure 12).



Shifting (Remove regions $\Delta x = -0.3 * \bar{d}$)

An anaglyph with smaller disparity and ghosting effect

Fig. 11. Disparity modification in case of upsizing



Shifting (Remove regions $\Delta x = 0.3 * AvrgDisp$)

An anaglyph with preserved stereo effect

Fig. 12. Disparity modification in case of downsizing

4 Results and Discussions

All mentioned algorithms were implemented in software application. User interface of the application is shown on figure 13. For creation of user interface we applied Windows Presentation Foundation API. Mathematical parts of algorithms were developed on C programming language in separate DLL. The software application allows:

- make calibration procedure:

  1. print color pattern;
  2. create new calibration scheme for given glasses;
  3. choose and edit existing calibration scheme;

- make alignment of stereo pair in manual and automatic mode;
- resize and placement an anaglyph on the printed page;
- save and print an anaglyph.

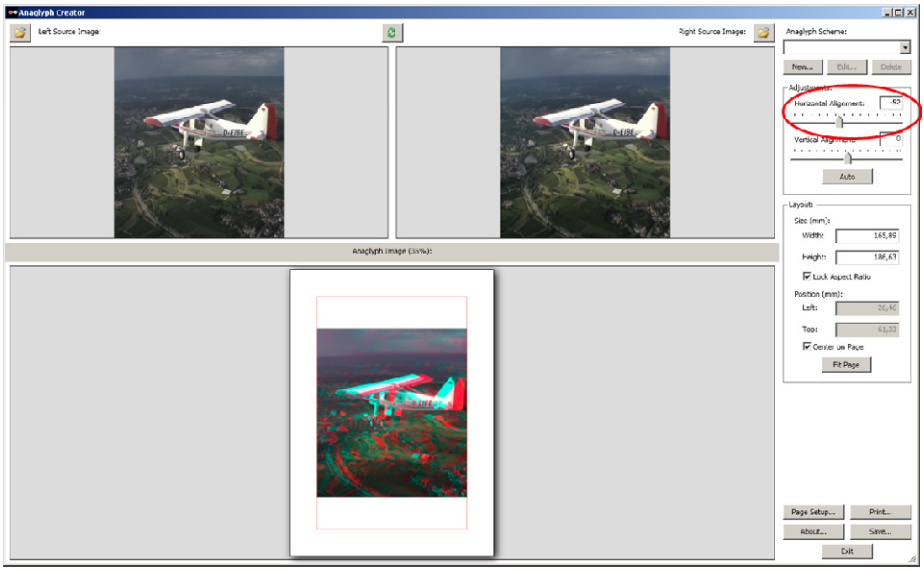


Fig. 13. Main window of the application for anaglyph generation and printing

Proposed technique utilizes low amount of memory and has relatively low computational complexity. Figure 15 demonstrates plot of processing time for anaglyph generation procedure depending on size of stereo-pair images. The computational time data were obtained on PC with dual-core 3 GHz CPU. It is already quite acceptable. Some additional platform-dependent optimizations are possible.

In order to perform benchmarking we used 5 mentioned above solutions for anaglyph generation and our software application. Evaluation was done by 14 observers with the same viewing conditions and glasses. We screened the subjects for normal color vision by Ishihara test plates. In our survey we used test set of 6 stereo pairs. Generated anaglyphs were printed on Samsung CLP-6240 printer. We propose to calculate quality factor of anaglyph generation as weighted sum of two subjective estimations:

$$E = \alpha_1 V_1 + \alpha_2 V_2,$$

where V1 is subjective quality of 3D visualization, V2 is subjective level of color naturalness, $\alpha_1 = 0.8$, $\alpha_2 = 0.2$. V_1 and V_2 are changed from 0 to 1 with step of 0.2; lower V_1 and V_2 are better. We prioritized weights α_1 and α_2 by using Analytic Hierarchy Process [8]. Table 3 contains comparison of the solutions for anaglyph generation obtained during our survey.

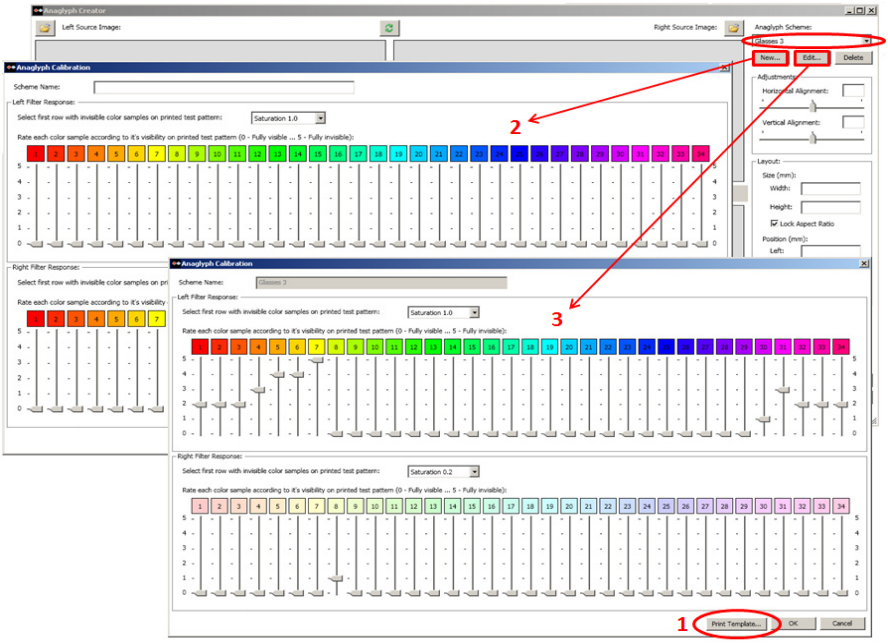


Fig. 14. Dialog for transmission function estimation

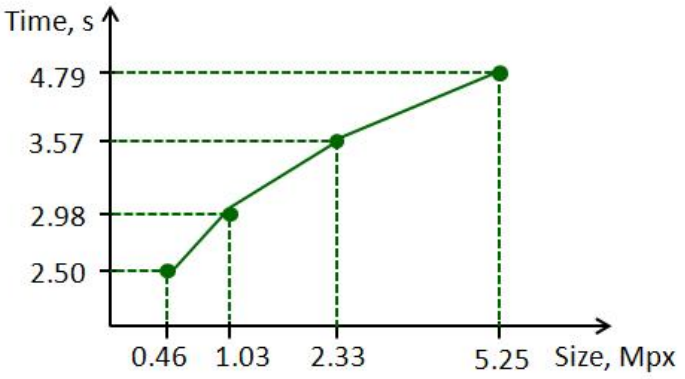


Fig. 15. Processing time for anaglyph generation procedure

Table 3. Comparison of applications for anaglyph generation

	Average E	Maximum E
StereoPhoto Maker	0.33	0.52
Anaglyph Maker	0.57	0.72
Anaglyph	0.45	0.56
Anaglyph Workshop	0.58	0.76
Z-Anaglyph	0.55	0.74
Proposed technique	0.30	0.58

In general our algorithm outperforms all tested solutions. The anaglyph generated by StereoPhoto Maker is presented on figure 16a. The anaglyph on hardcopy has well noticeable cross-talk noise. The anaglyph generated by proposed method is presented on figure 16b. The picture looks better due to enhancements.

a) The anaglyph generated by StereoPhoto Maker

b) The anaglyph generated by proposed method.

Fig. 16. Generated anaglyph examples

Acknowledgment. Authors would like to thank Konstantin Kryzhanovsky for help with algorithm implementation and discussions about the paper.

References

1. Franklin, C.: Summed-area Tables for Texture Mapping. In: 11th Annual Conference on Computer Graphics and Interactive Techniques SIGGRAPH 1984, New York, pp. 207–212 (1984)
2. Dubois, E.: A Projection Method to Generate Anaglyph Stereo Images. In: IEEE International Conference on Acoustic, Speech, and Signal Processing, vol. 3, pp. 1661–1664. IEEE Press, Salt Lake City (2001)
3. Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F.: Computer Graphics. Principles and Practice, 2nd edn. (1990)

4. Ideses, I., Yaroslavsky, L.: Three Methods That Improve the Visual Quality of Colour Anaglyphs. J. Opt. A: Pure Appl. Opt. 7, 755–762 (2005)
5. Lo, W.-Y., van Baar, J., Knaus, C., Zwicker, M., Gross, M.: Stereoscopic 3D Copy & Paste. ACM Transactions on Graphics (TOG) - Proceedings of ACM SIGGRAPH Asia 2010 29(6) (2010)
6. McAllister, D.F., Zhoub, Y., Sullivan, S.: Methods for Computing Color Anaglyphs. In: Stereoscopic Displays and Application XXI, SPIE Electronic Imaging, San Jose, vol. 7524 (2010)
7. Rziza, M., Aboutajdine, D.: Dense Disparity Map Estimation Using CUMULANTS. In: 18th IEEE International Conference on Image Processing, p. 984. IEEE Press, Brussels (2011)
8. Saaty, T.L.: Decision Making for Leaders: The Analytic Hierarchy Process for Decisions in a Complex World, New Edition. Analytic Hierarchy Process Series, vol. 2 (2001)
9. Sanders, W., McAllister, D.F.: Producing Anaglyphs from Synthetic Images. In: Electronic Imaging Conference, San Francisco, pp. 348–358 (2003)
10. Yun, Z., Pingping, X., Hui, L.: Data Fusion for Multi-scale Colour 3D Satellite Image Generation and Global 3D Visualization. In: ISPRS Commission VII Midterm Symposium on Remote Sensing: From Pixels to Processes, Enschede (2006)
11. Zeng, R., Zeng, H.: Printing Anaglyph Maps Optimized for Display. In: XVI Conference on Color Imaging: Displaying, Processing, Hardcopy, and Applications - Proceedings of SPIE, 0277-786X, v.7874, San Francisco, vol. 7866-63 (2011)

Audio-Adaptive Animation from Still Image

Konstantin Kryzhanovsky, Aleksey Vil'kin, Ilia Safonov, and Zoya Pushchina

Samsung Moscow Research Center, Dvintsev str., 12 bldg. 1, Moscow, Russia
{k.kryzhanovs,a.vilkin,ilia.safonov,p.zoya}@samsung.com

Abstract. In this paper we propose new approach of automatic generating **real time content adaptive** animation effects from the still images adapted for the low-powerful embedded HW platforms. Displayed animation behaves uniquely each time it's played back, and does not repeat itself during playback duration, creating vivid and lively impression for the viewer. Adaptation of the effect parameters according to background audio greatly increases aesthetic impression of the viewer. Three animation effects such as *Flashing Light*, *Soap Bubbles* and *Sunlight Spot* are described in details. We propose several ways of controlling the effect parameters by music. User opinion survey demonstrates that majority of users are excited by such effects and wants to see them in their devices with multimedia capability.

Keywords: animation from photo, audio-adaptive effect, multimedia slideshow, attention zones detection.

1 Introduction

Creation and sharing of multimedia presentations and slideshows has become a pervasive activity. The development of tools for automated creation of exciting, entertaining and eye-catching photo transitions and animation effects, accompanied by background music and/or voice comments, has become a modern trend [1]. One of the most impressive effects is the animation of still photo, for example, grass swaying in the wind or rain drop ripples in the water, etc.

Special interactive authoring tools, such as Adobe After Effects and Ulead Video Studio, are used to create animation from an image. Development of fast and realistic animation effects is hard task itself; and it is a topical problem of modern computer graphics. For example, paper [2] discusses algorithm for generation of plausible motions animation. In authoring tools the effects are selected and adjusted manually, which may require considerable effort from a user. Resulting animation is saved as video clip, thus requiring noticeable amount of space for storage. During playback, such movie will always be the same, thus leading to repetitiveness feeling of the viewer.

For the multimedia presentations and slideshows it's preferable to generate animated effects on-the-fly with a high frame rate. Very fast and efficient algorithms are necessary to provide required performance. It's especially difficult for low-powerful embedded HW platforms.

Our research was defined as development and implementation of automatically generated animated effects of Full HD images on ARM Cortex A8 and A9 – based

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XIX, LNCS 7870, pp. 48–62, 2013.

embedded platforms, with the CPU frequency 800 - 1000 MHz and without use of GPU – based APIs, such as OpenGL. Only ARM commands were available to use, including SIMD instructions of ARM NEON co-processor. Creation of realistic and complex animated effects in such limited conditions is a challenging task itself, particularly for the well experienced in computer games on powerful PCs and play stations users.

We have developed several algorithms for generation of content-based animation effects from still images, such as Flashing Light, Soap Bubbles, Sunlight Spot, Magnifier Effect, Rainbow, Portrait Morphing Transition Effect, Snow, Rain, Fog, etc. For those effects we propose a new approach of automatic audio-aware animation generation.

In the paper we demonstrate our concept, i.e. adaptation of effect parameters according to background audio, for three effects: Flashing Light, Soap Bubbles and Sunlight Spot. Obviously the concept can be extended to other animated effects.

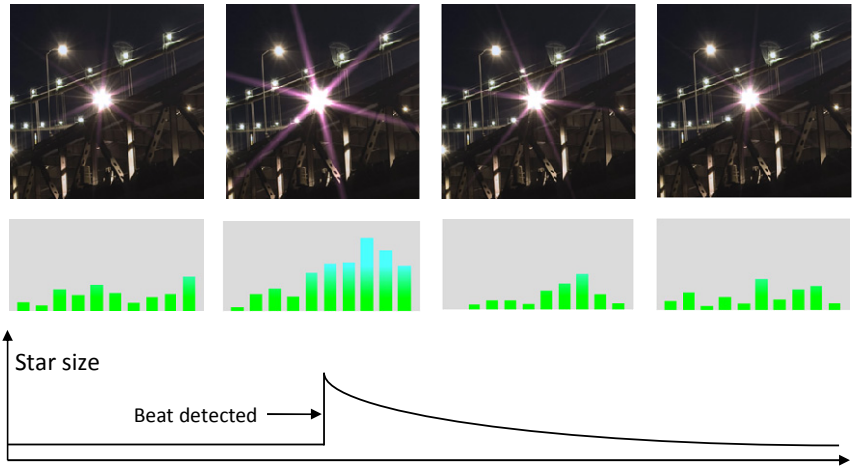


Fig. 1. Detected beats affect size of flashing light

2 Related Works

Recently, some content-adaptive automatic techniques for generation of animation from static photo were proposed. Paper [3] describes Animated Thumbnail which is a short looped movie demonstrating main objects of the scene in sequence. Animation simulates camera tracking-in, tracking-out and panning between detected visual attention zones and whole scene.

Music plays an important role in multimedia presentations. There are some methods towards to aesthetical audiovisual composition in slideshow. Tiling Slideshow [4] describes two methods for analysis of background audio in order to select timing for photos and frames switching. First one is beats detection. Second one is energy dynamics, calculated using root mean square values of adjacent audio frames.

Also there are other concepts of combining audio and visual information with the automatic generation of multimedia presentations exists. For example, paper [5] suggests approach that focuses on an automatic sound track selection. The process attempts to comprehend what the photos depict and try to choose music accordingly.

3 Animation Effects from Single Image

3.1 General Workflow

In general, the procedure of creation of animation effect from single still image consists of the following major stages: effect initialization and effect execution.

During effect initialization certain calculations which has to be made only once for entire effect span, are performed. Such operations may include source image format conversion, pre-processing, analysis, segmentation, creation of some visual objects and elements displayed during effect duration, etc. On execution stage for each subsequent frame the background audio analysis is performed, visual objects and their parameters are modified depending on time elapsed and calculated audio features, and entire modified scene is visualized. The generalized animation effect processing flow chart is displayed on fig. 2.

3.2 Flashing Light

The Flashing Light effect displays several flashing and rotating colored light stars over the bright spots on the image. In this effect, size, position and color of flashing light stars are defined by detected position, size and color of the bright areas on the source still image.

Algorithm performs the following steps to detect small bright areas on the image:

- calculating the histogram of luma channel of the source image
- calculating segmentation threshold as luma level corresponding to specified fraction of brightest pixels of the image using the luma histogram;
- segmenting source image by thresholding; while thresholding, the majority morphological filter is used to filter out localized bright pixel groups;
- calculation of the following features for each connected region of interest (ROI):

  a. Mean color C_{mean}
  b. Centroid (x_c, y_c)
  c. Image fraction F – fraction of the image area, occupied by ROI;
  d. Roundness – relation of the diameter of the circle with same area as ROI to maximum dimension of the ROI:

$$K_r = \frac{2\sqrt{S/\pi}}{\max(W,H)},$$

  where S is the area of the ROI and W, H are ROI bounding box dimensions;

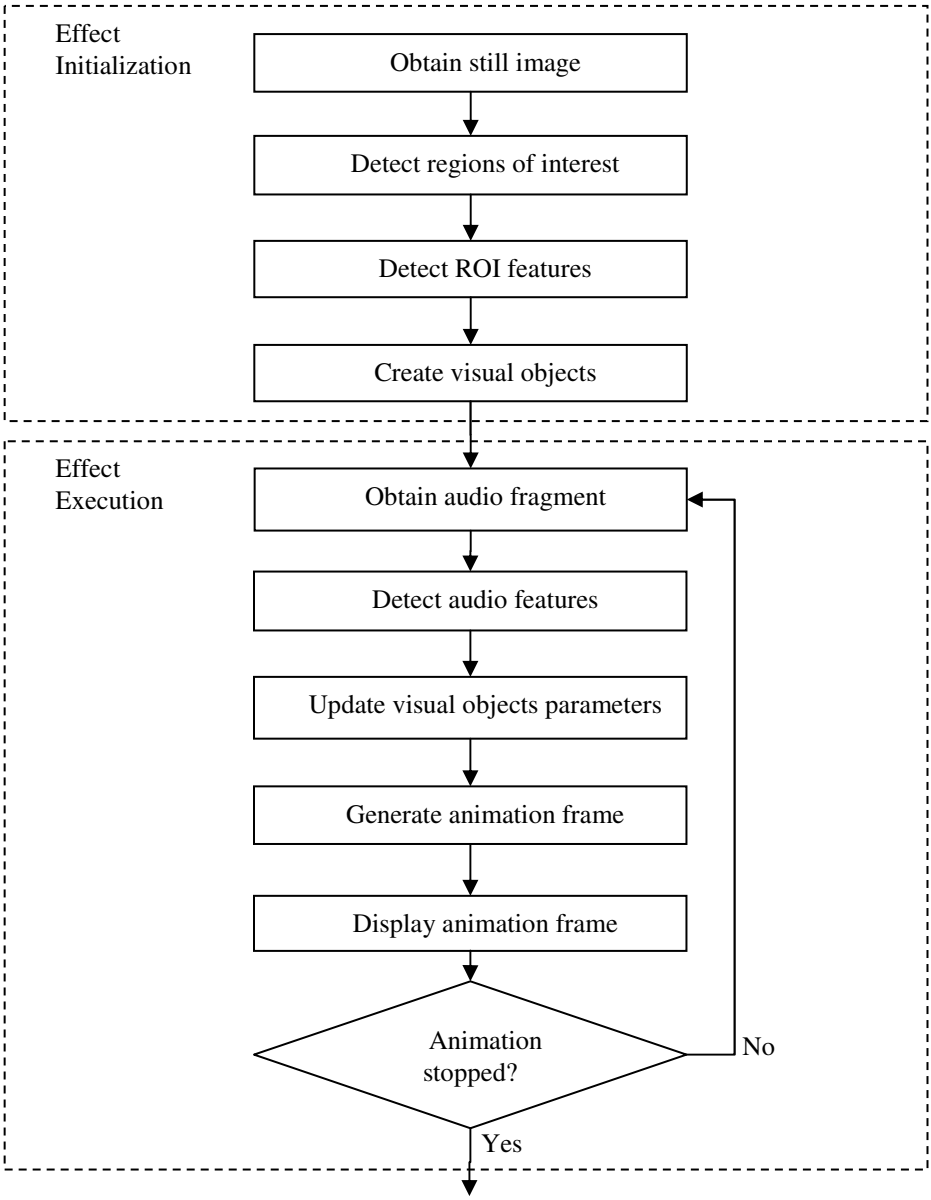


Fig. 2. Animation effect processing flow chart

e. Quality – integral parameter, characterizing the possibility of to ROI to be a light source and calculated as following:

$$Q_L = w_{Y\max} \cdot Y_{\max} + w_{Ymean} \cdot Y_{mean} + w_R \cdot K_r + w_F \cdot K_F ;$$

where $Y_{\max}$ - maximum luma of the ROI,

Y_{mean} - mean luma of the ROI,

K_F - coefficient of ROI size,

$K_F = \begin{cases} F / F_0, \ if \ F \le F_0 \\ F_0 / F, if \ F > F_0 \end{cases}$, where F_0 - image fraction normalizing coefficient for an optimal lightspot size;

$w_{Y\max}$, w_{Ymean}, w_R, w_F - weighting coefficients.

Weighting coefficients w and optimal lightspot size normalization coefficient F_0 are obtained by minimizing differences between automatic and manual light sources segmentation results.

- selection of regions with appropriate features.

All bright spots, with image fraction falling within appropriate range (F_{min}, F_{max}), and roundness Kr is larger than certain threshold value K_r^0 are considered as potential light sources. Potential light sources are sorted by their quality value Q_L. Specified number of light sources with the largest quality is selected as final positions of "light stars" objects.

Centroids of selected light regions are used as star positions. Star size is determined by dimensions of appropriate light region. Mean color of the region determines the color of the light star. Fig. 3. shows an image with bright spots and corresponding light stars.

a) b)

Fig. 3. a) Bright spots on the image and b) corresponding light stars

Every light star is composed from bitmap templates of two types, representing star shape elements: halo shape and star ray (or spike) shape. These templates are alpha maps scaled independently. Examples of templates are shown on fig. 4. During rendering, the alpha map of complete star of appropriate size is prepared in separate buffer, and then the star is painted with appropriate color with transparency value extracted from star alpha map.

During animation, light star sizes and intensities are changed gradually and randomly to make an expression of flashing lights.

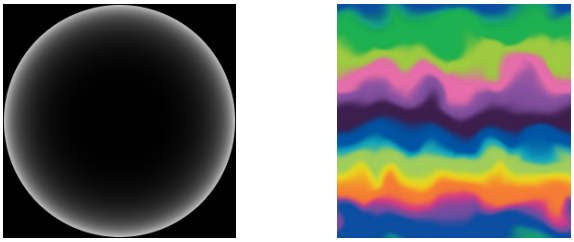
a) b)

Fig. 4. Light star shape templates: a) Halo template; b) Ray template

3.3 Soap Bubbles

The effect displays soap bubbles moving over the image. Each bubble is composed from color map, alpha map and highlight map. The set of highlight maps with the different highlight orientation is precalculated for each bubble. Highlight position depends on lighting direction in corresponding area of the image. Lighting gradient is calculated using downscaled brightness channel of the image.

Fig. 5 shows Soap Bubble components. Color map is modulated with highlight map, selected according average lighting direction around the bubble, and then combined with source image using alpha blending with bubble alpha map.



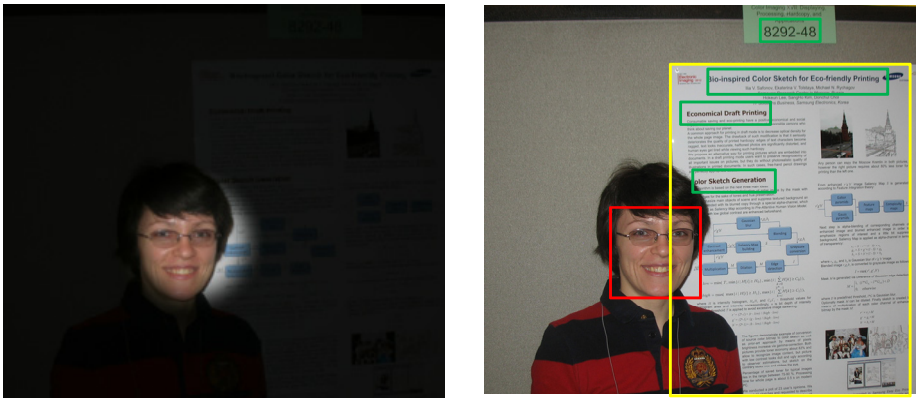
a) b)

Fig. 5. Soap Bubble components: a) Alpha map b) Color map

During animation, soap bubbles are moved smoothly over the image from bottom to top or vice versa while oscillating slightly in horizontal direction to make an impression of real soap bubbles floating in the air.

3.4 Sunlight Spot

The effect displays bright spot moving over the image. Prior starting effect, the image is dimmed according to its initial average brightness. Fig. 6 shows an image with sunlight spot effect. The spotlight trajectory and size are defined by attention zones on the photo.



a) b)

Fig. 6. a) Frame of *Sunlight Spot* effect; b) selected attention zones

Similar to many existing publication we find human faces and salient regions using pre-attentive vision model. Basing on these regions we form attention zones. In addition we consider text inscriptions as attention zones too. For example it can be the name of hotel or town on the background of which the photo was made. Or, in case of the newspaper, it will be headlines.

Well-known OpenCV software library contains implementation of face detection for front and profile faces. In general the technique that is based on state-of-the-art Viola-Jones face detector [6] provides good results. However it gets a lot of false positives. The number of false positives can be decreased with additional skin tone segmentation and processing of downsampled image [7]. We have ported OpenCV 2.3 to our embedded platform. Time of face detection for 480x320 images is about 0.8s.

So far the universal model of human vision does not exist, but pre-attentive vision model based on feature integration theory is well-known. Since in this case, the observer is on attentive stage while viewing photo, a model of human pre-attentive vision is not strictly required. However existing approaches for the detection of regions of interest are based on saliency map and they often provide reasonable outcomes, whereas the use of attentive vision model requires too much prior information about the scene and it is not generally applicable. Classical saliency map building algorithms like [9] have a very high computational complexity. That is why researchers recently devote a lot of efforts to develop fast saliency map creation techniques. Paper [10] compares several modern algorithms for salient regions detection.

We implemented on our embedded platform Histogram- based Contrast (HC) method. The time of salient regions detection for 480x320 images is about 0.1 s.

While developing the algorithm for detection of areas with text, we take into account the fact that text components are ordered the same way and are similar in texture features, color. Firstly, we apply LoG edge detector and restore missing parts using combination of morphological operations. After edge detection we end up with many circuits, which can be ordered as connected tree of objects and voids

Then we filter resulting connected components based on the analysis of the texture features. We use the following features [8], selected with the help of AdaBoost toolbox [15]:

- average block brightness B_i: $\overline{B}_i = \frac{\sum_{r=1}^{N}\sum_{c=1}^{N} B_i(r,c)}{N^2}$

- average difference of average brightnesses of the blocks B_k in 4-connected neighborhood of the block B_i: $\overline{dB}_i = \frac{\sum_{k=1}^{4} | \overline{B}_i - \overline{B}_k |}{4}$

- average of vertical dB_y^i and horizontal dB_x^i block derivative:

$$\overline{d_{x,y}B}_i = \frac{\sum_{r=1}^{N}\sum_{c=1}^{N-1} dB_x^i(r,c) + \sum_{r=1}^{N-1}\sum_{c=1}^{N} dB_y^i(r,c)}{2N(N-1)}$$

- block homogeneity B_i: $H = \sum_{i,j} \frac{N_d(i,j)}{1+|i-j|}$,

where N_d is a normalized co-occurrence matrix, d defines the spatial relationship.

- the percentage of pixels with the gradient greater than the threshold:

$$P_g = \sum_{\forall(r,c)\in Bi} \{1 | \nabla B_i(r,c) > T\} / N^2,$$

where $\nabla B_i(r,c)$ is calculated as square root of the sum of the squares of horizontal and vertical derivatives.

- the percentage of pixel value changes after morphological operation of opening B_i^o on a binary image B_i^b, obtained by binarization with a threshold of 128:

$$P_m = \sum_{\forall(r,c)\in Bi} \{1 | B_i^o(r,c) \neq B_i^b(r,c)\} / N^2 .$$

Also, an analysis of geometric dimensions and relations is performed. We merge closely located connected components, arranged the same order and similar in color, texture features, in groups. Then we classify resulting groups. We form final zones with the text on the basis of groups that are classified as text. Time of text regions detection for 480x320 images is about 0.5 s.

Fig. 5 shows detected attention zones. Red rectangle depicts face detected; green rectangles denote text regions; yellow is bounding box of the most salient area according to HC method.

4 Adaptation to Audio

What animation parameters may depend on characteristics of background audio signal? Firstly it is size and intensity of animated objects, also speed of their movement and rotation can be adjusted. In addition, we investigated the question: How can we change color of animate objects, depending on music? Attempts to establish a connection between music and color are known for long time. French mathematician Louis Bertrand Castel is considered as a pioneer in this area. In 1724 in his work *Traité de physique sur la pesanteur universelle des corps* he described an approach to direct "translation" of music to color on "specter – octave" basis. To illustrate his ideas, Castel even constructed *Clavecin pour les yeux* (*Ocular Harpsichord*, 1725). Famous Russian composer and pianist Alexander Scriabin about 100 years ago also proposed a theory of connection between music and color. Colors corresponding to notes are shown on fig. 7. This theory connects major and minor tonality of the same name.

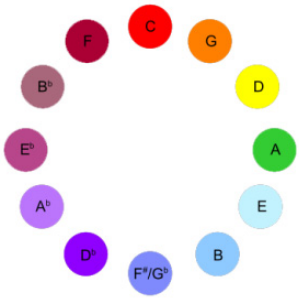


Fig. 7. Accords with the circle of fifths corresponding to Scriabin's theory

Scriabin's theory was embodied in *clavier à lumières (keyboard with lights)*, which he invented for use in his work Prometheus: Poem of Fire. The instrument was supposed to be a keyboard (fig. 8), with notes corresponding to colors as given by Scriabin's synesthetic system.

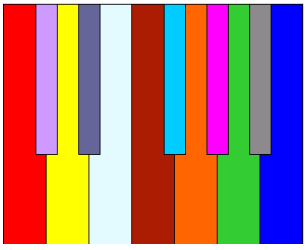


Fig. 8. Tone-to-color mapping on Scriabin's *clavier à lumières*

On our platform we work with stereo audio signal on frequency 44 kHz. We consider 4 approaches to connect animation of 3 effects mentioned above with background audio. In all approaches we analyze the average of two signal channels in

frequency domain. The spectrum is built 10 times per second for 4096 samples. Spectrum is divided into several bands as in conventional graphic equalizer. The number of bands depends on approach selected.

For fast Fourier transform computing with fixed point arithmetic we use kiss_fft library. It is open source library distributed under BSD license. This library does not use platform-specific commands and is easily ported to ARM. On our platform processing time for one buffer is about 0.004 s.

Our first approach of visualizing music by colors was inspired by Luke Nimitz demonstration of "Frequency spectrograph – Primary Harmonic Music Visualizer". It is similar to Scriabin idea. It can be considered as a specific visualization of the graphic equalizer. In this demonstration music octaves are associated with HSL color wheel as shown in fig. 9 using statement:

$$Angle = 2\pi \log_2(\frac{f}{c}),$$

where f is frequency, c is origin on frequency axis. Angle defines hue of current frequency.

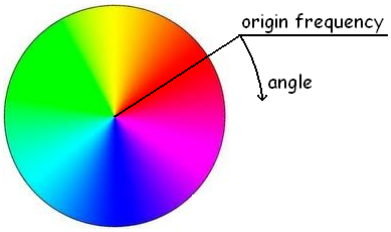


Fig. 9. Color circle corresponding to each octave

Depending on value of current note we define brightness of selected hue and draw it on color circle. We use three different approaches to display color on the color wheel: paint sectors, paint along radius or use different geometric primitives inscribed into the circle.

Fig. 10. Generated color distribution of soap bubbles depending on music

In Soap Bubbles effect, depending on generated color circle, we determine color of bubble texture. On fig. 10 there is an example of soap bubbles with color distribution depending on music. In Sunlight Spot effect generated color circle determines distribution of colors on highlighted spot (fig. 11).

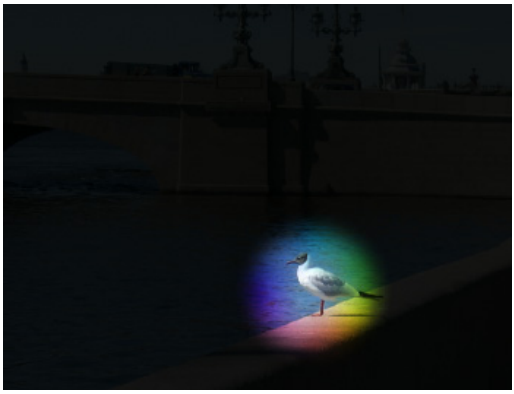


Fig. 11. Generated color distribution of sunlight spot depending on music

In second approach we detect beats or rhythm of the music. We tried several techniques for beats detection in time and frequency domains [11, 12, 13, 14]. We faced constraints due to real-time performance limitation and we were dissatisfied with the outcomes for some music genres. Finally we assume that the beat is present if there are significant changes of values in several bands. This method meets performance requirements with acceptable quality of beats finding. Fig. 1 illustrates how detected beats affect size of flashing light. If the beat is detected we instantly maximize size and brightness of lights and then they gradually return to their normal state until next beat happens. Also it is possible to change flashing lights when beat happens (turn on and off light sources). In the Soap Bubbles effect we maximize saturation of the soap bubble color when the beat takes place. We also change the direction of moving soap bubbles as beat happened. In Sunlight Spot effect if the beat is detected we maximize brightness and size of spot and then they gradually returned to their normal state.

In third approach we analyze presence of low, middle and high frequencies in audio signal. This principle is used in color music installations. In Soap Bubbles effect we assign frequency range for each soap bubble and define its saturation according value of corresponding frequency range. In Flashing Light effect we assign each light star to its own frequency range and define its size and brightness depending on value of the frequency range. On fig. 12 you can see how presence of low, middle and high frequencies affect on flashing lights.

Another approach is not to divide spectrum to low, middle and high frequencies but rather to assign it to different tones inside octaves. So, we work with equalizer containing large amount of bands, where each octave have enough corresponding bands. We accumulate values of each equalizer band to buffer cell, where corresponding cell number is calculated using the following statement:

$$num = \frac{(\log_2(\frac{f}{c}) \times 360) \mod 360}{\frac{360}{length}} + 1,$$

where f is frequency, c is origin on frequency axis, length is number of cells.

Fig. 12. Low, middle and high frequencies affect on brightness and saturation of corresponding flashing lights

Each cell controls behavior of selected objects. In Soap Bubbles effect we assign each soap bubble to corresponding cell and define its saturation depending on the value of the cell. In Flashing Light effect we assign each light to corresponding cell and define its size and brightness depending on the value of the cell.

5 Results and Discussion

The major issue is how can we implement the functions in modern multimedia devices for real-time animation? The algorithms were optimized for ARM Cortex A8 and A9 – based platforms with CPU frequency 800 - 1000 MHz. Limited computational resources of the target platform combined with absence of graphics hardware acceleration is serious challenge for implementation of visually rich animation effects. Therefore comprehensive optimization is required to obtain smooth framerates. Total performance win is 8.4 times in comparison to initial implementation. The most valuable optimization approaches are listed in table 1.

Table 2 contains performance data for described effects. Such figures provide smooth and visually pleasant animation.

As objective evaluation of the proposed audiovisual presentation is difficult, we evaluate the advantage of our technique through subjective user opinion survey. Flashing Light, Soap Bubbles and Sunlight Spot effects with octave based audio

adaptation were used for demonstration. Two questions were asked for three audio-visual effects: 1. Are you excited by the effect? 2. Would you like to see that effect in your multimedia device?

Table 1. Optimization approaches

Approach	*Speeding-up, times*
Fixed-point arithmetic	4.5
SIMD CPU instructions (NEON)	3
Effective cache usage	1.5
Re-implementing of key glibc functions	1.25

Table 2. Performance of proposed effects for HD photo

Effect	*Initialization time, s*	***FPS***
Flashing Light	0.15	20
Soap Bubbles	0.08	45
Sunlight Spot	1.4	50

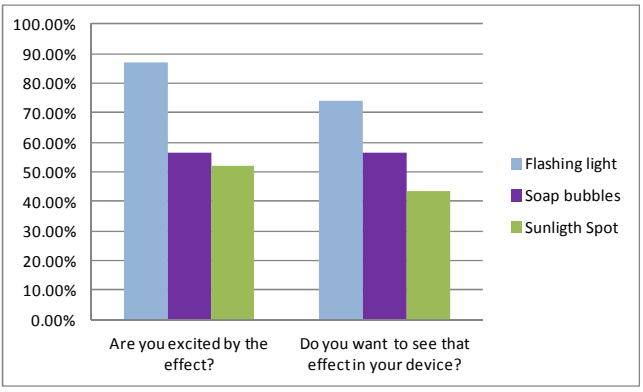


Fig. 13. Survey results

23 observers participated in the survey. Diagram on fig. 13 reflects survey results. In general, absolute majority of the interviewees rates effects positively. Only two people said that they do not like not only demonstrated effects, but any multimedia effects. Some observers stated: it's entertaining, but I cannot say "I'm excited", because such expression would be too strong. Several participants of the survey said that they do not like photos or background music used for demonstration. It is also worth to notice that 8 of the respondents were women and, on average, they rated the effects much higher than men.

So we can claim that the outcomes of subjective evaluation demonstrate the satisfaction of the observers with this new type of audiovisual presentation, because audio-aware animation behaves uniquely each time it is played back, and does not repeat

itself during playback duration, thus creating vivid and lively impression for the observer. A lot of observers were excited by the effects; and they want to see such features in their devices with multimedia capabilities.

6 Future Works

Several other animation effects have been considered from audio-awareness point of view. Fig. 14 shows screenshots of several audio-aware effect prototypes.

Fig. 14. Audio-aware animation effect prototypes: a) Rainbow effect, b) Confetti effect, c) Magnifier effect, d) Lightning effect

In "Rainbow" effect (fig. 14a), color distribution of the rainbow changes according to background audio specter. Movement direction, speed and color distribution of confetti and serpentines are adjusted with music rhythm in "Confetti" effect (fig. 14b). Magnifier glass movement speed and magnification is affected by background music temp (fig. 14c). In "Lightning" effect (fig. 14d), the moments of lightning bolt strikes are matched to accents in background audio.

Obviously, other approaches to adapt behavior of animation to the background audio are also possible. In particular, it is possible to analyze the left and right audio

channels separately and apply the different behavior to the left and right sides of the screen, respectively. Other effects friendlier for music adaption can be created.

The upcoming generation of target platform is equipped with Mali 6xx GPU, which supports graphics acceleration through OpenGL ES 2.1 API and general purpose calculations acceleration using OpenCL 1.1 framework. So, the most computation intensive routines, which, by the nature of used processing methods, can be effectively parallelized and moved to GPU allowing to reach even better effects performance and detail level.

References

1. Chen, J., Xiao, J., Gao, Y.: iSlideshow: a Content-Aware Slideshow System. In: ACM Intelligent User Interface Conf., Hong Kong, pp. 293–296 (2010)
2. Sakaino, H.: The photodynamic tool: generation of animation from a single texture image. In: IEEE ICME, Amsterdam (2005)
3. Safonov, I., Bucha, V.: Animated thumbnail for still image. In: GRAPHICON 2010, St. Petersburg, pp. 79–86 (2010)
4. Chen, J.C., Chu, W.T., Kuo, J.H., Weng, C.Y., Wu, J.L.: Tiling slideshow. In: ACM Multimedia 2006, Santa Barbara, pp. 25–35 (2006)
5. Dunker, P., Popp, P., Cook, R.: Content-aware auto-soundtracks for personal photo music slideshows. In: IEEE ICME 2011, Barcelona, pp. 1–5 (2011)
6. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proc. of Conference Computer Vision and Pattern Recognition, Kauai, pp. 511–518 (2001)
7. Egorova, M.A., Murynin, A.B., Safonov, I.V.: An Improvement of face detection algorithm for color photos. Pattern Recognition and Image Analysis 19(4), 634–640 (2009)
8. Vil'kin, A.M., Safonov, I.V., Egorova, M.A.: Bottom-up Document Segmentation Method Based on Textural Features. Pattern Recognition and Image Analysis 21(3), 565–568 (2011)
9. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(11), 1254–1259 (1998)
10. Cheng, M.M., Zhang, G.X., Mitra, N.J., Huang, X., Hu, S.M.: Global Contrast based Salient Region Detection. In: IEEE CVPR 2011, Colorado, pp. 409–416 (Springs 2011)
11. Goto, M.: Real-time music-scene-description system: predominant-F0 estimation for detecting melody and bass lines in real-world audio signals. Speech Communication 43(4), 311–329 (2004)
12. Dixon, S.: Audio Beat Tracking Evaluation: BeatRoot. In: MIREX at 7th International ISMIR 2006 Conference, Victoria (2006)
13. Scheirer, E.D.: Tempo and beat analysis of acoustic musical signals. J. Acoust. Soc. Amer. 103(1), 588–601 (1998)
14. McKinney, M.F., Moelants, D., Davies, M.E.P., Klapuri, A.: Evaluation of Audio Beat Tracking and Music Tempo Extraction Algorithms. Journal of New Music Research 36(1), 1–16 (2007)
15. Vezhnevets, A., Vezhnevets, V.: Modest AdaBoost – teaching AdaBoost to generalize better. In: Proc. of Graphicon Conf., Moscow, pp. 322–325 (2005)

Auto-calibration for Image Mosaicing and Stereo Vision

Alexey Spizhevoy and Victor Eruhimov

Itseez Ltd. and Lobachevsky State University of Nizhni Novogord, Russia
{alexey.spizhevoy,victor.eruhimov}@itseez.com

Abstract. The paper investigates the auto-calibration problem for mobile device cameras. We extend existing algorithms to get a robust method that computes internal camera parameters given a series of distant objects images. The algorithm is tested on real images generated by several different cameras. We estimate the impact of errors in camera calibration parameters in image mosaicing and 3D reconstruction problems.

Keywords: auto-calibration, camera parameters, errors effect, real datasets, image mosaicing, stereo.

1 Introduction

The goal of calibration is to determine internal camera parameters within the given projection model. The problem arises in a number of emerging computer vision applications such as augmented reality, 3D reconstruction, and image mosaicing (or stitching). As academy and industry becomes gradually more interested in using mobile devices for computer vision, the importance of phone/tablet cameras calibration is clear.

Nowadays the problem of camera calibration is usually solved by using special calibration patterns (see [3], [4]). While pattern-based methods are quite accurate, it can be difficult to use them due to necessity of taking shots of a special calibration object like a chessboard. Also, manual calibration harms user experience that is considered crucial for mobile applications. As a result software developers and researchers are very interested in auto-calibration methods.

Auto-calibration is the process of estimating internal camera parameters directly from multiple uncalibrated images. This area of computer vision is in active research stage. From one hand there are papers describing successful attempts of using auto-calibration methods in practical tasks (e.g. augmented reality, 3D reconstruction, image mosaics, see [6], [7], [8], [10], [11]). As the topics of these papers aren't camera auto-calibration itself, they don't contain thorough investigations of the used methods with numerical evaluation, tested on challenging dataset. As a consequence, when one faces a computer vision problem that requires camera parameters, it's very difficult to select a robust auto-calibration method and reuse previous results. There is research that is directly devoted to

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XIX, LNCS 7870, pp. 63–77, 2013.

the auto-calibration problem (see [9], [12]). Unfortunately, these papers either don't compare with state-of-the-art pattern-based calibration methods or provide evaluation for synthetic datasets only. Some of these papers describe results for real datasets, but obtained under almost ideal conditions like no noise, no hand shaking, see [12]. So to the best of our knowledge we are not aware of a research paper that describes an auto-calibration method and provides sufficient experimental evidence showing robustness for practical applications.

While classical calibration methods are well studied, they suffer from some drawbacks, which follow from the fact that these methods use some extra information. For instance, there are calibration methods (see [1]) which require location of vanishing points (i.e. points where infinite lines are terminated under projective transformation) as input, but finding of these points automatically is a difficult problem.

This paper shows that under moderate assumptions an auto-calibration algorithm for rotational cameras presented in [1] can be used for practical applications with a necessary pre-processing step. We evaluate an implementation of the method for both simulated datasets and real image sequences generated by mobile phone cameras.

2 Rotational Camera Auto-calibration

2.1 Problem Statement

We use the following camera model which describes how a 3D scene point $(X, Y, Z)^T$ is projected into an image pixel with coordinates $(u, v)^T$:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \simeq K(R|T) \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{1}$$

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2}$$

where K is camera intrinsic parameters matrix (f_x, f_y are focal lengths in pixels, c_x, c_y are principal point coordinates); R, T are camera rotation 3×3 matrix and translation 3D vector; the sign $\simeq$ here denotes similarity up to scale.

The class of auto-calibration methods that we will consider requires an existence of homography mapping between all input images. The easiest way of generating a sequence of images with homography relationship using a mobile camera is to take shots of distance objects. Hence, within the scope of this paper we will make an assumption that camera translation T is negligibly small compared to the distance to the objects. We will call a device with $T = 0$ a "rotational camera".

We formulate the auto-calibration problem in the following way: given keypoints in input images taken by a rotational camera, and the keypoint correspondences between images, find the camera matrix K.

2.2 Intrinsic Parameters Error Effect

The estimation of K is never the final goal of a computer vision application. So, in order to understand how precise an auto-calibration method has to be, we need to consider a specific application. This section contains a theoretical and experimental analysis for the image mosaicing problem and provides experimental evaluation on the stitching module of OpenCV library [17]. Throughout this section we make an assumption that f_x equals to f_y for the sake of simplicity and without loss of generality, as images always can be scaled to achieve of unit pixel aspect ratio.

It is possible to stitch images without involving camera matrix. In that case a user wouldn't be able to select another surface for projection except for plane, that can be inappropriate for big panoramas because of big deformations. A plane projection surface generates deformations in the panorama image are visible when the vector of camera orientation differs a lot from the projection plane normal. The most convenient projection surface for the case of rotational cameras is a sphere.

Below we analyze warping errors when the projection surface is a sphere. To compute the error for each image we do the following:

1. For each pixel $q = (x, y, 1)^T$ of the source image we find a ray, passing through the corresponding scene point from camera center, as $r = K^{-1}q$, where K is the camera matrix.
2. We find the intersection point $(X, Y, Z)^T$ of the ray with the unit sphere centered at the origin. This point spherical coordinates u, v after scaling by constant s are point coordinates on the final panorama (s is usually selected being roughly close to the focal length in pixels):

$$u = s \cdot \tan^{-1}(\frac{X}{Z}) \tag{3}$$

$$v = s \cdot (\pi - \cos^{-1}(\frac{Y}{\sqrt{X^2 + Y^2 + Z^2}})) \tag{4}$$

3. To calculate per pixel error we project points using the ground truth camera matrix

$$K^{(gt)} = \begin{pmatrix} f^{(gt)} & 0 & c_x^{(gt)} \\ 0 & f^{(gt)} & c_y^{(gt)} \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

and its estimation

$$K^{(est)} = \begin{pmatrix} f^{(gt)} f^{(rel)} & 0 & c_x^{(gt)} c_x^{(rel)} \\ 0 & f^{(gt)} f^{(rel)} & c_y^{(gt)} c_y^{(rel)} \\ 0 & 0 & 1 \end{pmatrix} \tag{6}$$

where $f^{(rel)}, c_x^{(rel)}, c_y^{(rel)}$ are estimated camera parameters relative to the ground truth. The distance between two points obtained using $K^{(gt)}$ and $K^{(est)}$ is the warping error in the pixel p.

According to the presented algorithm we first get two ray directions:

$$r^{(gt)} = \begin{pmatrix} X^{(gt)} \\ Y^{(gt)} \\ Z^{(gt)} \end{pmatrix} = \left(K^{(gt)}\right)^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{7}$$

$$r^{(est)} = \begin{pmatrix} X^{(est)} \\ Y^{(est)} \\ Z^{(est)} \end{pmatrix} = \left(K^{(est)}\right)^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{8}$$

Then we use (3) and (4) to get pixels coordinates $(u^{(gt)}, v^{(gt)})^T$ and $(u^{(est)}, v^{(est)})^T$. The final pixel warp error equals to $\sqrt{(u_{gt} - u_{est})^2 + (v_{gt} - v_{est})^2}$. We assess warping errors for the case of 2048×1536 images and using the following camera matrix as a reference:

$$K^{(gt)} = \begin{pmatrix} W + H & 0 & \frac{W}{2} \\ 0 & W + H & \frac{H}{2} \\ 0 & 0 & 1 \end{pmatrix} \tag{9}$$

where W and H are image width and height respectively.

The warping error function charts for 5% relative errors in camera intrinsic parameters are shown in figures 1 and 2. We can see from charts, that when relative error in camera parameters is 5% warp error reaches 60 pixels, that seems to be high enough for leading to visible artifacts.

In order to evaluate the artifacts, we stitched 1536×2048 images using camera matrix $K^{(pt)}$ as ground truth $K^{(gt)}$, where $K^{(pt)}$ was the camera matrix obtained via a pattern based calibration method. Also we did experiments using camera matrix $K^{(est)}$, where each parameter was modified (one at a time) to get 10% error (relative to $K^{(pt)}$). We got panoramas without visible artifacts, see figure 3. Small artifacts are highlighted with red color, but the quality of the panoramas is much higher that we could expect from theoretical analysis.

Such results are obtained because current stitching applications (including the one used for testing) use seam estimation methods to minimize visible artifacts, see [13]. After estimating seams special blending methods are used to hide discrepancies between images, see [14]. So even if the image registration step introduces moderate errors, a combination of modern seam estimation and blending methods can remove a lot of possible artifacts. But if errors in camera

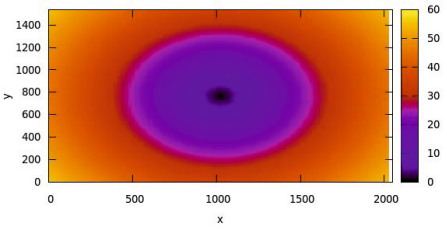


Fig. 1. Pixel warp error for $f^{(rel)} = 1.05$

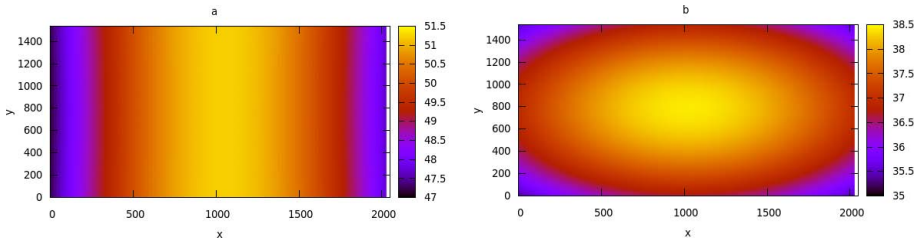


Fig. 2. Pixel warp error for a) $c_x^{(rel)} = 1.05$, b) $c_y^{(rel)} = 1.05$

parameters is too high then it's almost impossible to hide stretches and other artifacts, see figure 4 with results for $f^{(rel)} = 0.7$ (i.e. 30% relative error).

Also it should be mentioned that motions between images are estimated to minimize overall re-projection error (that is minimizing visible mis-registration error) according to the current camera matrix. This step is very important as minimizing re-projection errors leads to minimizing visible artifacts even if the camera matrix was estimated inaccurately.

From these results it follows that if one has a high quality stitching algorithm then the effect of errors in camera matrix isn't very high, and methods less accurate than pattern based calibration can be used for camera parameters estimation. This is a good application for auto-calibration that is not as precise as pattern-based calibration but still generates a reasonable estimation of camera intrinsic parameters.

2.3 Proposed Algorithm

A robust auto-calibration algorithm faces many challenges coming from data generated by a mobile device. Some input images can be noisy, can differ in illumination, and undesired objects such as user fingers can be present in the camera field of view. All these issues can affect the quality of extracted features, and can lead to mis-registration. Hence, let alone the core auto-calibration problem, we have to address these issues. This is why we start with a description of our registration algorithm.

The outputs of the registration algorithm is the images graph, where vertices are images from the input image sequence, and two images are connected with

Fig. 3. Panoramas for $f^{(rel)} = 1.1$, $c_x^{(rel)} = 1.1$, and $c_y^{(rel)} = 1.1$ respectively

Fig. 4. Panorama for $f^{(rel)} = 0.7$ with visible artifacts and stretches

the edge iff we were able to register them with a homography transformation. Here is the description of the registration pipeline:

1. Find keypoints and their descriptors of each image. We use SURF detector and descriptor implemented in OpenCV library, see [15].
2. For each image pair find matches between keypoints. We use FLANN matcher integrated into OpenCV library, see [16].
3. For each image pair estimate 2D homography and compute number of inlier matches, see 2.3.
4. For each image pair determine whether matches between these images are trustworthy, see section 2.3. The decision is made for image pair, not for each match. So if we're confident then we add an edge between two corresponding vertices into images graph.
5. Retain the biggest connected component from the images graph. Also retain only matches for confident image pairs and continue working with this connected component.

Computing Match Confidence. We follow the method proposed in [2], where it is applied to extract a subset of images from the original raw set for subsequent stitching.

Suppose we have n_f feature matches. The correctness of an image match is represented by the binary variable $m \in \{0,1\}$. The event that the i^{th} feature match $f^{(i)} \in \{0,1\}$ is an inlier/outlier is assumed to be independent Bernoulli event, so the total number of inliers n_i is Binomial. If $m = 1$ then n_i has the $B(n_i; n_f, p_1)$ distribution function, and $B(n_i; n_f, p_0)$ otherwise, where p_1 is the probability that a feature is an inlier given a correct image match, and p_0 is the probability a feature is an inlier given a false image match.

Here is the final criterion used by the authors to accept an image match

$$\frac{B(n_i; n_f, p_1)P(m=1)}{B(n_i; n_f, p_0)P(m=0)} \geq \frac{p_{min}}{1-p_{min}} \tag{10}$$

Choosing the values for $p_1 = 0.6$, $p_0 = 0.1$, $P(m=1) = 10^{-6}$ and $p_{min} = 0.999$ gives the condition

$$n_i > \alpha + \beta n_f \tag{11}$$

for a correct image match, where $\alpha = 8.0$ and $\beta = 0.3$. We decide whether a feature match is an inlier or an outlier by comparing reprojection error with a fixed threshold. We used the same value of 3 pixels for all datasets and that value worked good enough in practice, while for each particular dataset another threshold value can be better.

The value $\frac{n_i}{\alpha+\beta n_f}$ is used as the measure of confidence that it makes sense to use matches between an image pair. If it's greater than 1 then an image match is correct, false otherwise. In some practical cases it could be useful to increase this threshold as was found in experiments.

Figure 5 shows how reprojection error threshold affects on average camera parameters estimation relative error Q for one of real datasets.

$$Q = \frac{1}{4}(|f_x^{(rel)} - 1| + |f_y^{(rel)} - 1| + |c_x^{(rel)} - 1| + |c_y^{(rel)} - 1|) \tag{12}$$

Proposed Algorithm Details. For auto-calibration we use the algorithm for the rotation only cameras case proposed in [1]. Here is the brief description of that algorithm:

1. Normalize the homographies $H_{i,j}$ between views i and j such that $\det H_{i,j} = 1$.
2. Compute $\omega = (KK^T)^{-1}$ from the equations

$$\omega = H_{j,i}^T \omega H_{j,i}$$

   for all image pairs i, j.
3. Compute K solving $\omega = (KK^T)^{-1}$ with the Cholesky decomposition.
4. Refine K by minimizing the re-projection error function

$$err(K, R_1, ..., R_n) = \sum_{i,j,k} \|x_j^{(k)} - H_{i,j} x_i^{(k)}\|$$

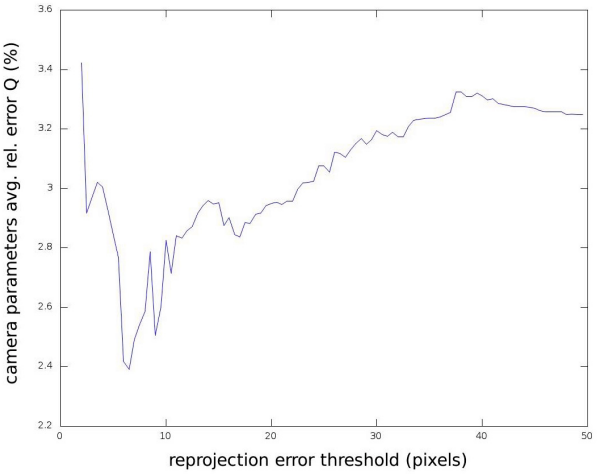


Fig. 5. Reprojection error threshold effect on camera parameters estimation errors. When the threshold is too low the algorithm is too sensitive to noise, while in the case of too high threshold even incorrect matches can be classified as inliers.

using parametrization of $H_{i,j} = KR_jR_i^TK^{-1}$ over camera rotations R_i, R_j and camera matrix K, where n is the number of images and $x_i^{(k)}, x_j^{(k)}$ are the position of k-th point measured in the i-th and j-th images respectively. We parametrize a rotation with a 3-dimensional vector directed parallel to the rotation axis and with the length equal to the rotation angle.

2.4 Experiments

We performed experiments on real datasets taken with Nokia 6303C mobile phone (1536×2048 resolution) and Logitech QuickCam Pro 900 (1600×1200 resolution).

Results for Nokia 6303C. Table 1 presents results we got using Nokia 6303C camera. We compare the auto-calibration results with pattern-based calibration: $f_x^{(err)} = f_x^{(rel)} - 1 = \frac{f_x^{(est)}}{f_x^{(pt)}} - 1$. The auto-calibration algorithm gives relative errors less than 10% on 3 out of 5 datasets. We have showed before that a relative error of less than 10% in camera parameters is enough for getting visually acceptable panoramas.

There are two factors affecting calibration quality. The first factor is the number of images in input dataset, because if the input dataset is too small then it doesn't provide enough information for camera auto-calibration. The second factor is non-zero translation presence, as the auto-calibration method we use was designed under the rotational camera assumption. This assumption is easily violated in practice as a user tends to rotate camera not around its optical center, but around device center (or itself), which is not the same.

Table 1. Relative errors in intrinsic camera parameters

Number of Images	Distance (m)	Relative Error (%)			
		$f_x^{(err)}$	$f_y^{(err)}$	$c_x^{(err)}$	$c_y^{(err)}$
6	2	8.5	11.1	4.7	4.6
7	0.5	-3.8	-2.6	-12.4	-6.2
9	2	-3.4	0.1	2.5	5.4
13	2	2.6	7.6	1.5	8.9
14	30	5.6	6.5	-1.9	4.2

Results for Logitech QuickCam Pro 900. Table 2 presents result we got using Logitech QuickCam Pro 900 camera. For this camera we achieved the relative error less than 9% in comparison with OpenCV pattern based calibration results.

Table 2. Relative errors in intrinsic camera parameters

Number of images	Distance (m)	Relative Error (%)			
		$f_x^{(err)}$	$f_y^{(err)}$	$c_x^{(err)}$	$c_y^{(err)}$
10	2	0.5	5.3	3.6	-0.3
30	2	1.2	4.4	0.7	2
57	2	0.3	3.1	1.5	3.2
10	2	-1.8	0.7	-2.5	1.3
30	2	1.9	6	-0.3	8.6
74	2	0.1	4.3	0.2	7.6

3 Stereo Rig Auto-calibration

3.1 Problem Statement

Stereo camera (or stereo rig) is a rigid couple of two mono cameras described by the model (1). The mapping between a scene 3D point $(X, Y, Z)^T$ and the corresponding pixels $(u_1, v_1)^T, (u_2, v_2)^T$ on two images obtained by the stereo rig looks as follows:

$$\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} \simeq K(R|T) \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{13}$$

$$\begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} \simeq K(R_{rel}R|R_{rel}T + T_{rel}) \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{14}$$

where K is camera intrinsic parameters matrix defined in (2) and it's assumed to be same for the both cameras; R, T are stereo rig rotation 3×3 matrix and translation 3D vector; R_{rel}, T_{rel} are rotation 3×3 matrix and translation 3D vector between the cameras in the stereo rig.

Given pixel coordinates in a monocular image we can reconstruct only a 3D ray that contains the corresponding 3D point. But in the case of a stereo rig two cameras are available, so it's possible to reconstruct 3D scene points. To reconstruct a scene we must know rotation R_{rel} and translation T_{rel} between the cameras in the rig.

We formulate the problem of stereo rig auto-calibration in a way similar to rotation auto-calibration. Input data of the method are keypoints in image pairs taken by a stereo rig, which may undergo arbitrary Euclidean motion, and the correspondences between these keypoints. The final goal is to find camera intrinsic parameters matrix K, cameras relative rotation and translation, i.e. R_{rel} and T_{rel} respectively. Cameras relative rotation and translation are attributed as stereo rig parameters because the cameras are coupled rigidly, as consequence R_{rel} and T_{rel} remain constant over time.

3.2 Camera Intrinsic Parameters Error Effect

A typical task for a stereo rig is 3D reconstruction, i.e. inferring of 3D structure of a scene that is visible on input image pairs. In this section we estimate how errors in camera intrinsic parameters affect reconstruction precision. We make a thought experiment where we vary estimated camera intrinsic parameters while R_{rel} and T_{rel} remain constant and correct.

Suppose we have a 3D point $(X, Y, Z)^T$ and a stereo rig with two cameras located at points $c_1 = (0, 0, 0)^T$ and $c_2 = (0, 0, 1)^T$ respectively, so $T_{rel} = (0, 0, 1)^T$. We assume $R_{rel} = I$: that means both cameras in the stereo rig are oriented the same way. It should be mentioned that the measure unit isn't specified, so an estimation of T_{rel} is defined up to a scale. Regarding the cameras intrinsic parameters we make the same assumptions, as in section 2.2, equation (5). After all the assumptions we've made the stereo rig model looks like this:

$$\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} \simeq K^{(gt)} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{15}$$

$$\begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} \simeq K^{(gt)} \left(\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} - c_2 \right) \tag{16}$$

where $K^{(gt)}$ is defined by (6). Given a 3D point images $(u_1, v_1)^T$, $(u_2, v_2)^T$, and camera intrinsic parameters estimation K_{est} we can reconstruct the point coordinates, they are as follows:

$$\begin{pmatrix} X^{(est)} \\ Y^{(est)} \\ Z^{(est)} \end{pmatrix} = \begin{pmatrix} X + \frac{c_x^{(gt)}(1-c_x^{(rel)})}{f^{(gt)}} Z \\ Y + \frac{c_y^{(gt)}(1-c_y^{(rel)})}{f^{(gt)}} Z \\ f^{(rel)} Z \end{pmatrix} \tag{17}$$

We can build an error function which, obviously, doesn't depend on X and Y:

$$err(K^{(gt)}, K^{(est)}, Z) = \left| \begin{pmatrix} X^{(est)} \\ Y^{(est)} \\ Z^{(est)} \end{pmatrix} - \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \right| = |Z| \left| \begin{pmatrix} \frac{c_x^{(gt)}(1-c_x^{(rel)})}{f^{(gt)}} \\ \frac{c_y^{(gt)}(1-c_y^{(rel)})}{f^{(gt)}} \\ f^{(rel)} - 1 \end{pmatrix} \right| \tag{18}$$

Making the same assumption about view of the matrix $K^{(gt)}$ as in section 2.2, we present the reconstruction error function plots on figure 6.

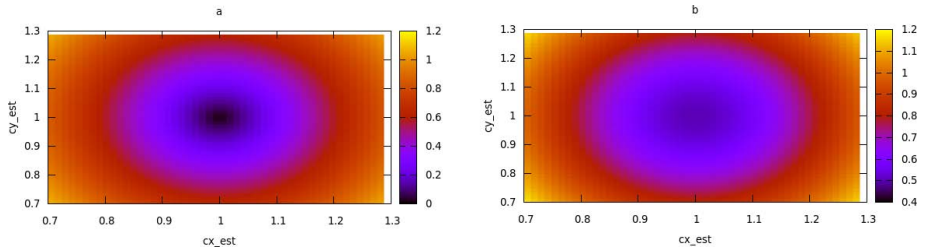


Fig. 6. Reconstruction error plots for $Z = 10$, a) $f^{(rel)} = 1$, b) $f^{(rel)} = 1.05$

Since the estimation of T_{rel} we get is defined up to a scale, we may assume without loss of generality, that the measure unit is 10 cm, so stereo rig baseline is 10 cm and point Z coordinate is 1 m. Then in figure 6, b we see that the reconstruction error achieves about a few centimeters when $f^{(rel)} = c_x^{(rel)} = c_y^{(rel)} = 1.05$, i.e. when there is 5% relative error in all intrinsic camera parameters.

3.3 Proposed Algorithm

In this section a stereo rig auto-calibration algorithm we built is described. The input of the algorithm are a set of image pairs, with keypoints and correspondences between them, and an initial guess for camera intrinsic parameters. The output of the method is refined camera intrinsic parameters K, rotation matrix R_{rel} and translation vector T_{rel} between cameras in the stereo rig. Here is a brief description of the algorithm:

1. For all input stereo pairs compute a fundamental matrix $F_{L,R}$ for points of left and right images of the pairs.
2. Select high quality subset of image pairs for future processing, see section 2.3.

   (a) Build a graph $G = (V, E)$, where vertices V are stereo pairs and E are edges. $(i, j) \in E$ iff matches between the left images from the i-th and j-th stereo pairs satisfy to the estimated fundamental matrix $F_{L,L}^{i,j}$ with error less than a threshold.
   (b) Leave only the biggest connected component in the graph G.
3. For each edge $(i, j) \in E$:
   (a) Compute projective reconstructions for the pairs i and j.
   (b) Find homography $H^{i,j}$ mapping i-th point cloud to j-th point cloud.
   (c) Upgrade the reconstructions from projective to Eucliden using camera intrinsic parameters initial guess and the homography $H^{i,j}$.
4. Refine camera intrinsic parameters matrix K, relative rotation matrix R_{rel} and relative translation vector T_{rel}.

For details on how to find a fundamental matrix, obtain projective reconstruction and other projective geometry related steps we refer to [1].

3.4 Experiments

We performed experiments on real datasets taken by a LG-P920 mobile phone stereo camera (1600×1200 resolution), and a stereo camera Videre STH-DCSG-9cm with resolution 640×480.

Results for LG-P920. To get ground truth stereo rig parameters we calibrated it using OpenCV. We conducted a few dozens of experiments, where intrinsic camera parameters initial relative error was selected from the $[-30\%, 30\%]$ range uniformly.

From table 3 we can see that on the first dataset we achieve less than 10% relative error in intrinsic camera parameters. Standard deviation of the relative error computed over the experiments is less than 8%. On the second dataset relative error in intrinsic camera parameters is less than 11%, while its standard deviation is less than 4%.

We also computed errors for relative rotation matrix R_{rel} and relative translation vector T_{rel} estimations. The ground truth values were computed using OpenCV calibration functionality. Instead of comparing rotation matrices directly, we first convert matrices to rotation vectors and then compare them. Also it should be mentioned that as reconstruction is built up to scale, so the relative translation vector is define up to scale too. That's why we work with translation vectors scaled in such way that its X component equals to 1. The table 4 contains relative rotation and translation vectors obtained using OpenCV calibration.

The results obtained for R_{rel} and T_{rel} using the proposed auto-calibration method are shown in tables 5 and 6.

Table 3. Camera intrinsic parameters relative errors

Dataset ID	Number of Images	Distance (cm)	Mean Relative Error (%)				Relative Error Std. Dev. (%)			
			$f_x^{(err)}$	$f_y^{(err)}$	$c_x^{(err)}$	$c_y^{(err)}$	$f_x^{(err)}$	$f_y^{(err)}$	$c_x^{(err)}$	$c_y^{(err)}$
1	6	30	1.8	2.5	-9.6	4.9	5.2	4.8	5.3	7.6
2	26	30	2.3	2.9	-10.9	1.1	1.1	1.1	1.7	3.9

Table 4. Ground truth rotation and translation vectors

Rotation Vector			Translation Vector		
x	y	z	x	y	z
0	0.04	0	1	-0.04	-0.12

Table 5. Estimated relative rotation and translation vectors means

Dataset ID	Mean Rotation Vector			Mean Translation Vector		
	x	y	z	x	y	z
1	0	0.003	-0.003	1	-0.005	-0.17
2	0.001	0.002	-0.003	1	-0.015	-0.13

Table 6. Estimated relative rotation and translation vector standard deviations

Dataset ID	Rotation Vector Std. Dev.			Translation Vector Std. Dev.		
	x	y	z	x	y	z
1	10^{-4}	$6 \cdot 10^{-5}$	$8 \cdot 10^{-5}$	0	0.002	0.017
2	$1.9 \cdot 10^{-5}$	$2.3 \cdot 10^{-4}$	$1.8 \cdot 10^{-5}$	0	0.002	0.002

Results for Videre STH-DCSG-9cm. To get ground truth stereo rig parameters we used the camera API. We conducted a few dozens of experiments, where intrinsic camera parameters initial relive error was selected from the $[-50\%, 50\%]$ range uniformly.

From table 7 we can see that on the first dataset we achieve less than 4% relative error in intrinsic camera parameters. Standard deviation of the relative error computed over the experiments is less than 2%. On the second dataset relative error in intrinsic camera parameters is less than 6%, while its standard deviation is less than 2%.

Table 7. Camera intrinsic parameters relative errors

Dataset ID	Number of Images	Distance (cm)	Mean Relative Error (%)				Relative Error Std. Dev. (%)			
			$f_x^{(err)}$	$f_y^{(err)}$	$c_x^{(err)}$	$c_y^{(err)}$	$f_x^{(err)}$	$f_y^{(err)}$	$c_x^{(err)}$	$c_y^{(err)}$
1	5	30	2.6	3.6	-1	0.2	1	0.3	1.6	0.9
2	5	30	0.5	-2.4	-5.4	-7.2	0.8	0.9	1.1	2

Table 8. Ground truth rotation and translation vectors

Rotation Vector			Translation Vector		
x	y	z	x	y	z
-0.005	$-2 \cdot 10^{-4}$	0.001	1	-0.004	-0.012

Table 9. Estimated relative rotation and translation vectors means

Dataset ID	Mean Rotation Vector			Mean Translation Vector		
	x	y	z	x	y	z
1	0.018	0.01	0.008	1	0.027	-0.024
2	0.026	0.03	0.001	1	0.007	-0.046

Table 10. Estimated relative rotation and translation vector standard deviations

Dataset ID	Rotation Vector Std. Dev.			Translation Vector Std. Dev.		
	x	y	z	x	y	z
1	1.8^{-4}	$5.8 \cdot 10^{-4}$	$1.7 \cdot 10^{-4}$	0	0.001	0.001
2	$3.7 \cdot 10^{-5}$	0.001	$4 \cdot 10^{-4}$	0	0.001	0.002

We also computed errors for relative rotation matrix R_{rel} and relative translation vector T_{rel} estimations. The table 8 contains relative rotation and translation vectors obtained using the camera API.

The results obtained for R_{rel} and T_{rel} using the proposed auto-calibration method are shown in tables 9 and 10.

4 Conclusion

We investigated the problem of auto-calibration for the case of rotational camera and stereo rig. We built a robust auto-calibration pipeline for both cases, that showed good results on real datasets.

We analyzed impact of errors in camera parameters on final results in such computer vision problem as image mosaicing. While errors in camera parameters can lead to big warping errors, we showed that using modern stitching algorithms relaxes requirements on camera parameters accuracy.

In one specific case we studied how errors in camera intrinsic parameters may affect reconstruction precision in the case of stereo rig auto-calibration.

We showed that it is possible to calibrate cameras without patterns, but the quality of input data is important for achieving accurate auto-calibration.

References

1. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press (2004) ISBN: 0521540518
2. Brown, M., Lowe, D.G.: Automatic Panoramic Image Stitching using Invariant Features. International Journal of Computer Vision 74(1), 59–73 (2007)
3. Zhang, Z.: A flexible new technique for camera calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 1330–1334 (2000)
4. Bradski, G., Kaehler, A.: Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media (2008)
5. Szeliski, R.: Image Alignment and Stitching: A Tutorial. Microsoft Research, TechReport, MSR-TR-2004-92 (2004)
6. Szeliski, R., Shum, H.-Y.: Creating full view panoramic image mosaics and texture-mapped models. In: Computer Graphics, SIGGRAPH Proceedings. Association for Computing Machinery, Inc. (1997)
7. Eden, A., Uyttendaele, M., Szeliski, R.: Seamless Image Stitching of Scenes with Large Motions and Exposure Differences. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2006)
8. Pollefeys, M.: Visual 3D Modeling from Images. Tutorial Notes (2000), `http://www.cs.unc.edu/~marc/tutorial/`
9. Nister, D.: Untwisting a projective reconstruction. International Journal of Computer Vision (2004)
10. Gibson, S., Cook, J., Howard, T., Hubbold, R., Oram, D.: Accurate Camera Calibration for Off-line, Video-Based Augmented Reality. In: ISMAR Proceedings of the 1st International Symposium on Mixed and Augmented Reality (2002)
11. Chen, J., Yuan, B.: Metric 3D reconstruction from uncalibrated unordered images with hierarchical merging. In: IEEE 10th International Conference on Signal Processing (2010)
12. Chandraker, M.K., Agarwal, S., Kriegman, D.J., Belongie, S.: Globally Optimal Algorithms for Stratified Autocalibration. IJCV 90(2), 236–254 (2010)
13. Kwatra, V., Schodl, A., Essa, I., Turk, G., Bobick, A.: Graphcut Textures: Image and Video Synthesis Using Graph Cuts. To appear in Proc. ACM Transactions on Graphics, SIGGRAPH (2003)
14. Adelson, E.H., Burt, P.J.: Multi-resolution Splining Using a Pyramid Image Representation. In: SPIE Applications of Digital Image Processing IV, pp. 204–210 (1983)
15. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: Speeded Up Robust Features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
16. FLANN - Fast Library for Approximate Nearest Neighbors (2011), `http://people.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN`
17. OpenCV stitching module documentation (2012), `http://opencv.itseez.com/modules/stitching/doc/stitching.html`
18. Ji, Q., Dai, S.: Self-Calibration of a Rotating Camera with a Translational Offset. IEEE Trans. on Robotics and Automation 20(1) (February 2004)
19. Junejo, I., Foroosh, H.: Practical PTZ Camera Calibration using Givens Rotations. In: IEEE ICIP (2008)
20. Spizhevoy, A., Eruhimov, V.: Problem of auto-calibration in image mosaicing. In: International Conference on Computer Graphics and Vision, GraphiCon, Conference Proceedings, pp. 27–32 (2012)

GPU Ray Tracing – Comparative Study on Ray-Triangle Intersection Algorithms

Vladimir Shumskiy

Moscow Institute of Physics and Technology
Air Graphics
v.a.shumskiy@gmail.com

Abstract. I present a comparative study on GPU ray tracing implemented for two different types of ray-triangle intersection algorithms used with BVH (Bounding Volume Hierarchy) spatial data structure evaluated for performance on three static scenes. I study how number of triangles placed in a BVH leaf node affects rendering performance. I propose GPU-optimized SIMD ray-triangle intersection method evaluated on GPU for path-tracing and compare it's performance with plain Moller-Trumbore and Unit Triangle intersection methods.

Keywords: ray-triangle intersection, GPU programming, Direct3D, Direct-Compute, performance study, ray tracing, bounding volume hierarches.

1 Introduction

While modern graphics cards (GPUs) allow for general computation in a parallel manner, one of the most prominent applications for a GPU is image synthesis. This is thanks to the inherent parallel nature of ray tracing and other global illumination algorithms – the decomposition of images into pixels provides a natural way of creating individual tasks for many parallel processors. Unlike the GPUs a few years ago, modern ones allow us full programmability similar to general CPUs, while the streaming computation model has its own specific issues. This has to be taken into account when adopting the data structures, traversal algorithms and intersection test routines for ray tracing on GPU architecture.

Testing framework for this paper is based on formerly published papers that implement ray-tracing with spatial data structures in GPU. We use bounding volume hierarchies as described in [5] and few different ray-triangle intersection methods, especially Moller-Trumbore [8] and Unit Triangle [14] routine. While performance of each of those algorithms was successfully studied separately, little attention was paid to how triangle-intersection method can affect spatial data structure traversal performance and vice versa. Furthermore, the performance has not been carefully compared on a current programmable GPU architecture, especially using a cross-vendor APIs like OpenCL, DirectCompute or C++ AMP. In this paper we first present such a comparison study dealing with efficiency of two different types of ray-triangle intersection algorithms for ray tracing on GPU.

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XIX, LNCS 7870, pp. 78–91, 2013.

This paper is further structured as follows. Section 2 summarizes the previous work of ray-triangle intersection on both CPU and GPU and performance comparison on those algorithms. Section 3 describes our choices for implementation. Section 4 shows the result from measurements on two GPUs for a set of scenes. Further it discusses the bottlenecks of a contemporary GPU architecture for ray tracing algorithms. Section 5 concludes the paper with possible perspectives for future work.

2 Previous Work

Due to important role in computer graphics plenty of research has been done in the field of intersection testing algorithms. Algorithms proposed by Snyder and Barr [11], Badouel [3], Moller-Trumbore [8], Woop [14], Wald and Shevtsov et al. [10], have been successfully compared and studied [9], [2], [3], [8]. In our research we divide algorithms on those which use precomputed data and on those which do not. Based on previous work we decided to use Moller-Trubmore algorithm as a minimal storage, fast non-precomputed type and Swen Woop's Unit Triangle Test as the precomputed one as it requires only 48 bytes per triangle and doesn't need to store vertex list. In this section we describe chosen algorithms along with BVH spatial data structure.

We omit ray-packet algorithms in our work, because the coherence of the rays within the packet is very important since the vector instructions are fully used only if all rays go through the same branch of computation. In situations like physical simulation, collision detection or ray-tracing in scenes, where rays bounces into multiple directions (spherical or bumpmapped surfaces), coherent ray packets break down very quickly to single rays or do not exist at all. Ray-packets have proven [1], [6] to be ineffective in the above mentioned tasks.

2.1 Moller-Trumbore's Algorithm

The algorithm proposed by Moller and Trumbore does not test intersection with the triangle's embedding plane and therefore does not require the plane equation parameters. This is a big advantage mainly in terms of memory consumption especially on the GPU execution performance. The algorithm goes as follows [8]:

1. In a series of transformations the triangle is first translated into the origin and then transformed to a right-aligned unit triangle in the y-z plane, with the ray direction aligned with x. This can be expressed by a linear equation

$$\begin{pmatrix} t \\ u \\ v \end{pmatrix} \quad \frac{1}{P \cdot E_1} \begin{pmatrix} Q & E_2 \\ P & T \\ Q & D \end{pmatrix} \tag{1}$$

Where $E_1 = V_1 - V_0$, $E_2 = V_2 - V_0$, $T = O - V_0$, $P = D \times E_2$ and $Q = T \times E_1$.

2. This linear equation can now be solved to find the barycentric coordinates of the intersection point (u,v) and its distance t from the ray origin.

2.2 Unit Triangle Algorithm

The so called Unit Triangle intersection algorithm performs ray transformations and consists of two stages [14]. First the ray is transformed, using a triangle specific affine triangle transformation, to a coordinate system, in which the triangle looks like the unit triangle Δunit with the edge points (1, 0, 0), (0, 1, 0) and (0, 0, 0). In the second stage, a simple intersection test of the transformed ray with the unit triangle is done. The affine triangle transformation to a triangle $\Delta = (a, b, c)$ is an affine transformation $T\Delta(x) = m \cdot x + n$ with $m \in MatR(3 \times 3)$ and $n \in R3$ that maps the triangle Δ to the unit triangle Δunit.

2.3 Bounding Volume Hierarchies

Bounding volume hierarchies were successfully implemented on GPU. Thrane and Simonsen [12] in fact compare kd-trees, uniform grids, and bounding volume hierarchies implemented on a GPU (2005-year hardware). They conclude the performance of BVHs is low, however higher than the performance of other two data structures when no ray packets are used. Carr et al. [4] implemented a variant of BVHs in combination with geometry images. Günther et al. [5] use ray packets and yield interactive performance comparable or exceeding CPU-based implementation, but only for primary and shadow rays. Recently, Lauterbach et al. [7] present an algorithm for fast BVH construction on a GPU, where they report performance comparable to kdtrees [16] only for one scene. Torres et al. [13] published an algorithm for stack-less BVH traversal, where the use of stack is replaced by ropes connecting the nodes of a BVH in a sibling order.

3 Implementation

Fig. 1. Testing scenes: Stanford Buddha, Bunny and Dragon

We have implemented a standalone compact program called RenderBro, that does not need the support of other 3rd party libraries along with Autodesk 3DS Max Plugin (both can be obtained at http://renderbro.com). Standalone program is capable of loading 3D scene form OBJ file format along with MTL materials files. 3DS Max plugin is capable to work with any kind of geometry loaded into editor in question. While the data structures are built offline on a CPU, the created data structures and scene geometry are transferred to a GPU and used for ray tracing algorithm entirely on the GPU. This methodology is sufficient to study the efficiency of shooting rays

with different intersection algorithms. The traversal and intersection algorithms were implemented using Microsoft DirectX DirectCompute (Compute Shaders). Although this implementation limits target platforms to Microsoft Windows, it gives freedom to choose any GPU vendor to run GPU ray tracing, such as ATI/AMD, NVIDIA, Intel. DirectCompute code can be translated to C++ AMP version, which can be executed on any OS. We designed our solution to support as many hardware as possible, though only DirectX 10 compatible or newer hardware is supported. All shaders were compiled using latest Windows SDK 8.0 D3DCompiler_45.

In Moller-Trumbore setting the geometry of a scene consisting solely of triangles is represented by a list Lv of vertices and list of materials Lm, where each triangle has a list of three indices to Lv plus an index to the Lm. In the Unit Triangle Test each triangle is represented directly by the affine transformation matrix. For our tests we implemented path-trace setting with physically-based importance sampled shading including Phong, Blinn-Phong, Lambertian Diffuse, Oren-Nayar Diffuse, Ashikhmin-Shirley, Glass and Perfect Mirror BSDFs.

The BVHs were built in top-down fashion with surface area heuristics using the centroids of bounding boxes for scene triangles, following the paper by Günther et al. [5]. Each BVH node consists of AABB extents and indices to child nodes. If it's a leaf node, child indices are replaced with triangle offset along with triangle count. Those parameters are packed in 32 byte BVH node structure. As a BVH does not need to store the min and max intersection distances along the ray, only the node address is saved to the stack. Stack does not need to be shortened to only several entries, which minimizes the number of traversal steps. Serialization of write operation may occur as threads record their information. Each BVH node can contain any number of triangles. This hypothetically reduces the number of nodes of a hierarchy along with GPU memory needed and gives space for GPU traversal optimization. Traversal is done with "while-if" method and listed in appendix 8.3.

3.1 GPU Optimized Intersections

Moller-Trumbore and Unit triangle intersection tests are pretty straight-forward to implement and require a little knowledge of GPU architecture (see appendix, section 8.1). Such methods can experience poor register usage in architectures like VLIW which is used in many AMD GPUs.

In this work we introduce a method that strongly benefit from denser GPU register usage. The main idea is to exploit the wide vector width SIMD (Single Instruction Multiple Data) by testing intersection with one ray and four triangles at a time.

Firstly, at precompute time we need to try to fill BVH with four triangles per node, so each node will contain four pointers to triangle list. If this is not possible we would have one triangle per node at worst. Secondly, we need to align GPU scene data according to BVH node structure. So if particular node has only one triangle, we need to place three degenerate triangles in a triangle list to fulfill the alignment. Of course, this will result in a GPU memory footprint by the means of performance. Thirdly, when performing BVH traverse we will be able to linearly fetch four triangles. Here we will need to construct additional vectors, like:

```
// fetch triangle vertices
float3 v01, v11, v21;
float3 v02, v12, v22;
float3 v03, v13, v23;
float3 v04, v14, v24;
...
// construct per-component dir & orig vectors
float4 dir4x = ray.dir.xxxx;
float4 dir4y = ray.dir.yyyy;
float4 dir4z = ray.dir.zzzz;
float4 orig4x = ray.orig.xxxx;
float4 orig4y = ray.orig.yyyy;
float4 orig4z = ray.orig.zzzz;
```

This allows us to compute temporary values on per-component SIMD basis, so all non SIMD operations like scalar addition and multiplication can be performed on each triangle simultaneously. For example, when performing scalar multiplication on GPU we use only one computing block while using new approach we will perform four multiplication operations by the same cost (see appendix 8.2 for full listing):

```
// one triangle per pass
float divisor = dot(pvec, e1);

// four triangles per pass
float4 divisor4 = pvecx*e1x + pvecy*e1y + pvecz*e1z;
```

Table 1. Test scene properties, number of triangles per BVH leaf node, rendering times for different ray-triangle intersection method for Path Trace setting with 500 samples and max ray depth of 16. GPU NVIDIA GeForce GT 240M. Image resolution: 800x600.

Scene	Triangles per BVH node	GPU Scene Size, MB	Moller-Trumbore, seconds (average)	Unit Triangle, seconds (average)	Quad Moller-Trumbore, seconds (average)	Quad Unit Triangle, seconds (average)
Buddha, 100.006 triangles	1	9,35	103,33	103,94	-	-
	2	7,26	95,13	95,44	-	-
	3	6,38	92,72	93,07	-	-
	4	5,89	90,91	91,17	89,67	85,36
	8	5,15	92,57	94,01	-	-
	16	4,73	104,53	108,83	-	-
Bunny, 69.678 triangles	1	6,51	72,18	73,52	-	-
	2	4,87	65,48	66,67	-	-
	3	4,45	62,07	63,94	-	-
	4	3,99	60,98	62,17	57,35	52,84
	8	3,45	62,1	63,29	-	-
	16	3,14	69,67	70,61	-	-

Table 1. (*continued*)

Dragon, 100.012 triangles	1	9,35	210,89	211,67	-	-
	2	7,26	210,71	211,61	-	-
	3	6,35	210,68	211,44	-	-
	4	5,86	210,52	211,21	173,37	168,85
	8	5,11	210,69	211,37	-	-
	16	4,65	210,91	211,66	-	-

Table 2. Test scene properties, number of triangles per BVH leaf node, rendering times for different ray-triangle intersection methods for Path Trace setting with 500 samples and max ray depth of 16. GPU AMD Radeon HD 6870. Image resolution: 800x600.

Scene	Triangles per BVH node	GPU Scene Size, MB	Moller-Trumbore, seconds (average)	Unit Triangle, seconds (average)	Quad Moller-Trumbore, seconds (average)	Quad Unit Triangle, seconds (average)
Buddha, 100.006 triangles	1	9,35	49,27	49,43	-	-
	2	7,26	49,78	50,31	-	-
	3	6,38	51,94	52,18	-	-
	4	5,89	54,15	55,16	34,3	31,37
	8	5,15	66,3	68,64	-	-
	16	4,73	95,22	98,38	-	-
Bunny, 69.678 triangles	1	6,51	33,27	33,79	-	-
	2	4,87	33,57	34,41	-	-
	3	4,45	34,68	35,55	-	-
	4	3,99	38,05	39,06	23,9	20,64
	8	3,45	46,57	47,86	-	-
	16	3,14	60,26	62,56	-	-
Dragon, 100.012 triangles	1	9,35	99,27	99,75	-	-
	2	7,26	99,41	100,96	-	-
	3	6,35	104,6	105,61	-	-
	4	5,86	108,35	111,5	57,99	54,91
	8	5,11	132,57	138,06	-	-
	16	4,65	188,55	198,07	-	-

Table 3. Test scene properties, number of triangles per BVH leaf node, rendering times for different ray-triangle intersection methods for Path Trace setting with 1000 samples and max ray depth of 16. GPU NVIDIA GeForce GTX 560. Image resolution: 800x600.

Scene	Triangles per BVH node	GPU Scene Size, MB	Moller-Trumbore, seconds (average)	Unit Triangle, seconds (average)	Quad Moller-Trumbore, seconds (average)	Quad Unit Triangle, seconds (average)
Buddha	4	5,89	66,59	67,07	63,87	62,13
Bunny	4	3,99	56,82	57,52	53,18	51,17
Dragon	4	5,86	84,76	85,94	80,37	78,64

Table 4. Test scene properties, number of triangles per BVH leaf node, rendering times for different ray-triangle intersection methods for Path Trace setting with 1000 samples and max ray depth of 16. GPU AMD HD7850. Image resolution: 800x600.

Scene	# Triangles per BVH node	GPU Scene Size, MB	Moller-Trumbore, seconds (average)	Unit Triangle, seconds (average)	Quad Moller-Trumbore, seconds (average)	Quad Unit Triangle, seconds (average)
Buddha	4	5,89	38,03	39,41	33,43	31,35
Bunny	4	3,99	36,48	37,97	31,64	29,85
Dragon	4	5,86	65,56	67,09	56,72	54,92

4 Results

In this section we describe the results for measurement on three different scenes. We used scenes from individual objects courtesy of Stanford scene repository. These scenes are frequently used to test the performance of ray tracing and global illumination algorithms. Images were rendered in 800x600 pixels resolution. All performance results in this paper were measured on 4 different GPUs:

1. NVIDIA GeForce GT 240M (2009), 48 CUDA cores on 1210MHz, 1 GByte of memory with bandwidth of 54.4 GB/sec.
2. AMD HD 6870 (2010), 2 TFLOPs, 1120 Stream Processors on 900MHz, 1GByte of memory with bandwidth of 134.4 GB/sec.
3. NVIDIA GeForce GTX 560 (2011), 2.1 TFLOPs, 336 CUDA cores on 1620-1900MHz, 1 GByte of memory with bandwidth of 128 GB/s.
4. AMD HD 7850 (2012), 1.76 TFLOPs, 1024 Stream Processors on 860MHz, 2GByte of memory with bandwidth of 153.6 GB/s.

The static properties of data structures for all three scenes along with average computation time for path-tracing are shown in Tables 1 and 2. Performance results for BVHs build with different count of triangles per leaf node are shown in columns 4 and 5. Those results demonstrate that both the number of triangles per leaf node and the selected intersection method remarkably affect the performance. Results lead us to assumption that 4 triangles per leaf node is the optimal number for BVH traversal.

Moller-Trumbore kernel had proven to be up to 5% faster than Unit Triangle in all tests while Quad Unit Triangle kernel shown to be up to 14% faster than Quad Moller-Trumbore.

Our proposed Quad Unit-Triangle method brings moderate improvements of 5% to 11% on different generations of NVIDIA hardware except for Dragon scene setup on GT 240M GPU where it gain about 18% (tables 1 and 3). The situation is better on VLIW AMD/ATi hardware where it had shown to be up to 2x times faster than Moller-Trumbore (table 2). Furthermore we managed to analyze performance of the latest GPU generation AMD HD 7850 (table 4). As we expected, it showed consistent results in spite of new architecture and showed that Quad Unit-Triangle is approximately 20% faster than Moller-Trumbore method.

Good results are gained for VLIW architecture used in AMD GPUs where more functional units are available and may be scheduled by the compiler or hardware simultaneously. According to description of VLIW architecture, it's possible to perform compute operations along with memory access. We assume that this result is mainly achieved by performing more linear memory access to GPU global memory, avoiding branching by unrolling triangle-intersection loop and taking advantage of denser GPU register usage.

Things are bit different for NVIDIA GPUs like Fermi which internally operate in a SIMD manner by ganging multiple (32) scalar threads together into SIMD warps. If a warp's threads diverge, the warp serially executes both branches, temporarily disabling threads that are not on that path. Thus, ray tracing performance certainly can benefit from loop unrolling and more linear memory access. But the true cause of performance improvement lies much deeper in the GPU architecture and goes beyond the scope of this article.

5 Conclusion and Future Work

We have shown that triangle intersection routines that tend to have good performance when used separately can behave badly when used together with acceleration structures like BVH's due to incoherent memory access, lack of registers, and so on. So we focus our work on finding the robust combination of triangle intersection method and spatial data structure. For now it's the Quad Triangle intersection method used along with BVH.

As future work, the implementation could be extended by several other data structures, such as Kd-Trees, Uniform Grids along with few different ray-triangle intersection methods that can be efficiently mapped to GPUs and are likely to show unexpected results when used together.

Furthermore, shown results can hardly be called ambiguous as they are pretty much view dependent. Dragon scene showed 18% performance improvement on NVIDIA GT 240M for one angle of view. For different angles performance may vary, showing both improvement and deterioration. So, a part of our future work will be devoted to analysis of more complex and dynamic scenes where view dependency is not that great.

Unfortunately, we didn't manage to make in-depth performance study on latest discreet NVIDIA and integrated Intel GPUs. We would like to complete out research by taking this GPUs into account as a part of future work.

References

1. Aila, T., Laine, S.: Understanding the Efficiency of Ray Traversal on GPUs. In: Proceedings of High-Performance Graphics 2009, pp. 145–150. ACM, New York (2009)
2. Arenberg, J.: Ray-Triangle Intersection with Barycentric Coordinates. In: Haines, E. (ed.) Ray Tracing News, November 4, vol. 1(11) (1988)
3. Badouel, F.: An efficient Ray-Polygon intersection, Graphic Gems, pp. 390–393. Academic Press (1990)
4. Carr, N.A., Hoberock, J., Crane, K., Hart, J.C.: Fast GPU ray tracing of dynamic meshes using geometry images. In: GI 2006: Proceedings of Graphics Interface 2006, pp. 203–209. Canadian Information Processing Society, Toronto (2006)
5. Günther, J., Popov, S., Seidel, H.-P., Slusallek, P.: Realtime Ray Tracing on GPU with BVH-based Packet Traversal. In: Proceedings of the IEEE/Eurographics Symposium on Interactive Ray Tracing 2007, pp. 113–118 (September 2007)
6. Havel, J., Herout, A.: Yet Faster Ray-Triangle Intersection (Using SSE4). IEEE Transactions on Visualization and Computer Graphics 16(3), 434–438 (2010), doi:10.1109/TVCG.2009.73
7. Lauterbach, C., Garland, M., Sengupta, S., Luebke, D., Manocha, D.: Fast BVH Construction on GPUs. Computer Graphics Forum 28(2), 375–384 (2009) (Proceedings of Eurographics 2007)
8. Möller, T., Trumbore, B.: Fast, minimum storage ray-triangle intersection. Journal on Graphic Tools 2(1), 21–28 (1997)
9. Segura, R.J., Feito, F.R.: Algorithms to Test RayTriangle Intersection. Comparative Study. In: Skala, V. (ed.) WSCG 2001 Conference Proceedings (February 2001)
10. Shevtsov, M., Soupikov, A., Kapustin, A.: Ray-Triangle Intersection Algorithm for Modern CPU Architectures. In: Proceedings of GraphiCon 2007, pp. 33–39 (2007)
11. Snyder, M., Barr, A.H.: Raytracing complex models containing surface tesselations. In: Proceedings of the 14th Annual Conference on Computer Graphics, vol. 21(4), pp. 119–128 (1987)
12. Thrane, N., Simonsen, L.O.: A comparison of acceleration structures for GPU assisted ray tracing. M.Sc. Thesis, University of Aarhus, Denmark (2005)
13. Torres, R., Martin, P.J., Gavilanes, A.: Ray Casting using a Roped BVH with CUDA. In: 25th Spring Conference on Computer Graphics (SCCG 2009), Budmerice, Slovakia, pp. 107–114 (April 2009)
14. Woop, S., Schmittler, J., Slusallek, P.: RPU: A Programmable Ray Processing Unit for Realtime Ray Tracing. ACM Transactions Graphics 24(3), 434–444 (2005)

15. Wald, I.: Realtime ray tracing and interactive global illumination. PhD thesis, Saarland University (2004)
16. Zhou, K., Hou, Q., Wang, R., Guo, B.: Real-time KD-tree construction on graphics hardware. In: SIGGRAPH Asia 2008: ACM SIGGRAPH Asia 2008 Papers, New York, pp. 1–11 (2008)

A Appendix

A.1 Casual Moller-Trumbore GPU Ray-Triangle Intersection Routine (HLSL code)

```
float intersect(float3 orig, float3 dir, float3 v0,
float3 v1, float3 v2)
{
  float3 e1 = v1 - v0;
  float3 e2 = v2 - v0;

  float3 normal = normalize(cross(e1, e2));
  float b = dot(normal, ray.dir);

  float3 w0 = ray.orig - v0;
  float a = -dot(normal, w0);
  float t = a / b;

  float3 p = ray.orig + t * ray.dir;
  float uu, uv, vv, wu, wv, inverseD;
  uu = dot(e1, e1);
  uv = dot(e1, e2);
  vv = dot(e2, e2);

  float3 w = p - v0;
  wu = dot(w, e1);
  wv = dot(w, e2);
  inverseD = uv * uv - uu * vv;
  inverseD = 1.0f / inverseD;

  float u = (uv * wv - vv * wu) * inverseD;
  if (u < 0.0f || u > 1.0f)
    return -1.0f;

  float v = (uv * wu - uu * wv) * inverseD;
  if (v < 0.0f || (u + v) > 1.0f)
    return -1.0f;

  UV = float2(u,v);
  return t;
}
```

A.2 Quad Moller-Trumbore GPU Triangle-Ray Intersection Routine (HLSL code)

```
float intersect(float3 orig, float3 dir, float3 v01,
float3 v11, float3 v21, float3 v02, float3 v12, float3
v22, float3 v03, float3 v13, float3 v23, float3 v04,
float3 v14, float3 v24)
{
  float3 e11 = v11 - v01;
  float3 e21 = v21 - v01;
  float3 e12 = v12 - v02;
  float3 e22 = v22 - v02;
  float3 e13 = v13 - v03;
  float3 e23 = v23 - v03;
  float3 e14 = v14 - v04;
  float3 e24 = v24 - v04;
  float4 v0x = float4(v01.x, v02.x, v03.x, v04.x);
  float4 v0y = float4(v01.y, v02.y, v03.y, v04.y);
  float4 v0z = float4(v01.z, v02.z, v03.z, v04.z);
  float4 e1x = float4(e11.x, e12.x, e13.x, e14.x);
  float4 e1y = float4(e11.y, e12.y, e13.y, e14.y);
  float4 e1z = float4(e11.z, e12.z, e13.z, e14.z);
  float4 e2x = float4(e21.x, e22.x, e23.x, e24.x);
  float4 e2y = float4(e21.y, e22.y, e23.y, e24.y);
  float4 e2z = float4(e21.z, e22.z, e23.z, e24.z);
  float4 dir4x = ray.dir.xxxx;
  float4 dir4y = ray.dir.yyyy;
  float4 dir4z = ray.dir.zzzz;
  float4 pvecx = dir4y*e2z - dir4z*e2y;
  float4 pvecy = dir4z*e2x - dir4x*e2z;
  float4 pvecz = dir4x*e2y - dir4y*e2x;
  float4 divisor = pvecx*e1x + pvecy*e1y + pvecz*e1z;
  float4 invDivisor = float4(1, 1, 1, 1) / divisor;
  float4 orig4x = ray.orig.xxxx;
  float4 orig4y = ray.orig.yyyy;
  float4 orig4z = ray.orig.zzzz;
  float4 tvecx = orig4x - v0x;
  float4 tvecy = orig4y - v0y;
  float4 tvecz = orig4z - v0z;
  float4 u4;
  u4 = tvecx*pvecx + tvecy*pvecy + tvecz*pvecz;
  u4 = u4 * invDivisor;
  float4 qvecx = tvecy*e1z - tvecz*e1y;
  float4 qvecy = tvecz*e1x - tvecx*e1z;
  float4 qvecz = tvecx*e1y - tvecy*e1x;
  float4 v4;
  v4 = dir4x*qvecx + dir4y*qvecy + dir4z*qvecz;
  v4 = v4 * invDivisor;
  float4 t4;
  t4 = e2x*qvecx + e2y*qvecy + e2z*qvecz;
```

```
  t4 = t4 * invDivisor;
  float t = -1.0f;

  if(t4.x < t && t4.x > 0)
    if(u4.x >= 0 && v4.x >= 0 && u4.x + v4.x <= 1)
      t = t4.x;

  if(t4.y < t && t4.y > 0)
    if(u4.y >= 0 && v4.y >= 0 && u4.y + v4.y <= 1)
      t = t4.y;

  if(t4.z < t && t4.z > 0)
    if(u4.z >= 0 && v4.z >= 0 && u4.z + v4.z <= 1)
      t = t4.z;

  if(t4.w < t && t4.w > 0)
    if(u4.w >= 0 && v4.w >= 0 && u4.w + v4.w <= 1)
      t = t4.w;

  return t;
}
```

A.3 BVH Traversal Routine (HLSL code)

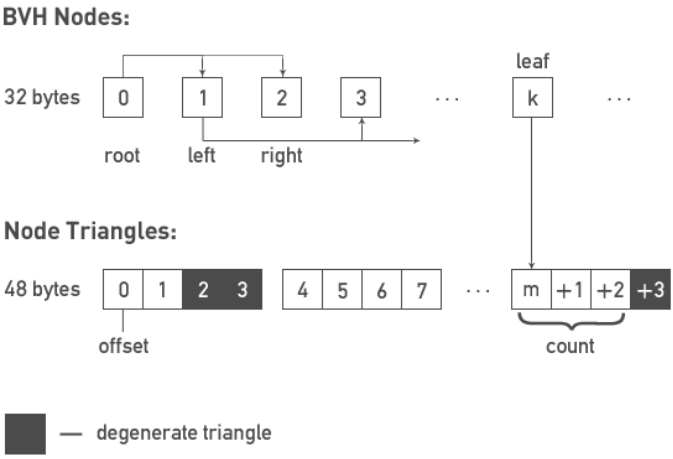
```
struct BvhCell
{
 float4 vmin; //float3 min + uint children
 float4 vmax; //float3 max + uint count
};
bool RayIntersectScene(Ray ray)
{
 uint stack[64], stackPos = 0, node = 0;
 float t  = FLT_MAX;
 bool intersect = false;
 BvhCell cellLeft, cellRight;
 BvhCell current = GetNode(node);

 while(1)
 {
  uint count  = GetNodeTriangleCount(current);
  if(count > 0)
  { // Leaf Node
   uint offset = GetNodeTriangleOffset(current);
   intersect = RayTrisTest(ray, t, offset, count);
   if(stackPos > 0)
   {
    node = stack[--stackPos];
```

```
      current  = LoadNode(node);
     }
     else return intersected;
    }
    else
    {
     uint  leftNode  = GetLeftChildID(node);
     uint  rightNode = GetRightChildID(node);
     float lMin, rMin;
     cellLeft  = GetNode(leftNode);
     cellRight = GetNode(rightNode);
     bool wantLeft = RayAABBTest(ray, cellLeft, lMin);
     bool wantRight = RayAABBTest(ray, cellRight,
                                  rMin);
     if(wantLeft && wantRight)
     {
      bool firstLeft = leftMin < rightMin;
      if(firstLeft)
      {
       current = cellLeft;
       node    = leftNode;
       stack[stackPos++] = rightNode;
      }
      else
      {
       current = cellRight;
       node    = rightNode;
       stack[stackPos++] = leftNode;
      }
     }
     else if(wantRight)
     {
      current = cellRight;
      node    = rightNode;
     }
     else if(wantLeft)
     {
      current = cellLeft;
      node    = leftNode;
     }
     else
     {
      if(stackPos > 0)
      {
       node = stack[--stackPos];
       current = GetNode(node);
      } else return intersected;
```

```
    }
   }
  }
  return intersected;
}
```

A.4 BVH Data Layout



Learning Graph Laplacian for Image Segmentation

Sergey Milyaev[1] and Olga Barinova[2]

[1] Radiophysics Department, Voronezh State University, Voronezh, Russia
[2] Department of Computational Mathematics and Cybernetics Lomonosov Moscow State University, Moscow, Russia

Abstract. In this paper we formulate the task of semantic image segmentation as a manifold embedding problem and solve it using graph Laplacian approximation. This allows for unsupervised learning of graph Laplacian parameters individually for each image without using any prior information. We perform experiments on GrabCut, Graz and Pascal datasets. At a low computational cost proposed learning method shows comparable performance to choosing the parameters on the test set. Our framework for semantic image segmentation shows better performance than the standard discrete CRF with graph-cut inference.

1 Introduction

We consider the task of semantic image segmentation that implies assigning a label from a given set to each image pixel. Various discrete CRF models have been proposed for this task [1] [2], [3]. It was shown that learning the parameters of CRF improves its performance [4], [5]. In this work we propose an alternative view on semantic image segmentation.

Methods based on graph Laplacians show state-of-the-art results for interactive image segmentation [6] and image matting [7]. They require just a few local computations and solving one sparse linear system, which can be done very efficiently. In this work we propose a formulation of image segmentation task in terms of manifold embedding and discretize the problem using graph Laplacian approximation.

Graph Laplacian methods have the parameters very similar to those of discrete CRFs. While a remarkable progress has been done in the direction of learning the parameters of discrete CRFs (see e.g. [4], [5]). However the methods for learning parameters of discrete CRF are not applicable to graph Laplacian. Thus the parameters of graph Laplacian are usually chosen by validation on hold-out dataset. The use of validation limits the potential number of parameters used. Moreover, the optimal values of parameters can vary significantly from one image to another, therefore choosing the parameters individually for each image is desirable.

Our formulation of image segmentation problem leads to a novel method for unsupervised learning of graph Laplacian parameters, which is the main contribution of this paper. Our method is designed specifically for the task of semantic image segmentation and provides the values of parameters individually for each test image without using any kind of supervision. Proposed method is computationally efficient and achieves performance comparable to choosing the parameters on the test set, which eliminates the need of using hold-out set or cross-validation. In experimental comparison on Graz

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XIX, LNCS 7870, pp. 92–106, 2013.

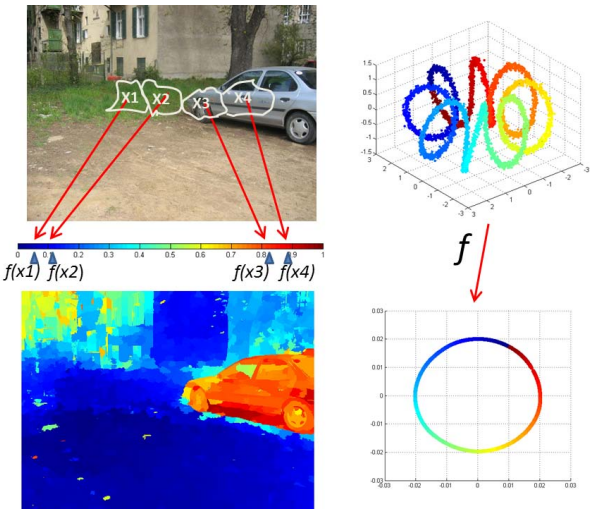


Fig. 1. We formulate semantic image segmentation task as one-dimensional manifold embedding problem. This allows for unsupervised learning of graph Laplacian parameters individually for each image.

and Pascal datasets shows proposed method shows better performance than the standard discrete CRF with graph-cut inference.

The remainder of the paper if organized as follows. We start by discussing related work. In section 3 we describe the image segmentation framework proposed in this paper. In section 4 we present our method for unsupervised learning of graph Laplacian parameters. We proceed to the experimental evaluation of the proposed method.

2 Related Work

The task of semantic image segmentation implies assigning a label from a given set to each image pixel. Various discrete CRF models have been proposed for this task [1] [2], [3]. Learning the parameters of CRF can improve performance of semantic image segmentation [4], [5]. In this work we use an alternative formulation of semantic image segmentation problem that leads to using graph Laplacian instead of discrete CRF.

Methods based on graph Laplacians have emerged recently and proved very efficient for interactive image segmentation [6] and image matting [7]. Graph Laplacian methods allow interpretation in terms of MAP estimation in real-valued CRF [8]. A few other interpretations of graph Laplacian methods have been suggested in the literature.

In [6] Grady suggested explanation of using Laplacians for interactive segmentation in terms of random walks. In [9] the use of graph Laplacian for interactive image segmentation was explained in terms of transductive inference. Hein et al. [10] showed that graph Laplacian provides a good approximation for s-weighted Laplace operator. Therefore, graph Laplacians propose a discrete alternative to the problem of finding a smooth function such that it's values in seed pixels are close to the associated labels and it is allowed to vary only on low-density regions of the input space.

In contrast to these works we derive the graph Laplacian by viewing image segmentation as a manifold embedding task. In contrast to [11] we use manifold embedding for semantic image segmentation and not the unsupervised image segmentation. This formulation of semantic image segmentation allows for unsupervised learning of graph Laplacian parameters.

The idea of our learning method is based on the properties of graph Laplacian approximation of Laplace-Beltrami operator studied in [12]. Coifman at al. in [13] proposed a method for automatic selection the kernel bandwidth of graph Laplacian for the problem of optimal manifold embedding. In contrast to [13] we design the method specifically for the task of semantic image segmentation. While the method proposed in [13] aims at choosing one parameter (kernel bandwidth), our method can handle multiple parameters.

3 One-Dimensional Manifold Embedding for Semantic Image Segmentation

First we discuss the task of manifold embedding and then explain our formulation of image segmentation problem.

3.1 Manifold Embedding

Suppose we have a set of input points $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathbb{R}^l$. Let $d : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}$ be a symmetrical function giving the distance in $\mathbb{R}^l$. The optimal manifold embedding task is to find a smooth differentiable function f that maps the input space $\mathbb{R}^l$ on the embedded Riemanian manifold M of dimensionality m ($m < l$) (see Figure 1, left column). The function f should preserve distances between the points, such that if $d(\mathbf{x}_i, \mathbf{x}_j)$ is small, then $\|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|$ should be small.

Let us focus on the case when the dimension of manifold M equals one ($M = \mathbb{R}$). Consider two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^l$. They are mapped to $f(\mathbf{x}), f(\mathbf{y}) \in \mathbb{R}$ respectively. It can be shown [12] that

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq d(\mathbf{x}, \mathbf{y}) \|\nabla f(x)\| + o(d(\mathbf{x}, \mathbf{y})) \,, \tag{1}$$

where $\nabla f(x)$ is the gradient of function $f(x)$. Thus we see that $\nabla f(x)$ provides us with the measure of how far apart f maps nearby points.

We consider the problem of *initialized one-dimensional manifold embedding* when 1) $M = \mathbb{R}$ and 2) initial estimates $y_1, ..., y_N$ of $f(\mathbf{x_1}), ..., f(\mathbf{x_N})$ in $\mathbb{R}$ are given. Suppose we know confidences $c_i \geq 0, i = 1, ..., N$ that reflect our belief in initial estimates of $f(\mathbf{x_i}), i = 1, ..., N$. Using (1) the problem of initialized one-dimensional manifold embedding can be formulated as minimization the following energy with respect to f

$$E(f) = \sum_i c_i \left(y_i - f(\mathbf{x}_i)\right)^2 + \int_M ||\nabla f||^2 dV \,, \tag{2}$$

where the integral is taken with respect to a standard measure on a Riemanian manifold. The first term in (2) guarantees that corresponding one-dimensional vectors $f(\mathbf{x}_i)$ are

close to their initial estimates y_i. The second term guarantees that if the points $\mathbf{x}_i, \mathbf{x}_j$ are close in the input space then their images $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ are close in M.

It follows from the Stokes' theorem that $\int_M ||\nabla f||^2 dV = \int_M \triangle_M(f) f dV$, where $\triangle_M(f)$ is the Laplace-Beltrami operator. It is a second order differential operator defined as the divergence of the gradient of a function defined on M.

In many cases finding the mapping f explicitly is not required. The goal then is to find a set of points $f(\mathbf{x}_1), ..., f(\mathbf{x}_N) \in M$ such that represent $\mathbf{x}_1, ..., \mathbf{x}_N$.

3.2 Image Segmentation as the Manifold Embedding Problem

For the sake of clarity first we consider the task of object/background image segmentation. We aim to find real-valued alpha-matting coefficients for each image pixel, the segmentation is then done by thresholding the result. Below we formulate image segmentation problem as the problem of initialized one-dimensional manifold embedding.

Suppose each image pixel is mapped in a feature space $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathbb{R}^l$. For example the features can include the spatial coordinates and color of the pixels. Suppose we have defined a distance function between two pixels $d : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}$ that tells how likely it is that both pixels belong to object/background.

Suppose that for each image pixel we know the output of some local model $0 \leq y_i \leq 1, i = 1, ..., N$ that tells how likely it is that the pixel is a part of the object. Suppose also that we know confidences $c_i \geq 0, i = 1, ..., N$ that indicate how much belief we put in the local model.

Our goal is to find real-valued $f_1, ..., f_N$ that refine the outputs of local model $y_1, ..., y_N$. We require that $f_1, ..., f_N$ lie in the optimally embedded one-dimensional manifold M and each f_i corresponds to $\mathbf{x}_i$. Therefore the problem of image segmentation reduces to minimization of energy (2).

3.3 Approximation of Laplace-Beltrami Operator

We will now define a graph Laplacian that is an approximation of Laplace-Beltrami operator. Denote weight matrix by $W : W_{ij} = \exp\left(-d\left(\mathbf{x}_i, \mathbf{x}_j\right)\right)^2\Big)$ (in this work we consider only Gaussian kernel). Let $g_i = \sum_j w_{ij}$ stand for a sum of W along the i-th row. Denote diagonal matrix with values g_i on diagonal by D. Graph Laplacian is defined as a matrix $L = W - D$.

Belkin et al. [14] showed that graph Laplacian L converges to Laplace-Beltrami operator in the limit $N \to \infty$. In this sense, the graph Laplacian is a numerical machinery for approximating a specific operator on the underlying manifold, by using a finite subset of points.

3.4 Discretization of the Problem with Graph Laplacian

Using approximation of Laplace-Beltrami operator by graph Laplacian the problem (2) reduces to minimization of the following energy function with respect to vector $\mathbf{f} = (f_1, ..., f_N)$:

$$E(\mathbf{f}) = \sum_i c_i \left(f_i - y_i\right)^2 + \sum_{i,j} w_{ij} \left(f_i - f_j\right)^2 . \tag{3}$$

The first term in (3) repeats the first term in (2) and the second term in (3) is a dicrete approximation of the second term in (2) according to [14]. Minimization of the energy $E(\mathbf{f})$ can also be interpreted as MAP inference in a real-valued CRF, which are given by the real-valued outputs $f_1, ..., f_N$.

In the matrix form (3) takes the following form:

$$E(\mathbf{f}) = (\mathbf{f} - \mathbf{y})^T C (\mathbf{f} - \mathbf{y}) + \mathbf{f}^T L \mathbf{f}\,, \tag{4}$$

where C denotes a square diagonal matrix with c_i on diagonal and $\mathbf{y}$ denotes an N-dimensional vector of initial likelihood scores y_i. This optimization problem reduces to solving a sparse linear system:

$$(L + C)\mathbf{f} = C\mathbf{y}. \tag{5}$$

The object/background segmentation algorithm then consists in: 1) computing graph Laplacian matrix L; 2) solving the sparse linear system (5); 3) thresholding the output.

Described formulation fits both in interactive segmentation scenario and in semantic image segmentation scenario. In case of interactive segmentation confidence values c_i are infinite for pre-labelled seed points, and 0 for a test points, $y_i = 1$ for seeds marked as object and equals 0 for background seeds. For semantic segmentation we assume that initial estimates y_i and confidences c_i are provided by local models (e.g. appearance model of a specific category).

We notice that instead of pixels we can use image superpixels without making any changes in the algorithm. These superpixels may be represented by overlapping image segments produced by multiple over-segmentations. All possibly overlapping superpixels can be combined in one graph Laplacian, therefore solving a single linear system is required. This allows accounting for similarities between superpixels produced by different over-segmentations. After solving linear system one can get a pixel-wise output by averaging at each pixel real-valued outputs of the method for all superpixels

This framework can be extended to a multi-class segmentation. Let K denote the number of labels corresponding to object categories. If we solve (5) for each label l vs all other labels $1, \cdots, l-1, l+1, \cdots, K$ and obtain the values $y_i^{(l)}$ for all image pixels; at the end, an i-th image pixel is assigned to the label l_{max}, where $l_{max} = arg\max_{l=1,\cdots,K} y_i^{(l)}$.

4 Unsuperivised Learning of Graph Laplacian Parameters

Suppose that the distance function d is represented as a weighted sum of metrics d_i : $\mathbb{R} \times \mathbb{R} \to R^+; i = 1, ..., K$:

$$d(\mathbf{x}_i, \mathbf{x}_j)^2 = \frac{1}{\epsilon} \sum_{k=1}^{K} \alpha_k d_k(\mathbf{x}_i, \mathbf{x}_j)^2\,, \tag{6}$$

with fixed $\alpha_1 = 1$. Therefore the parameters of graph Laplacian $\alpha_i, i = 2, ..., l$ are the weights of features $\mathbf{x}^k, i = 2, ..., l$ and the kernel bandwidth ϵ. Below we show that optimal value of ϵ is determined by the values of $\alpha_i, i = 2, ..., l$.

4.1 Kernel Bandwidth ϵ Selection with Fixed α

We start by fixing the parameters $\alpha_i, i = 2, ..., l$. As shown in [13], if we assume that L provides a good approximation of Laplace-Belrami operator then the following condition holds:

$$\log \sum_{i,j} w_{ij}(\epsilon) \approx m/2 \log(\epsilon) + \log \left(\frac{N^2 (2\pi)^{m/2}}{vol(M)} \right) , \tag{7}$$

where m is a dimensionality of corresponding manifold M and w_{ij} are the elements of the weight matrix W.

Consider the *logarithmic plot* of $\log \sum_{i,j} w_{ij}$ with respect to $\log \epsilon$. Figure 2 shows the plot of $\log \sum_{ij} w_{ij}$ with respect to $\log \epsilon$ and $\log \alpha$ for one image from GrabCut dataset. According to (7) if the approximation is good then the slope of this plot ϵ should be about the half dimensionality of corresponding manifold. In the limit $\epsilon \to \infty$, $w_{ij} \to 1$, so $\sum_{ij} w_{ij} \to N^2$. On the other hand, as $\epsilon \to 0$, $w_{ij} \to \delta_{ij}$, so $\sum_{ij} w_{ij} \to N$. These two limiting values set two asymptotes of the plot and assert that logarithmic plot cannot be linear for all values of ϵ.

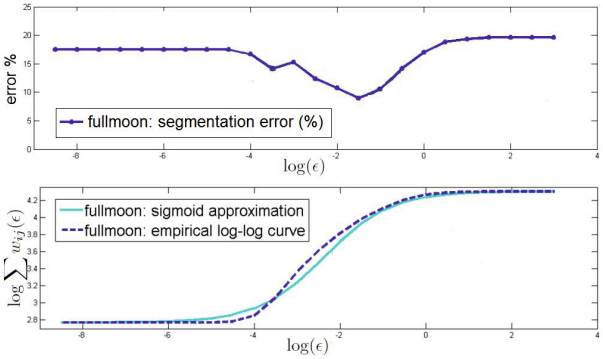


Fig. 2. *Top:* segmentation errors for the "fullmoon" image from GrabCut database with respect to $\log \epsilon$ (α is fixed). *Bottom:* Dashed line - logarithmic plot for the "fullmoon" image with respect to $\log \epsilon$ (α is fixed). The optimal value of ϵ is chosen in the point of maximum derivative of the logarithmic plot; Solid line - sigmoid fit of the logarithmic plot.

Therefore in order to get better approximation of Laplace-Beltrami operator with $\alpha_1, ..., \alpha_K$ fixed we have to choose the value of ϵ from the linear region of logarithmic plot. We use the point of maximum derivative as the point of maximum linearity.

4.2 Implementation Details

We use the distance function from [9]:

$$\tilde{d}^2(\mathbf{x}_i, \mathbf{x}_j) = \frac{\|r_i - r_j\|^2}{\sigma_r^2} + \frac{\|x_i - x_j\|^2}{\sigma_g^2} , \tag{8}$$

where r encodes mean RGB color in the superpixel, x encodes coordinates of the center of the superpixel, parameters of the method $\sigma_r > 0$ and $\sigma_g > 0$ are the scales of chromatic and geometric neighbourhoods respectively.

This distance function (8) can be rewritten in the form of (6) as:

$$\tilde{d}^2(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\epsilon}\left(\|r_i - r_j\|^2 + \alpha \|x_i - x_j\|^2\right), \tag{9}$$

where $\epsilon = 0.5\sigma_r^2$ and $\alpha = \sigma_r^2/\sigma_g^2$. Therefore, the distance function has two parameters ϵ and α.

As follows from (7) the slope of the logarithmic curve near optimal value of ϵ has to be close to $m/2$, where m is the dimensionality of manifold M. In our case $m = 1$, therefore the slope of the logarithmic plot has to be 0.5. If the plot has different slope in the linear region, this indicates that the second term in (7) is large.

So in the first step of our learning method we should find such α that the slope of logarithmic plot of $\log \sum_{ij} w_{ij}(\epsilon)$ from ϵ is equal to 0.5. In the second step we use the sigmoid fit of the logarithmic plot. The shape of logarithmic plot can be approximated with a sigmoid function: $T(x) = A/(B + \exp(Cx + D)) + E$. Since the asymptotes of the sigmoid are set by (7) and the slope in the linear region of the sigmoid should be 0.5 the sigmoid has only one free parameter that controls the shift of the sigmoid along horizontal axis. Figure 2 illustrates the choice of ϵ according to sigmoid approximation. In our experiments the values of ϵ take values as degrees of 10 and the values of α take values as degrees of 2.

We found empirically that usually the slope of the logarithmic plot is greater than 0.5 for large α and is less than 0.5 for small α. In most cases the slope of the logarithmic plot $S(\alpha)$ is monotonic function of α. One of the possible explanations of this fact can be the following. Small α correspond to using only spatial information. This implies that the dimension of manifold where the data lives is 2 and it is difficult to reduce dimensionality further. By decreasing α we decrease the weight of spatial information in the distance function therefore it gets easier to find the corresponding one-dimensional manifold. On the other hand large α corresponds to increased weight of color information. Infinite α corresponds to using color information alone. As long as the color space is three-dimensional and the color distribution of object and background is complex it is difficult to embed one-dimensional manifold in the input points. Example of the logarithmic plot with respect to both ϵ and α is shown in Fig. 3.

5 Experiments

For the experiments we used GrabCut, Graz and Pascal 2007 datasets. In all experiments graph Laplacian operated with superpixels produced by image over-segmentation methods. Each superpixel was linked with a fixed number of it's nearest neighbours, and the distances to other superpixels were assumed infinite. For all experiments we used confidences that are a linear function of the outputs of local appearance models $c_i = 0.5(1 - |p_i - 0.5|)$.

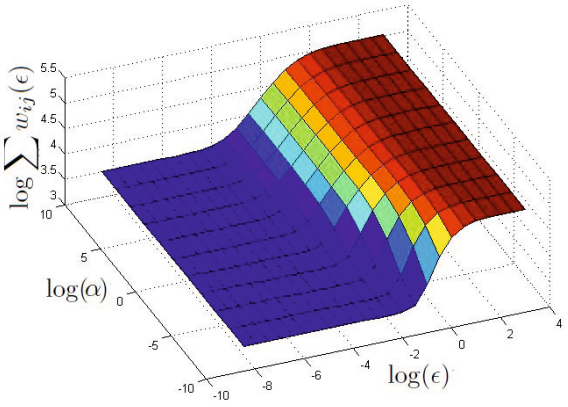


Fig. 3. The plot of $\log \sum_{ij} w_{ij}$ with respect to $\log \epsilon$ and $\log \alpha$. The plot shown is shown in (2, bottom) corresponds to the 2-d slice of this 3-d plot for fixed α. Note that the slope of linear region are not constant for all values of α. We seek for α such that the slope in the linear region equals 0.5.

5.1 GrabCut Image Dataset

GrabCut image dataset contains 50 images of different objects [1]. Each image has a binary mask which indicates pixels belonging to the object. In the experiments we used the set of superpixels which is the union of oversergmentations provided by Colour Structure Code and Watershed segmentation methods.

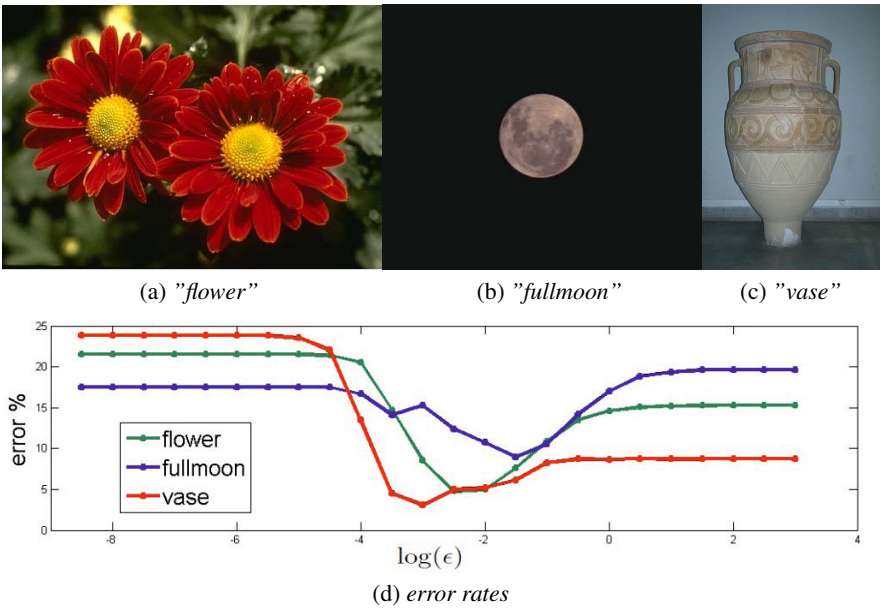
GrabCut image database considers interactive image segmentation scenario, when we have user-provided seeds for objects on image. Initial labellingfor each pixel is provided, every pixel is labelled asobject, background or unknown. We used the following strategy to get the seed labels for superpixels. The superpixel was marked as object seed if it contains pixels marked as object and it doesnt contain any background pixels. Similarly the superpixel was marked as background if any of the pixels inside is marked as background and no pixels are marked as object. Otherwise the superpixel is marked as unknown and the labels for all its pixels are inferred by graph Laplacian method.

Figure 4 (d) shows the error on 3 different images from GrabCut database with respect to $\log \epsilon$ (α is fixed). Depending on the choice of ϵ one can get different values of errors and for each image, and the optimal values of ϵ are different for different images.

In all the experiments we report the errors obtained on the whole GrabCut image database. We did not split the dataset into training/ validation/test sets.

We measured performance on GrabCut dataset according to standard metric [1]. In the first experiment we compared results of using single over-segmentation to using multiple over-segmentations. Parameters α and ϵ of the method were validated on the whole image database and thus correspond to the peak performance of the framework with constant parameters α and ϵ. The results are presented in Table 1. Using two different over-segmentations leads to significant improvement of performance compared

[1] available at `http://research.microsoft.com/~en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm`



(a) *"flower"* (b) *"fullmoon"* (c) *"vase"*

(d) *error rates*

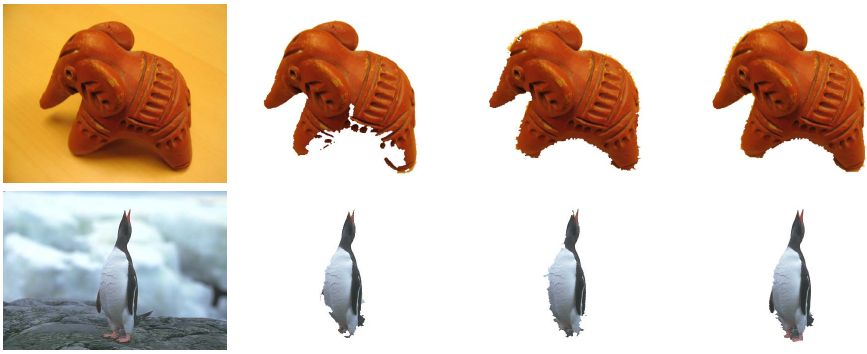
Fig. 4. Segmentation errors depending on graph Laplacian parameter ϵ on three images from GrabCut dataset. Note that minimal error is achieved on different values of ϵ for different images.

to using only one of the over-segmentations, either Watershed or CSC. The results of segmented images using different oversegmentations are shown on Fig. 5.

Table 1. Errors on GrabCut dataset. Watershed - using only watershed over-segmentation, parameters of graph Laplacian framework chosen on the whole GrabCut image database; CSC - using only Color Structure Code over-segmentation, parameters chosen on the whole GrabCut image database; Watershed+CSC1 - using both over-segmentations with parameters validated on the same GrabCut dataset; Watershed+CSC2 - using both over-segmentations, parameters chosen by proposed self-tuning method individually for each image. Note that self-tuning shows almost identical performance as the peak performance with parameters validated on the same dataset, and the standard deviation of errors of self-tuned graph Laplacian is even smaller than of Laplacian with fixed parameters.

	Error
Watershed	$13.61 \pm 5.14\%$
CSC	$11.02 \pm 7.59\%$
Watershed+CSC1	$9.86 \pm 4.31\%$
Watershed+CSC2	$10.25 \pm 4.01\%$

In our second experiment we compared three versions of graph Laplacian with multiple oversegmentations. First, we chose single set of parameters for the whole dataset by validation on the test dataset from the previous experiment. This corresponds to upper bound on performance of the method with fixed parameters. The resulting error rate



(a) (b) (c) (d)

Fig. 5. Comparison of using multiple over-segmentations to using one over-segmentation. (a) - input image; (b) - result of using CSC over-segmentation; (c) - result of using watershed over-segmentation; (d) - result of using both over-segmentations as superpixels.

is $9.9 \pm 4.3\%$. Then we evaluated the performance of graph Laplacian with the parameters learnt individually for each image by our method. The resulting error rate is $10.2 \pm 4.0\%$. Finally, we chose the best parameters for each image individually by validation on the same image in order to obtain the top bound on performance of graph Laplacian. The resulting error rate is 8.7%. The results of graph Laplacian with learnt parameters is very close to the upper bound of performance of graph Laplacian with fixed parameters. Notably, the standard deviation of errors obtained with the parameters learnt individually for each image is smaller than of Laplacian with fixed parameters. The results are presented in Table 1.

The learning phase took from 0.5 seconds to 3 seconds, solving linear system 5 took from 0.05 second to 0.5 seconds depending on the number of superpixels in the image (the total number of superpixels varied from 500 to 30000).

5.2 Graz Image Dataset

Graz dataset [2] contains 1096 images of three classes: "person", "bike" and "car". In our experiments we solved a separate binary segmentation problem for each category. To measure the quality of segmentation we used a standard metric - percent of incorrectly classified pixels in the image.

In our experiments we used an open-source VlBlocks toolbox [3], which implements the method described in [15]. We chose it for comparison for the following reasons. First, it allows using different local appearance models. The method has a parameter N meaning number of neighbouring superpixels which features are used for classification of each particular superpixel. So we report performance metrics for different values of N to illustrate the performance of proposed graph Laplacian framework applied to different local models. Second, the toolbox includes implementation of discrete CRF with

[2] available at `http://www.emt.tugraz.at`

[3] code available at `http://vlblocks.org/index.html`

graph-cut inference, which we use for comparison. Note, this CRF model uses similar types of features (color and spatial coordinates of superpixels) to those used in our graph Laplacian. We used the same over-segmentation and the same local appearance model based on SVM as [15]. To obtain initial estimates y_i for graph Laplacian framework we scaled SVM outputs to $[0, 1]$ interval for each image.

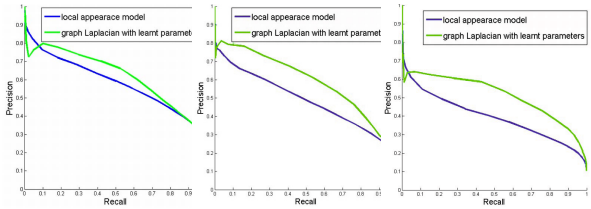
In the first experiment the parameters ϵ and α were validated on the GrabCut dataset. In the second experiment we validated the parameters on the test set. In the third experiment we used our unsupervised learning method for choosing the parameters individually for each image. We also compared with Vlblocks implementation of CRF with graph-cut inference. The strategy for choosing internal parameters of CRF was the same as in [15].

Table 2 contains results of the comparison. Our unsupervised learning gives results comparable to upper bound on performance of graph Laplacian with fixed parameters from the second experiment. The value of performance gain compared to local appearance model differs for different values of parameter N. The smaller N is the smaller neighborhood is considered by low-level model, and the more significant is the gain in performance attained by both CRF and graph Laplacian. The gain in performance of graph Laplacian is almost uniformly higher than the performance gain obtained by discrete CRF. Note that the gain achieved by graph Laplacian is several times higher than the one achieved by discrete CRF for $N = 0$. Figure 6 shows precision-recall curves for local appearance models and for our method to illustrate the gain in performance due to graph Laplacian.

Table 2. Performance on Graz dataset at equal precision and recall rates for "cars", "bike" and "person" classes. First row: local appearance model (from VlBlocks toolbox). Second row: result of applying discrete CRF with graph cut inference (from VlBlocks toolbox). Third row: graph Laplacian with parameters validated on GrabCut dataset. Fourth row: graph Laplacian with parameters validated on the test set. Fifth row: graph Laplacian with parameters learnt individually for each image. For each appearance model used in our experiments (we varied the number of neighboring regions as in [15]) the best result is shown in **bold font**. Underlined are the best overall results.

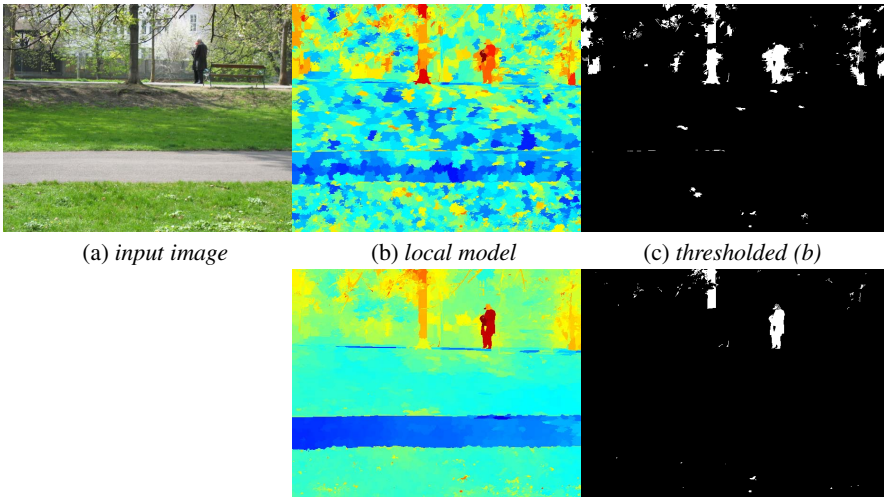
	N=0			N=1			N=2			N=3			N=4		
	cars	bike	pers	cars	bike	pers	cars	bike	pers	cars	bike	pers	cars	bike	pers
SVM	41.9	56.5	49.4	59.6	66.9	63.6	68.0	69.2	66.6	69.4	70.7	65.2	66.5	71.9	63.6
+CRF	43.0	57.7	49.3	60.2	67.1	63.9	70.1	70.2	66.9	70.7	71.0	65.4	68.8	72.2	64.2
+Laplacian (valid.GrabCut)	50.0	60.1	56.0	65.5	68.7	68.5	71.6	70.8	70.8	72.2	72.0	69.5	70.0	**73.2**	67.3
+Laplacian (valid.test set)	**56.6**	**63.3**	**59.1**	66.3	68.4	68.8	71.9	70.4	70.4	72.6	71.2	69.4	70.8	72.2	68.0
+Laplacian (learnt)	54.2	60.9	58.5	65.1	66.8	**69.4**	**72.0**	69.5	**71.3**	**73.3**	70.3	**70.2**	**71.4**	71.5	**68.9**

Figure 7 shows results provided by local appearance model (SVM) and corresponding results of using graph Laplacian with learnt parameters. Figure 8 shows how the results vary for different local models.



(a) *"bike" class* (b) *"person" class* (c) *"car" class*

Fig. 6. Precision-recall curves for "bike", "person" and "car" classes of Graz dataset. *Blue curves* - local appearance model (N=0); *Green curves* - graph Laplacian with learnt parameters.



(a) *input image* (b) *local model* (c) *thresholded (b)*

d *Laplacian* (e) *thresholded (d)*

Fig. 7. Results of SVM and graph Laplacian method for images from Graz dataset. (a) - input image "person" class; (b) - real-valued output from local SVM model; (c) - results of thresholding the SVM outputs; (d) - real-valued output of graph Laplacian using SVM as a local model with the parameters learnt by our method; (e) - thresholded output of our method. Note how graph Laplacian refines the output from SVM. It doesn't oversmooth the result and preserves fine details and the small figure of the person.

The running time on Graz dataset is the following: learning phase takes about 0.2 seconds on average, solving of linear system 5 takes about 0.02 seconds on average.

5.3 Pascal 2007 Image Dataset

Pascal 2007 dataset [4] contains 21 classes. Again in this experiment we use local models from VlBlocks toolbox trained with parameters as in [15]. We compare our graph

[4] available at `http://pascallin.ecs.soton.ac.uk/~challenges/VOC/voc2007/`

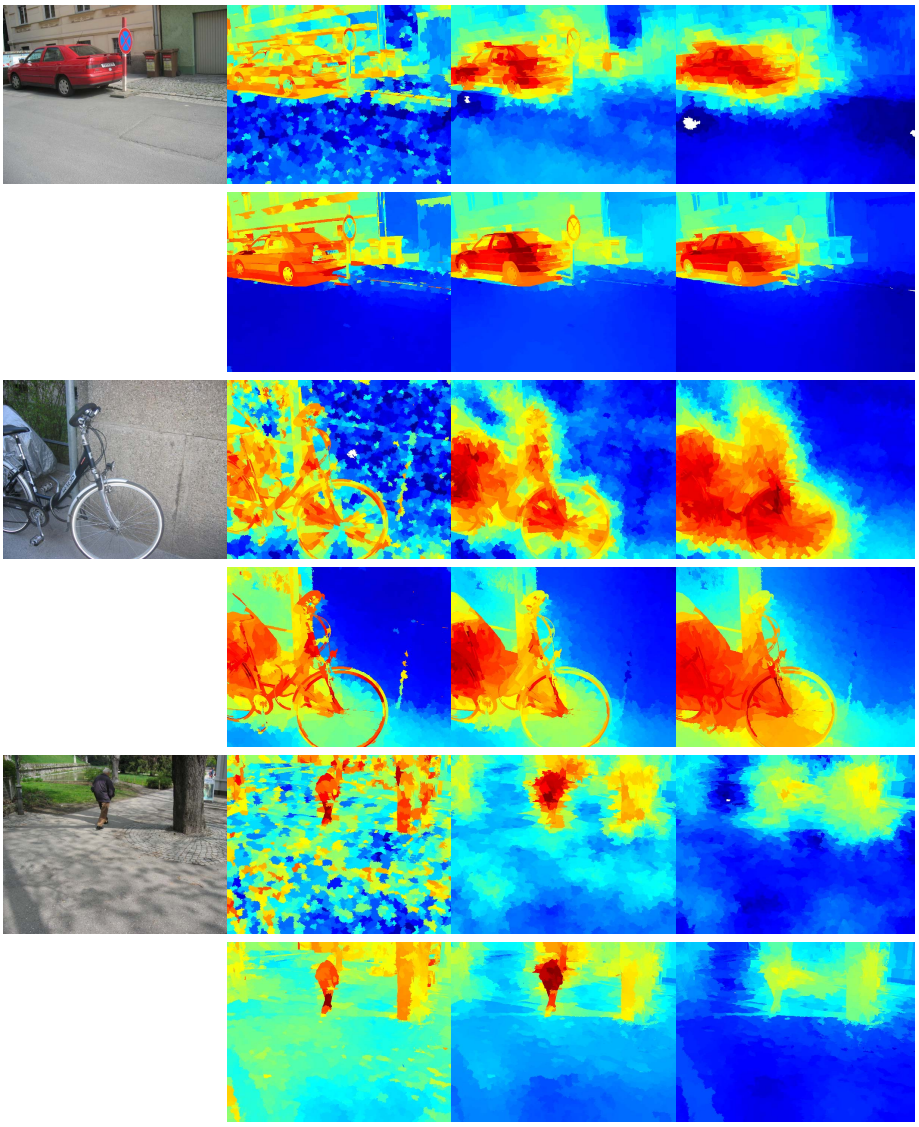


Fig. 8. Results on Graz dataset. Odd rows show real-valued outputs of SVM, even rows show results of Graph Laplacian framework with self-tuning for different number of neighbouring superpixels used for feature computation: (b)N=0; (c)N=2; (d)N=4. Note that while looking very reasonable the result of presented framework for bike image is a failure case according to ground truth labelling in Graz since inside part of the wheel should be classified as ”bike”. This is a typical error of presented framework on ”bike” class which leads to higher error rate.

Laplacian method with unsupervised learning to the discrete CRF implemented in Vl-Blocks toolbox. The training and testing split is defined in the challenge. We train local model on the train set and choose the parameters of discrete CRF on the validation set. We do not use the validation set in the experiment with our graph Laplacian method and use our unsupervised learning method for choosing Laplacian parameters for each particular image.

Table 3 shows the results of comparison to using local model alone, discrete CRF and our graph Laplacian method with unsupervised learning. For comparison we also reproduce results from [16], [17] and [15]. On this dataset, adding graph Laplacian improves the results significantly compared to using local model alone, and provides a consistent boost for the accuracy as well.

Table 3. Results on Pascal 2007 dataset. Best result for each category is shown in bold.

	background	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	dinningtable	dog	horse	motorbike	person	pottedplant	sheep	sofa	train	tv monitor	Average	Pixels %
Pantofaru et al. [17]	59	27	1	8	2	1	32	14	14	4	8	32	9	24	15	81	11	**26**	1	28	17	20	-
Shotton et al. [16]	33	46	5	14	11	14	34	8	6	3	10	39	40	28	23	32	19	19	8	24	9	20	-
Fulkerson et al. [15]	56	26	**29**	**19**	16	3	**42**	44	**56**	23	6	11	62	16	68	46	16	10	21	52	**40**	**32**	51
local model (N=0)	16	17	8	9	13	8	9	9	29	15	9	12	12	7	13	5	16	16	26	14	21	14	15
+discrete CRF (N=0)	17	18	8	9	**21**	8	9	8	28	14	9	13	12	7	13	5	16	16	26	14	22	14	16
+Ours(learnt) (N=0)	23	20	26	12	18	12	17	14	49	17	1	23	16	11	50	4	**42**	20	**44**	30	33	23	22
local model (N=1)	24	13	18	13	9	12	14	24	35	16	9	11	28	15	36	23	14	21	20	35	27	21	38
+discrete CRF (N=1)	42	6	16	9	6	5	11	14	56	19	4	11	16	16	55	36	24	16	8	56	21	21	38
+Ours(learnt) (N=1)	33	11	18	14	10	13	16	26	51	16	7	9	35	**22**	67	29	31	25	16	**60**	33	26	32
local model (N=2)	39	10	22	15	11	12	18	36	44	23	8	11	33	15	53	37	17	17	16	36	28	24	36
+discrete CRF (N=2)	58	9	25	14	6	3	20	38	54	**27**	12	10	31	7	59	44	12	17	13	43	27	25	50
+Ours(learnt) (N=2)	52	9	24	17	8	12	19	41	55	21	10	10	**37**	14	68	42	14	12	14	51	32	27	46
local model (N=3)	52	13	14	18	8	5	23	38	45	17	7	10	30	21	63	50	17	20	19	43	23	25	46
+discrete CRF (N=3)	65	10	14	15	5	2	24	40	60	13	6	8	24	19	68	55	18	19	16	46	26	26	56
+Ours(learnt) (N=3)	63	11	15	17	7	2	22	40	60	13	5	10	32	20	**71**	57	13	17	14	48	26	27	55
local model (N=4)	59	6	15	18	4	0	25	44	46	17	3	4	24	20	62	56	15	14	13	38	33	25	51
+discrete CRF (N=4)	63	8	15	18	4	0	26	**46**	48	17	2	4	25	20	64	58	12	14	12	38	34	25	54
+Ours(learnt) (N=4)	69	5	11	**19**	4	0	28	45	55	13	2	3	18	18	69	**60**	7	13	7	44	36	25	**59**

6 Conclusion

We presented an semantic segmentation framework based on graph Laplacian approximation of manifold embedding problem. The main contribution of this work is a method for choosing internal parameters of graph Laplacian in a fully unsupervised manner individually for each test image. Proposed unsupervised learning method has a low computational cost and shows better performance compared to discrete CRF with graph-cut inference.

References

1. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In: ICCV, vol. 1, pp. 105–112 (2001)
2. Kohli, P., Ladicky, L., Torr, P.: Robust higher order potentials for enforcing label consistency. In: CVPR (2008)
3. Ladicky, L., Russell, C., Kohli, P., Torr, P.H.S.: Graph cut based inference with co-occurrence statistics. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 239–253. Springer, Heidelberg (2010)
4. Szummer, M., Kohli, P., Hoiem, D.: Learning CRFs using graph cuts. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 582–595. Springer, Heidelberg (2008)
5. Nowozin, S., Gehler, P.V., Lampert, C.H.: On parameter learning in CRF-based approaches to object class image segmentation. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part VI. LNCS, vol. 6316, pp. 98–111. Springer, Heidelberg (2010)
6. Grady, L.: Random walks for image segmentation. IEEE Trans. on Pattern Analysis and Machine Intelligence 28(11), 1768–1783 (2006)
7. Levin, A., Lischinski, D., Weiss, Y.: A closed form solution to natural image matting. IEEE Trans. on Pattern Analysis and Machine Intelligence (2008)
8. Singaraju, D., Grady, L., Vidal, R.: P-brush: Continuous valued mrfs with normed pairwise distributions for image segmentation. In: CVPR (2009)
9. Duchenne, O., Audibert, J.Y., Keriven, R., Ponce, J., Segonne, F.: Segmentation by transduction. In: CVPR (2008)
10. Hein, M., Audibert, J.Y., von Luxburg, U.: From graphs to manifolds - weak and strong pointwise consistency of graph laplacians. ArXiv Preprint, Journal of Machine Learning Research (2006) (forthcoming)
11. Zhou, H., Cheng, Q.: O(n) implicit subspace embedding for unsupervised multi-scale image segmentation, pp. 2209–2215 (2011)
12. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computataion 15, 1373–1396 (2003)
13. Coifman, R.R., Shkolnisky, Y., Sigworth, F.J., Singer, A.: Graph laplacian tomography from unknown random projections. IEEE Trans. on Image Processing
14. Belkin, M., Niyogi, P.: Towards a theoretical foundation for laplacian-based manifold methods. In: Auer, P., Meir, R. (eds.) COLT 2005. LNCS (LNAI), vol. 3559, pp. 486–500. Springer, Heidelberg (2005)
15. Fulkerson, B., Vedaldi, A., Soatto, S.: Class segmentation and object localization with superpixel neighborhoods. In: ICCV (2009)
16. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image categorization and segmentation (2008)
17. Pantofaru, C., Schmid, C., Hebert, M.: Object recognition by integrating multiple image segmentations. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 481–494. Springer, Heidelberg (2008)

Virtual Reality Technology for the Visual Perception Study

Galina Menshikova, Yuriy Bayakovski,
Elizaveta Luniakova, Maxim Pestun, and Denis Zakharkin

Moscow State University, Moscow, Russia
{gmenshikova,eluniakova,max.pestun,denis.zakharkin}@gmail.com,
ymb-lab@yandex.ru
http://www.msu.ru/en/

Abstract. Lately the virtual reality (VR) techniques were applied successfully to investigate 3D visual perception. In our study the effects of 2D vs. 3D displays on lightness perception was assessed using the CAVE system. In current models of lightness perception it has been suggested that 2D visual cues in a scene play a crucial role in lightness estimations. In some studies the role of depth cues was investigated, but the results were contradictory. In our study the effect of 2D vs. 3D displays of the simultaneous lightness contrast (SLC) illusion on its strength was investigated. Namely the articulation effect was studied for 2D vs. 3D displays of the SLC illusion. Three modified configurations of 2D SLC articulated displays were constructed having a) depth cues for test squares and b) depth cues for background squares. For all configurations test squares were equally moved out of their backgrounds. The background squares consisted of different objects: 2D patches for the first configuration, 3D cubes or 3D balls for the other configurations. The number of objects in any configuration remained constant. Twenty five observers were tested. They were asked to estimate the illusion strength for 2D and three 3D versions of the SLC illusion. The method of constant stimuli was used. The results showed that the illusion strength decreased for all 3D displays relative to the 2D displays of the SLC illusion. There were no significant differences in illusion strength between three modified versions of the SLC displays with 3D backgrounds. The results allowed introducing the modified articulation rule for 3D complex scenes.

Keywords: 3D visual illusions, lightness perception, simultaneous lightness contrast Virtual Reality, VR, CAVE.

1 Introduction

A new Virtual Reality (VR) technology was widely used in psychological research last decade. Its effectiveness has been proven by medicine, neuropsychology, cognitive and social psychology data. The VR technology equips experimental psychology with methods that have certain differences from traditional laboratory instruments. A heated dispute of the advantages and disadvantages of using VR

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XIX, LNCS 7870, pp. 107–116, 2013.

systems in psychology has been held in all experimental and review works carried out within this new technique [1][2][3]. As for the studies in visual perception VR technology provides 1) active 3D viewing and 2) complex 3D scenes with controlled parameters.

In our study VR technique was applied to investigate lightness perception. The problem of lightness perception is tightly connected with the perception of lightness illusions, which were often used as demonstrations of theoretical assumptions made within different approaches to lightness perception. Recently the anchoring theory of lightness perception was frequently debated [4]. It assumed that the ratios of the test surface luminance to the luminance of other surfaces determined the process of lightness estimations. Using these ratios visual system could estimate the relative reflectance of all surfaces, which were equally illuminated. To estimate the absolute surface reflectance the anchoring rule should be applied: one of relative reflectance values anchored to some absolute value, for example, to the most luminous object in the scene which supposed to be white [4][5]. The anchoring rules "worked" in the range of local and global frameworks. Local frameworks were used to estimate the luminance ratio of the test and adjacent background surfaces while global frameworks were used for estimating the luminance ratio of test and distant surfaces. The net lightness of the test surface was a weighted average of its computed luminance ratio values in each of these frameworks, in proportion to the strength of each framework. The strength of the framework was determined by its angular size and its articulation. The effect of articulation was defined as the number of patches with different reflectance within a scene. These theoretical hypotheses were used to explain some lightness illusions, for example the simultaneous lightness contrast (SLC) illusion [4][6] (Fig. 1). Two identically gray target squares, located on the light or dark backgrounds, were perceived as dark-gray and light-gray respectively.

It was found that local frameworks played a crucial role in the SLC illusion. Two local frameworks could be picked out in the SLC display: 1) the grey target

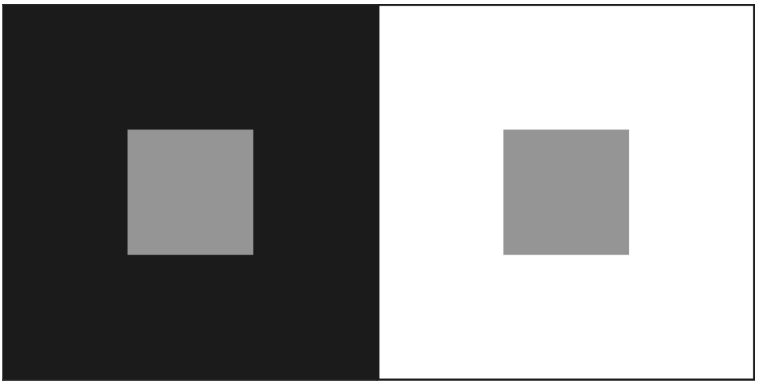


Fig. 1. The classic display of the SLC illusion

in the white surround and 2) the grey target in the black surround. The SLC display as a hole could be considered as a global framework. The anchoring in the first local framework and in the global framework should lead to the same correct lightness estimations because of the same anchor – a white surface. The anchoring in the second local framework should yiel a higher value for target's lightness because the target itself became the local anchor.

The anchoring theory supposed to explain lightness perception in complex scenes, where lightness estimations were accomplished into two stages. At the first stage the groups of coplanar surfaces (= equally illuminated) were picked out, and at the second stage lightness was estimated in accordance with the anchoring theory for each group separately.

In complex scenes depth and articulation cues would result in lightness assessment. The role of depth cues was tested in a number of works, but the results were contradictory. The main idea of these studies was to manipulate 3D positions of test and background surfaces. In accordance with the anchoring theory it would lead to decreasing the relationship between test and background surfaces, and as a result to a shift in lightness assessment. Some works [7][8] confirmed these predictions. Other studies did not reveal or determined the very weak influence of depth cues on lightness estimations [9][10]. So, the question of the role of depth cues remained unclear. The influence of the articulation cues on lightness perception was proposed and investigated by D. Katz [11]. Articulation effects were determined as the influence of the background complexity on lightness estimations. The term "complexity" referred to the number of patches with different reflectance within a whole scene. The rule of articulation was formulated by D. Katz [11] as following: the more colored patches were located around the test patch the better lightness estimations were. For example, it was shown that lightness estimations were much more accurate if the background was not a uniform-colored surface but a surface consisted of 48 patches of different reflection changing from black to white [12]. The influence of articulation effect also was shown in the famous Gelb effect [13], where illuminated black disk on the dark unlit background was demonstrated. Observers estimated the disk color as white or light-grey. Color estimations became more accurate if another small white disk was added in the field of illumination. Similar results were obtained in a recent study [4]. It was shown that the lightness of the test disk was estimated as 7.5 (light-grey), 4.5 (middle-grey) or 3.3 (very dark grey) Munsell units if the test disk was surrounded by 2, 5 or 10 patches having different grey shades.

According to articulation rule proposed by D. Katz the strength of the SLC illusion in articulated version (Fig. 2) would decrease because of higher accuracy of lightness estimations. However, some studies have revealed the opposite results: the illusion strength increased for the articulated version [4]. To explain this result the rule of articulation was modified [14]: the higher degree of articulation within a framework would result in stronger local anchoring. According to this modified rule the strength of the SLC illusion should increase due to stronger local anchoring, which played a main role in the SLC illusion.

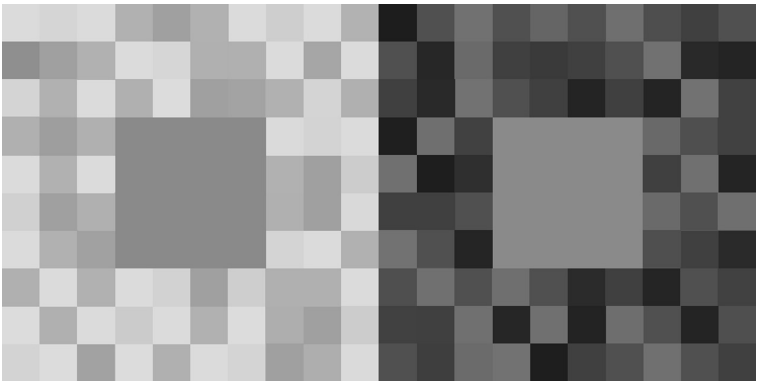


Fig. 2. The articulated version of the SLC illusion

The articulation effect was mainly investigated for 2D scenes. How can be formulated the articulation rule for scenes composed of 3D objects? The distinct patch in 2D scenes could be displayed by the single luminance value, while the 3D object of uniform reflectance should be displayed by several luminance values depending on its shape. So, in 2D displays the number of patches having different reflectance was the same as the number of patches of different luminance, but in 3D scenes the number of 3D objects with different reflectance would be obviously less than the number of patches of different luminance. The question arises which of mentioned parameters – reflectance or luminance – determines the articulation effect.

Two answers to this question may be proposed. The articulation effect could be determined by 1) the number of patches which reflectance imaged by the single luminance value, or 2) the number of 3D uniform-colored objects which reflectance imaged by several luminance values depending on its shape. We named the first type of articulation as a "luminance articulatio" and the second type – as an "an object articulation".

In our study we investigated which type of articulation rules would determine lightness perception in complex 3D scenes.

Using the VR technology, we studied the strength of 3D SLC illusion to find out the role of 1) the depth cues and 2) articulation cues of 3D backgrounds on the strength of 3D lightness illusions.

Two hypotheses were offered:

1. Locating the test and background surfaces in different space positions would break their local relationship resulting in weakening local anchoring and, in their turn, in reducing the illusion strength. So, the strength of 3D SLC illusions would decrease relative to its classical 2D configuration.
2. If the articulation effect is determined by the number of patches with reflectance imaged by the single luminance value then the illusion strength would increase under 2D to 3D backgrounds transformation. On the contrary, if the articulation effect is determined by the number of objects with

reflectance imaged by several luminance values depending on its shape the illusion strength would not change.

2 Method

2.1 Observers

Twenty five observers (age range 17-30) with normal or corrected to normal vision were tested. Before the experiment all observers were tested for 3D viewing ability. They were unaware of experiment's purpose.

2.2 Stimuli

The 2D articulated version of the SLC illusion (Fig. 3.1) was used as a basic display. Three

modified 3D configurations of a basic display were constructed having a) depth cues for test squares and b) depth cues for background squares. For all configurations test squares were equally moved out of their backgrounds. The background squares consisted of different objects: 2D articulated patches for the first configuration (Fig. 3.2), 3D cubes for the second (Fig. 3.3) and 3D balls for the third (Fig. 3.4) configurations. 3D backgrounds varied from simple (the first type) to complex (the third type) variant of the articulation effect. The number of background objects in any configuration remained constant.

The average luminance of backgrounds was constant for all types of stimuli.

The method of constant stimuli was used to estimate the strength of the SLC illusion. The gray squares on the light backgrounds were standard. Its lightness was 30% of white shade in Grayscale units and was not changed during the experiment. Seven variable stimuli were created for every 2D-3D configuration, for which the value of lightness for the test squares lying on the dark backgrounds decreased from 30% to 15% of white shade with a step of 2.5%. So, 28 stimuli were created: four 2D-3D configurations, each having seven variable lightness values stimuli of the test square.

2.3 Apparatus

The 2D articulated version of the SLC illusion and three types of 3D displays were presented using the CAVE system (Fig. 4).

The CAVE system had four large flat screens (Barco ISpace 4), which were connected into one cube consisting of three walls and a floor. The length of each screen side was about 2.5 meters. Shutter eye glasses were made by CrystalEyes 3 Stereographics. Projection system was based on BarcoReality 909. The projector's matrix resolution was 1400x1050 with 100 Hz update frequency. Tracking system produced by ArtTrack2. VirTools 4.0 was used for software developing. It supported DX9/GL2, HAVOK, particle systems and shaders.

The observer stood motionless in front of the central screen at a distance of 2.5 m. Virtual stimulus configuration was located before him with the background

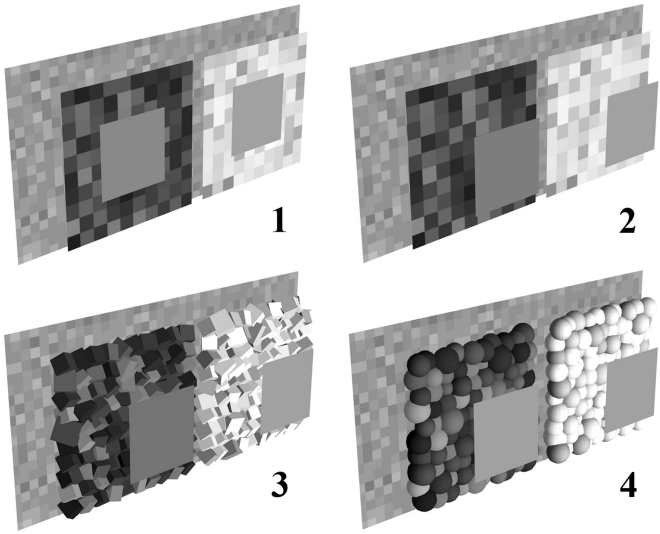


Fig. 3. Different types of 2D-3D configurations of the SLC illusion: 1 – 2D classic articulated configuration; 2 – 3D configuration with 2D articulated background; 3 – 3D configuration with 3D articulated background (cubes); 4 – 3D configuration with 3D articulated background (balls)

placed on the screen plane. It subtended 30 of visual angle horizontally and 15 vertically.

The visual angle of the test squares in 3D configurations (2, 3 and 4) was the same as those in the 2D display (1). Thus, when projected on the retina, 3D and 2D displays produced practically the same pattern.

The laboratory room was darkened; there were no any light sources, except CAVE systems projectors. The luminance range in stimulus scene was 1:230. The maximum luminance was $5.5cd/m^2$, the minimum - $0.02cd/m^2$.

2.4 Procedure

The observer was given the following instructions: "Each trial you will see two gray test squares on the different backgrounds. Please, choose the lighter of two central squares, using a special joystick. Try to stand motionless during the experiment."

The experiment included four series: 1 – 2D articulated version of the SLC illusion; 2 – 3D articulated version of the SLC illusion with backgrounds consisted of 2D patches; 3 – 3D articulated version of the SLC illusion with backgrounds consisted of cubes; 4 – 3D articulated version of the SLC illusion with backgrounds consisted of balls.

Each series lasted about 5 minutes. The stimuli sequence was completely randomized. Every series consisted of 70 trials: each of seven variable stimuli was repeated 10 times. The left/right position of light and dark backgrounds was changed randomly.

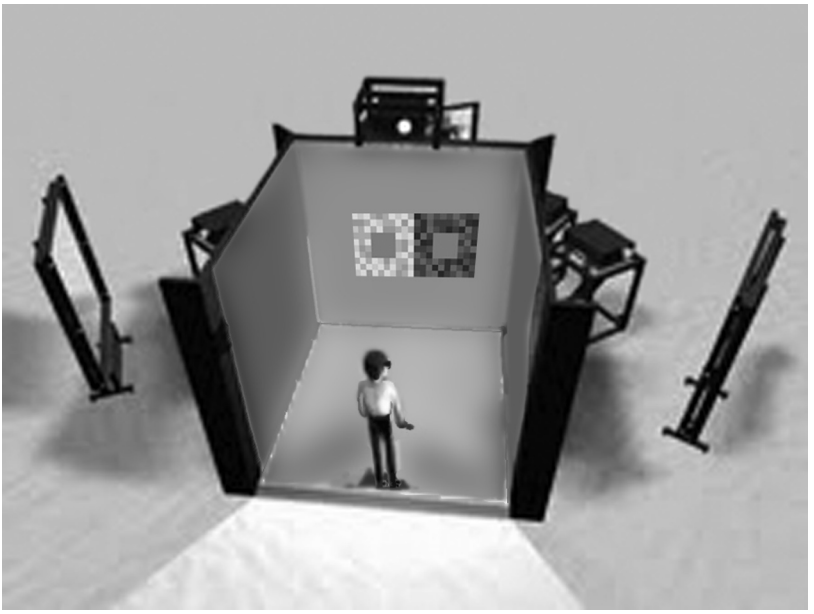


Fig. 4. The CAVE system

3 Results

Psychometric functions for 2D and three different 3D configurations were obtained and used to evaluate the strength of the SLC illusion for each participant and each 2D-3D configuration. The illusion strength was calculated as $IS = (LSt - LT)/LSt * 100\%$, where LSt was luminance of standard square; LT – PSE (Point of Subject Equality) – luminance of test square with 50% probability of answers "lighte".

Statistical processing of the data included the Kolmogorov-Smirnov test testing for normality of the received data distribution and the paired Student's t-test.

The results averaged across 25 observers are shown in Fig. 5. The horizontal axis plots the different 2D-3D configurations. The vertical axis plots the average strength of the SLC illusion (%).

The significant differences were revealed between the type 1 (2D articulated configuration) and the other different 3D configurations (the type 2 ($t(24) = 9.9, p < 0.001$), the type 3 ($t(24) = 5.4, p < 0.001$), the type 4 ($t(24) = 8.01$, $p < 0.001$).

The strength of 2D classic display was twice more than the strength of any 3D display of the SLC illusion. As to articulation effects, there were no significant differences between the values of SLC strength calculated for three types of 3D backgrounds: for 2 and 3 types ($t(24) = 1.88, p = 0.05$), 2 and 4 types ($t(24) = 0.29, p = 0.05$), 2 and 3 types ($t(24) = 2.22, p = 0.01$).

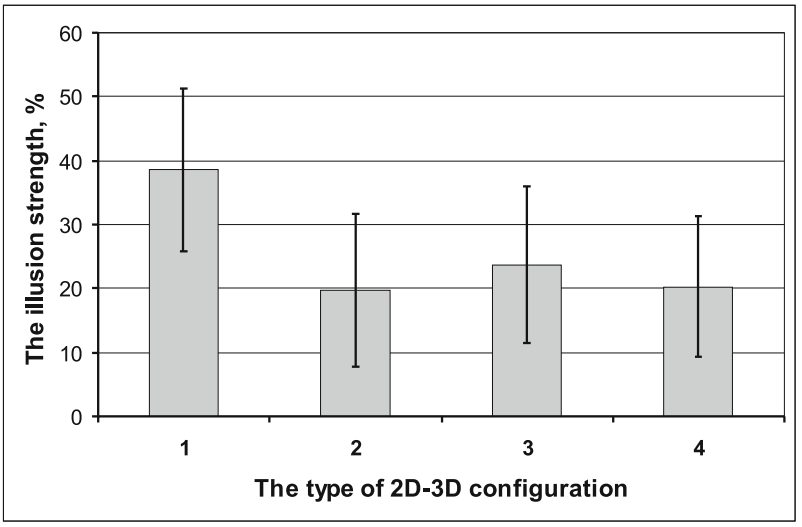


Fig. 5. The strength of the SLC illusion for 4 types of stimulus configurations: 1 – 2D articulated configuration; 2 – 3D configuration with backgrounds consisted of 2D patches; 3 – 3D configuration with backgrounds consisted of cubes; 4 – 3D configuration with backgrounds consisted of balls

4 Conclusion

Our results revealed strong influence of depth cues on the illusion strength. It was shown that the illusion strength decreased for all 3D displays of the SLC illusion relative to 2D articulated classic display. This result was in good agreement with the anchoring theory. Different depth positions of the test and background surfaces resulted in weakening anchoring within the local framework, that should lead to more correct targets lightness estimation and, in turn, to the reduction of the illusion strength. The same results have been reported in our study with 3D configurations of classic SLC illusions [15]. So, our first hypothesis was successfully confirmed. It should be noted that the significant changes of the illusion strength cannot be explained in the framework of other approaches to lightness perception.

There were no significant differences in illusion strength for different types of 3D backgrounds. It seems that articulation effects weakly depend on depth cues of backgrounds. Tree types of the backgrounds differed by the number of patches with different luminance; while the number of uniform-colored objects remained constant within a scene. Our results showed that lightness estimations were defined only by the number of uniform-colored objects. So, our second hypothesis which asserted that in 3D scenes lightness should be estimated in accordance with an “object articulation” rule was also confirmed.

Virtual reality technologies may be effectively used in studies of lightness perception and 3D visual illusions. It enables to reproduce illusions in depth and to construct complex 3D scenes with controlled parameters to create 3D illusory effects.

Acknowledgments. The work is supported by the Program of Development of Lomonosov MSU and by the grant "The application of modern information technologies to the development of innovative methods in the study of human cognitive processes" within the framework of the Federal Target Program "Scientific and scientific-pedagogical personnel of innovative Russia" for 2009-2013 (Contract №8011).

References

1. Khan, Y., Xu, Z., Stigant, M.: Virtual Reality for Neuropsychological Diagnosis and Rehabilitation: A Survey. In: Proceedings of the Seventh International Conference on Information Visualization, pp. 158–163. IEEE Computer Society, Washington DC (2003)
2. Yee, N.: Psychological Research in Virtual Worlds, http://bps-researchdigest.blogspot.com/2007/06/psychological-research-in-virtual.html
3. Zinchenko, Y.P., Menshikova, G.Y., Bayakovsky, Y.M., Chernorizov, A.M., Voiskounsky, A.E.: Technologies of virtual reality in the context of World-wide and Russian psychology: methodology, comparison with traditional methods, achievements and perspectives. In: Zinchenko, Y.P., Petrenko, V.F. (eds.) Psychology in Russia. State of the Art. Scientific Yearbook, pp. 11–45. Lomonosov Moscow State University; Russian Psychological Society, Moscow (2010)
4. Gilchrist, A.L., Kossyfidis, C., Bonato, F., Agostini, T., Cataliotti, J., Li, X., Spehar, B., Annan, V.: An anchoring theory of lightness perception. Psychological Review 106(4), 795–834 (1999)
5. Land, E.H., McCann, J.J.: Lightness and retinex theory. Journal of the Optical Society of America 61, 1–11 (1971)
6. Economou, E., Zdravkovich, S., Gilchrist, A.: Anchoring versus spatial filtering accounts of simultaneous lightness contrast. Journal of Vision 7(12), 2–15 (2007)
7. Wolff, W.: Über die kontrasterregende Wirkung der transformierten Farben. Psychologische Forschung 18, 90–97 (1933)
8. Coren, S.: Brightness contrast as a function of figure ground relations. Journal of Experimental Psychology 80, 517–524 (1969)
9. Epstein, W.: Phenomenal orientation and perceived achromatic color. Journal of Psychology 52, 51–53 (1961)
10. Zaidi, Q., Spehar, B., Shy, M.: Induced effects of backgrounds and foregrounds in three-dimensional configurations: the role of T-junctions. Perception 26, 395–408 (1997)
11. Katz, D.: The world of color. Kegan Paul, Trench, Trubner & Co., London (1935)
12. Burzlaff, W.: Methodologische Beiträge zum Problem der Farbenkonstanz. Methodological notes on the problem of color constancy. Zeitschrift für Psychologie 119, 117–235 (1931)

13. Gelb, A.: Die “Farbenkonstanz” der Sehdinge. In: von Bethe, W.A., von Bergmann, G., Embden, G., Ellinger, A. (eds.) Handbuch der Normalen und Pathologischen Physiologie Band 12. 1. Halfte Receptionsorgane II, pp. 594–678. Springer, Berlin (1938)
14. Gilchrist, A., Annan, V.: Articulation effects in lightness: Historical background and theoretical implications. Perception 31, 141–150 (2002)
15. Menshikova, G., Nechaeva, A.: Does the strength of simultaneous lightness contrast depend on the disparity cue? Perception, ECVP Abstract Supplement 40, 104 (2011)

Locally Adapted Detection and Correction of Unnatural Purple Colors in Images of Refractive Objects Taken by Digital Still Camera

Mikhail Matrosov[1], Alexey Ignatenko[1], and Sergey Sivovolenko[2]

[1] Department of Computational Mathematics and Cybernetics
Lomonosov Moscow State University, Moscow, Russia
{matrosov,ignatenko}@graphics.cs.msu.ru
[2] OctoNus Software Ltd.
sivovolenko@octonus.com

Abstract. We discovered significant error in color in images produced by a digital still camera used to capture scenes with a special setup. Setup includes several LEDs as point light sources and a light-refractive object. Due to light dispersion in the object, vivid monochromatic colored flares appear. However, images captured with a digital still camera occasionally exhibit bright purple (almost pink) colors, which do not correspond to any monochromatic color.

In this paper, we analyze the origins of this effect by examining different properties of the setup and analyzing RAW images. We propose a simple and efficient algorithm for correction of unnatural purple colors by using only a final JPEG image produced by camera. We develop a continuous transform which maps all unnatural colors to the natural ones in a perceptually uniform color space. We also propose a simple segmentation technique to identify image areas to be corrected.

Keywords: color management, color calibration, segmentation, color correction, monochromatic colors, RAW-processing, perceptually uniform color spaces, light dispersion, digital still camera.

1 Background

Consumer digital still cameras are very powerful tools for capturing real world images. Since they are broadly available, well-studied and intensively developed, the cameras are frequently used not only by photographers, but also in many research and engineering applications. The latter applications require precise, repeatable and calibrated results.

Color calibration of a camera is not a trivial task, since camera perceives world in a model-specific color space. In order to be able to process and correctly display such an image, it is necessary to convert it to a certain conventional color space, such as CIE XYZ [12]. The complete chain of color processing in digital cameras including this component is briefly and clearly described by Adams et al. [5]. They describe this conversion to be handled by a 3×3 matrix converting

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XIX, LNCS 7870, pp. 117–130, 2013.

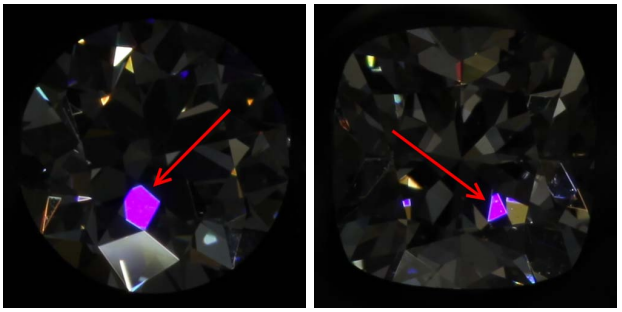


Fig. 1. Images with unnatural purple color for two different objects

camera-specific RGB response to universal XYZ values. The tricky part is that camera spectral sensitivities cannot be represented as linear combinations of CIE color matching functions forming XYZ values.

Therefore, this conversion matrix is usually designed to minimize the average error for a specific set of colorants. Spaulding et al. [13] used Macbeth Color Checker [9] as a target set and an RMS error of CIELAB ΔE^*_{ab} color-difference measure to find an optimal matrix:

$$\Delta E^*_{RMS} = \sqrt{\sum_{i=0}^{N} (\Delta E^*_i)^2},$$

where N is the number of color patches and

$$\Delta E^*_i = \sqrt{(L^*_{si} - L^*_{di})^2 + (a^*_{si} - a^*_{di})^2 + (b^*_{si} - b^*_{di})^2},$$

where L^*_{si}, a^*_{si} and b^*_{si} are the CIELAB scene color values for the i-th color path and L^*_{di}, a^*_{di} and b^*_{di} are the CIELAB reproduced color values for the i-th color path.

Hong et al. [7] used a broader collection of colorants: an ANSI IT8.7/2 [6] chart on Kodak Ektacolor Professional Paper, and the textile samples selected from The Professional Colour Communicator [11] using reactive dyes on cotton. They also performed a polynomial regression with least-squares fitting to minimize the color-reproduction error.

Thus, there are many available techniques to perform conversion of camera RGB response to XYZ values which can generally include some non-linear transformations or multidimensional look-up-tables. And we do not know precisely how a particular camera model handles this conversion since most of camera firmwares are proprietary and closed.

However, most of these techniques are focused on reproduction of colors usually observable in natural scenes, but not of all the physically available spectra. In specific engineering tasks, a certain spectra can be encountered, which a camera would not be able to handle properly. That is the case discussed in this paper.

We analyze optical properties of a transparent colorless object shaped as a polyhedron. Its refractive index is high enough to make it a dispersive media. Thereby, when illuminated by a white light, such an object appears with colored faces. Since colors are induced by light dispersion and the object's faces are small enough, color spectra of a single face is nearly constant and virtually monochromatic. We used a consumer digital still camera to capture images of the described scene and discovered vivid saturated purple colors appearing under certain conditions. Examples of such images are shown in figure 1.

However, such vivid purple colors do not correspond to any monochromatic spectra and we weren't able to witness the same purple faces with naked eyes observation. Thus we have decided that we've encountered the above mentioned case of a camera being unable to properly represent captured color. We present a simple correction algorithm to work with our setup. While the algorithm is very specific and aimed at our particular task, the conducted research is extensive and general.

2 Introduction

The detailed description of our setup, including the notes on illumination, properties of object and camera model, is given in section 3. This section also contains information on how images were obtained and how a collection of analyzed images was formed.

An examination of the issue was done in our previous work [8] and is briefly recalled in section 4.

Though we prove that a consumer camera cannot properly handle discussed colors, we want to use our setup for further research of the described objects. So in section 5 we propose a simple correction algorithm which can be applied to JPEG camera output images to replace unnatural vivid purple colors with ones of a more bluish hue, which can be observed as a result of light dispersion. Firstly, we briefly recall our previous algorithm, and secondly, we propose a number of enhancements to it. The main enhancement is the segmentation technique, used to determine areas in which the correction is applied, as opposed to globally applied previous approach.

Examples of images corrected with the proposed algorithm are shown in section 6. Conclusions and acknowledgments are given in section 7.

3 Setup and Photos

There are three essential components of our setup: illumination, an object and a camera. All of these components are enclosed in a closed box with illumination mounted on the top, an object mounted at the bottom and a camera placed at the front and directed to the object. Figure 2 represents the schematic illustration[1].

[1] Camera icon designed by Go Squared Ltd.

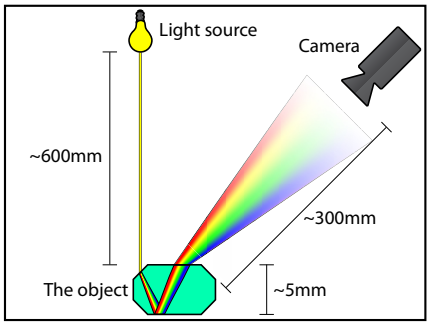


Fig. 2. Schematic illustration of the used setup. Relative sizes of objects and relative distances in the scene are not preserved for illustrative purposes.

Illumination consists of a several bright LEDs with wide warm spectra. The camera white-balance was adjusted automatically prior to the shooting of any images. A sample paper patch with a neutral color was used for this purpose. There were about 50 LEDs, each of which is small enough and is supposed to approximate a point light source.

An object has a shape of a polyhedron with 50-70 faces and is 4-6 millimeters in diameter. It is made of a transparent colorless material with refractive index about 2.41, hence it introduces strong light dispersion and its faces appear to be colored when observed under appropriate illumination from a suitable point of view. The object is fixed on a motorized holder, which allows rotation along the two perpendicular axes, situated in the plane orthogonal to optical axis of a camera. Controlling this holder, one may adjust the position of the object in which a face with a color of interest will be observable by the camera.

The camera is mounted in front of the box and is pointed at the object. It is plugged into and is operated by a computer, so one can capture images of the object without touching the camera, which can lead to undesired vibrations of the box and break down the current dispersive pattern. In our tests we used a Canon EOS 5D Mark II digital still camera with a Canon EF 100mm f/2.8 Macro USM lens and a Kenko Teleplus PRO 300 DGX 1.4x AF teleconverter. However, as we will show later, the explored effect poorly depends on a specific model of the digital camera.

With the given setup, the linear size of an object on captured images becomes 400-600 pixels. To obtain images of an object with purple faces (like the ones shown in figure 1) we rotated the holder slightly in an arbitrary direction and made a shot with the camera. We then studied the acquired image for purple faces and optionally suggested a direction for further rotation. The effect is not rare, therefore it is sufficient to make 3-5 shots of the object to detect a new purple face and additional 2-3 shots to select an appropriate exposure.

Once a purple face was detected and an appropriate central exposure was selected, we made 11-15 shots of the same scene with different exposures using a $\frac{2}{3}$ E.V. step. In other words, by making 15 shots we captured a number of images taken with exposures from $-4\frac{2}{3}$ to $+4\frac{2}{3}$ E.V. relative to the central exposure.

In total, we captured 11-15 exposures for each of the 3-5 positions of 5 objects resulting in 254 images.

4 Examination

A thorough examination of the effect is done in our previous work [8]. This section briefly summarizes it.

One particular set with 15 exposures is examined, images in this set are numbered from 1 to 15 in order of the increasing exposure. The purple faces are manually marked up for this set and the colors within a single image are averaged, as shown in figure 3.

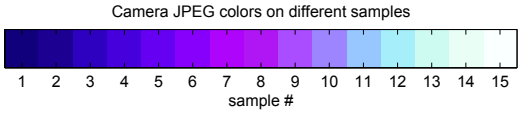


Fig. 3. Averaged colors of a masked purple face for the set of images with the increasing exposure

We then analyze low-level responses of camera sensors stored in camera RAW images. We use the free open-source utility dcraw [3] to obtain a non-interpolated Bayer mosaic in an unknown camera-specific CAM color space. We then conclude that the RAW images are linear on the exposure and the purple faces effect arises at a latter point.

Thus, we consider that the CAM→XYZ conversion is unreliable. As mentioned in section 1, the conversion is optimized to minimize errors in reproduction of normally observable colors and fails to properly handle values induced by purple faces.

5 Correction

In our previous work [8], we proposed a simple and efficient algorithm for correction of purple faces. However, due to its simplicity, it is unable to correctly handle a number of scenarios encountered in the real production environment. Several examples are given in figure 4, accompanied with results of the new proposed algorithm.

In this section we briefly recall the previous algorithm. We then propose several enhancements to it. The most significant enhancement is the introduced segmentation technique, which makes the algorithm locally adapted. Instead of applying the correction to the whole image, we correct only the areas marked as potential purple faces.

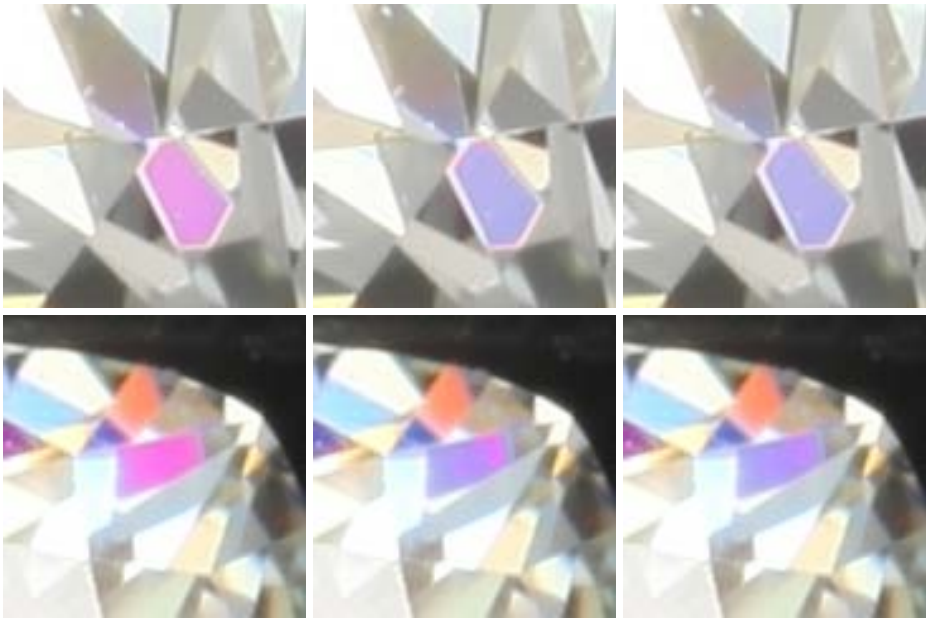


Fig. 4. The comparison between the proposed method (in the right column) and the previous method [8] (in the middle column). Two cropped source images are in the left column. Note the visual artifacts at the borders of the purple faces, produced by the previous method.

5.1 Previous Algorithm

First, we decide to work only with the final JPEG image produced by the camera, hence the correction is applied to this image, and no processing of the RAW file is done.

Then we propose to shift hue of a color and preserve its luminance. For this purpose we are using the Perceptually Uniform Lab color space [1, Uniform Perceptual Lab[2]], or simply the UPLab.

Rotation of the color radius-vector in UPLab within the ab plane corresponds to the Munsell hue [10] shift. Alteration of the radius-vector length corresponds to the shift in the Munsell chroma. We move pixels within the ab plane in the UPLab space preserving the L coordinate to keep luminance unchanged.

We divide the ab plane with four rays outgoing from the origin to four sectors of hues. These rays are shown in figure 5 as OA, OB, OC and OD.

Let the color hue be in the range XY if its ab coordinates lie inside a sector formed by the OX ray, which moves to the OY ray counter-clockwise. Then hues of all possible colors are in one of the ranges AB, BC, CD or DA.

For a given pixel, correction shift depends on its hue:

[2] `http://brucelindbloom.com/UPLab.html`

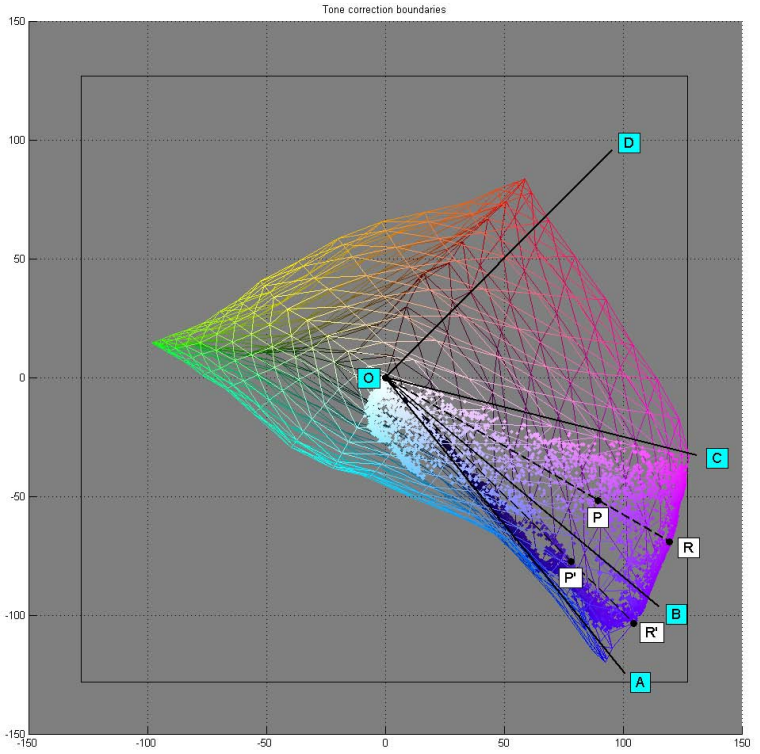


Fig. 5. The sRGB gamut in the UPLab with a several objects. The points depict all of the colors from purple faces in all images from the entire base. The teal marks correspond to correction boundaries. The white marks illustrate the correction.

- Hues in the DA range are not changed.
- Hues in the AC range are shrank to the AB range.
- Hues in the CD range are stretched to the BD range.

By shrinking/stretching one range to another we imply the following. Let P be a point in the ab plane in the UPLab space corresponding to the given color (with a persistent L coordinate). Let the prolongation of the OP ray intersect the sRGB gamut at the point R. Let the correction move P to P' with prolongation of OP' intersecting the sRGB gamut at the point R' (see figure 5). Then, if the color hue in the XY range is shrank/stretched to the $X'Y'$ range, the following equations are met:

$$\frac{\angle XOP}{\angle XOY} = \frac{\angle X'OP'}{\angle X'OY'} \quad \text{and} \quad \frac{|OR|}{|OP|} = \frac{|OR'|}{|OP'|}, \tag{1}$$

which gives us

$$|OP'| = |OP|\frac{|OR'|}{|OR|}. \tag{2}$$

The A boundary passes near a distinctive cluster of blue colors on the sRGB gamut. The B boundary specifies the strength of correction and passes near the "most purple" observable monochromatic color. The C boundary is selected in such a way that all colors of purple faces collected through the entire image base lie within the AC range. The D boundary is selected somewhat arbitrarily and has different locations for the previous and the present works.

The proposed correction algorithm includes intersection of rays with 3D gamuts and conversion from the sRGB color space to the UPLab color space and backwards. These tasks require significant time to be performed, but since we precompute the LUT, they do not affect overall efficiency of the correction algorithm. It took us about 5 minutes to construct a LUT for all possible 16,777,216 sRGB colors. We used CGAL AABB Trees [2, 3D Fast Intersection and Distance Computation (AABB Tree) [3]] to compute intersections of rays with gamuts. The size of full LUT is 48MiB, but we compressed it to ≈7MiB using a run-length-encoding technique since most of the colors are unaffected by the correction and it can still be efficiently accessed in the compressed form.

5.2 Proposed Algorithm

In the present work, we propose a number of enhancements to our previous work. The main enhancement is to use a segmentation technique to select areas in which the correction needs to be applied. It is different from our previous approach, in which we applied the correction globally to the whole image.

The other two enhancements are: expanded CD range and constraint on the OP' length, i.e. the chroma of the corrected color. We discuss these two enhancements in current section, and leave the segmentation technique for a separate section.

We also optimized the exact location of the B boundary to be the average of the three possible locations presented in the previous work.

Expanded Range. By analyzing collected data, we conclude that the previous algorithm tends to perturb smooth gradients in a number of situations. The simplest way to make the correction smoother is to expand the stretched range of colors, i.e. the CD range.

As mentioned in our previous work, the exact location of the D boundary was chosen somewhat arbitrarily. We tried to make the whole range of affected colors considerably smaller to minimize potential distortions of the image. In the current work we place the D boundary at 45° instead of 0° (see figure 5).

After this modification the correction affects more colors, especially close to reds. However, affected colors still lie along the purple line and should not interfere with monochromatic colors.

[3] `http://www.cgal.org/Manual/latest/doc_html/cgal_manual/AABB_tree/Chapter_main.html`

In practice, there are colors of almost any hue on the production images. Hence it is important to preserve colors that we are sure should not be corrected. This is achieved by a segmentation technique discussed below.

Constrained Chroma. The sRGB gamut in the UPLab color space has a complex shape. This is especially pronounced near the magenta-white edge of the RGB cube. It is where the colors of the purple faces are located. After applying the equations (1), $|OP'|$ may be significantly larger than $|OP|$. That means, that the chroma of the color is increased.

We discovered that this effect contributes to the problem of perturbed gradients. Hence we constrain the chroma not to be enlarged by the correction. We replace the equation (2) with the following:

$$|OP'| = |OP| \min\left(1, \frac{|OR'|}{|OR|}\right).$$

5.3 Segmentation Technique

The above correction is applied only to areas of the image marked by a segmentation as possible regions with purple faces. The segmentation produces a binary decision mask, which marks every pixel on the image as to be either corrected, or ignored.

All the colors are divided into three classes:

- Strong purples
- Weak purples
- Others

The strong purples are the colors for which we are sure that they correspond to a purple face. To form these, we compute distance to the colors from marked up purple faces throughout the entire image base. Note that every purple face becomes black on very low exposures and white on very bright exposures. However, black and white are neutral colors and can be encountered anywhere on the image. Therefore we include additional constraint on chroma when we select strong purples.

The weak purples are the colors from the whole range affected by the correction, i.e. the colors from the AD range, except the colors that are marked as strong purples. They need to be processed to smooth transitions between strong purples and the rest of the image.

Finally, *the others* are the colors that are not affected by the correction under any circumstances. These are the colors lying outside the correction boundaries, the AD range.

After colors of all the pixels on the image are classified to one of these three classes, a segmentation mask is constructed. It includes all the connected regions of weak purples, for which there is a neighboring strong purple. That is, strong purples act as a seed pixels for a flood-fill, constrained with weak purples, as follows.

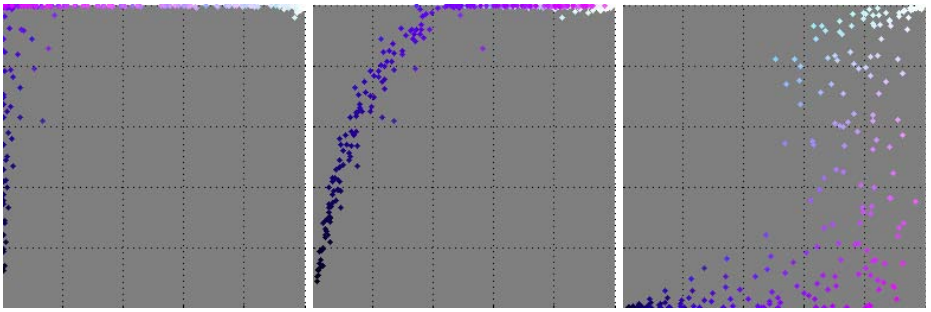


Fig. 6. Projections of median colors of the purple faces on the GB, RB and RG planes of the RGB cube, from left to right. The coordinates are in the sRGB color space.

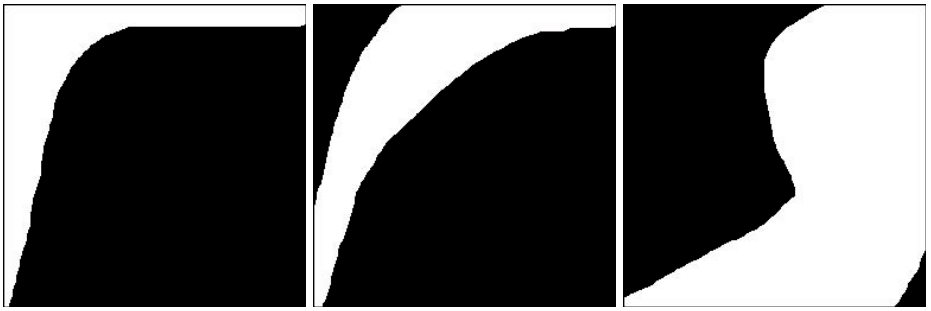


Fig. 7. The masks used for detection of strong purples. A color is considered close to the colors of purple faces if and only if its projections on the GB, RB and RG planes of the RGB cube correspond to white pixels of the masks, from left to right. The outer black strokes are added for visualization only.

Detection of Strong Purples. For every marked up purple face from the entire image base, median color is computed. All these median colors are placed in the RGB cube for the source sRGB color space. They are then projected on the different sides of the RGB cube. The projections are shown in figure 6.

Our task is to define a rule which can determine whether a given color lies within a cluster of colors formed by purple faces. This rule doesn't have to be very precise, since the selected strong purples only act like seeds.

We propose a simple solution for this rule. We manually compose masks for the three projection planes. We then assume, that a color lies within a cluster of colors formed by purple faces if and only if its projections on the planes of the RGB cube lie within the composed masks. These masks are shown in figure 7.

Finally, a color is considered a strong purple, if and only if:

- Its projections on the sides of the RGB cube lie within the given masks
- It lies within the AC range
- Its chroma is greater than 50

The first condition is computed in the sRGB color space. The remaining two are computed in the UPLab color space. For the sake of performance, a technique utilizing a look-up table is used, similar to the one used for the correction. All the sRGB colors are checked to be strong or weak purples, and the results are stored in 3D LUTs. The details about this process are given below.

Construction of the Mask. After colors of all the pixels on the image are classified into strong purples, weak purples, or others, the final segmentation mask is constructed.

Pixels with colors classified as weak purples are divided into connected components. A component is included in the mask if and only if it has a neighboring pixel with a color classified as strong purple. The process is illustrated in figure 8.

Fig. 8. Construction of a segmentation mask. The source image is shown in the left. Classification result is shown in the middle, with strong purples marked as white, weak purples marked as gray, and others marked as black. The constructed segmentation mask is shown in the right.

As one can see, the constructed mask is not very accurate or discriminative. But this is not important, since most pixels accidentally included in the mask are near-neutral. We only want to preserve chromatic colors while correcting purple faces. Neutrals are very slightly affected by the correction.

In order to reduce the noise impact, the mask of strong purples is morphologically eroded before the segmentation mask is constructed. A disk with a 2-pixel radius is used as a structural element for the erosion. The pixels excluded from the mask of strong purples in such a manner, are marked as weak purples.

Implementation. To accelerate the classification process we precompute the results for all sRGB colors. Two look-up tables are constructed at this stage: one for strong purples, and one for weak purples. It is important to mention

that the table with weak purples also contains strong purples. It makes the implementation easier.

These two tables are compressed using the same RLE-technique as the correction table. Additionally, since the masks are binary, they are stored as bit-encoded, with every byte describing eight neighboring colors. Using this encoding we reduced the total size of the tables from 32MiB to 1MiB (250KiB and 850KiB for strong and weak purples respectively).

The connected components are extracted using a simple flood-fill algorithm, started from each pixel marked as a strong purple. The flood-fill is spreading only on pixels marked as weak purples.

The morphological erosion and flood-fill are done using the corresponding functions from the highly optimized Intel® IPP library [4].

6 Results

To ensure that the proposed enhancements contribute to the quality of the results, we manually selected 33 samples from production photos, in which the previous algorithm exhibited visual artifacts. We then manually checked the results on all of the samples. Three of the samples are shown in figure 4. The others can be found in the supplementary materials available on the Internet[4].

Several examples of correction made by the proposed algorithm are given in figure 9. They are not very different from the ones generated by the previous algorithms. For differences see figure 4.

The introduction of local processing slowed the algorithm by 7 times. However, it is still very fast and it takes only 430ms to process a 21Mpix photo on Intel® E8500 CPU running two cores at 3.16GHz.

7 Conclusion

During our engineering work we have stumbled upon limitations of applicability of digital still cameras, where specific colors from our scene could not be reproduced properly. We analyzed this issue and proposed an efficient algorithm for its correction.

In the present work we proposed a several enhancements to the previous algorithm. With these enhancements, the quality of the algorithm is sufficient for processing of production photos. Despite the proposed enhancements complicated the algorithm, it is still very efficient.

The further enhancement of the algorithm may take into account characteristics of the illumination, discrimination of facets and stars and even processing of the geometry of an object.

[4] `ftp://graphics.cs.msu.ru/projects/PurpleFires/2012-11-28-Segmentation results/manual.html`

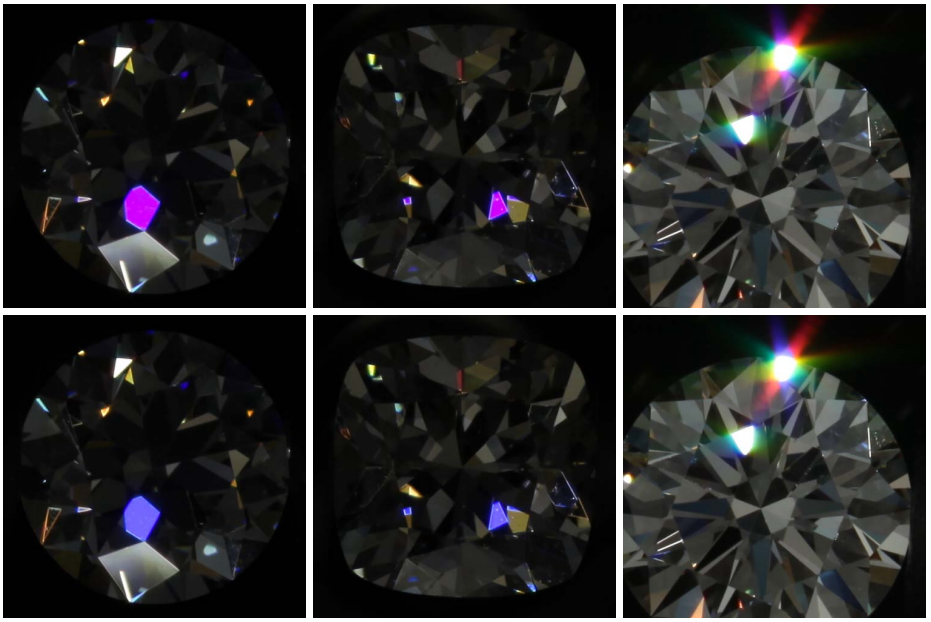


Fig. 9. Examples of the correction made by the proposed algorithm. The source images are shown in the top row. The corrected images are illustrated in the bottom row.

Acknowledgments. This work was done in cooperation with and with financial and technical support of OctoNus Software Ltd.

References

1. Bruce lindbloom's web site, `http://brucelindbloom.com` (accessed: May 27, 2012)
2. CGAL - computational geometry algorithms library, `http://www.cgal.org` (accessed: May 27, 2012)
3. Decoding raw digital photos in linux, `http://cybercom.net/~dcoffin/dcraw` (accessed: November 27, 2012)
4. Intel integrated performance primitives, `http://software.intel.com/en-us/intel-ipp` (accessed: November 28, 2012)
5. Adams, J., Parulski, K., Spaulding, K.: Color processing in digital cameras. IEEE Micro 18(6), 20–30 (1998)
6. ANSI, I.: 7/2-1993 (ISO 12641). Graphic Technology-Color Reflection Target for Input Scanner Calibration
7. Hong, G., Luo, M., Rhodes, P.: A study of digital camera colorimetric characterisation based on polynomial modelling (2001)
8. Matrosov, M., Ignatenko, A., Sivovolenko, S.: Detection and correction of unnatural purple colors in images of refractive objects taken by digital still camera. In: Graphicon (2012)

9. McCamy, C., Marcus, H., Davidson, J.: A color-rendition chart. J. App. Photog. Eng. 2(3), 95–99 (1976)
10. Newhall, S., Nickerson, D., Judd, D.: Final report of the OSA subcommittee on the spacing of the munsell colors. JOSA 33(7), 385–411 (1943)
11. Park, J., Park, K.: Professional colour communicator-the definitive colour selector. Journal of the Society of Dyers and Colourists 111(3), 56–57 (1995)
12. Smith, T., Guild, J.: The CIE colorimetric standards and their use. Transactions of the Optical Society 33, 73 (1931)
13. Spaulding, K., Vogel, R., Szczepanski, J.: Method and apparatus for color-correcting multi-channel signals of a digital camera, US Patent 5,805,213 (September 8, 1998)

Some Theoretical Issues of Scientific Visualization as a Method of Data Analysis

Victor Pilyugin[1], Eugeniya Malikova[1], Valery Adzhiev[2], and Alexander Pasko[2]

[1] National Research Nuclear University "MEPhI", Moscow, Russia
[2] National Centre for Computer Animation, Bournemouth University, Bournemouth, United Kingdom
{pilyugin,malikova}@sv-journal.com,
{vadzhiev,apasko}@bournemouth.ac.uk

Abstract. The paper discusses scientific visualization as a modern computer-based method of scientific data analysis in experimental and theoretical research. A definition of this method and descriptions of some of its characteristics are given as observed by the authors from the generalization of practical experience. An example of the implementation of this method are provided and illustrated on the basis of the HyperFun programming language and its supporting software tools.

Keywords: Scientific data analysis, scientific visualization, spatial scene, geometric modeling, Function Representation.

1 Introduction

As it appears in practice, scientific visualization is seen in the general sense as a set of techniques for graphical representation of data in theoretical or experimental research. In other words, scientific visualization is based on using computer tools for the transformation of the "invisible" raw data into visible graphical data ("to make invisible visible"). Such tools are applied with the purpose of the following analysis and understanding of this raw data.

We consider scientific visualization as a modern computer-based method of data analysis. The essence of this method is in establishing the correspondence between the initial data being analysed and its static or dynamic graphical interpretation, which is analysed visually, and the results of this analysis of graphical data are interpreted in terms of both the initial data and the reality behind it (Fig. 1).

With such an interpretation of scientific visualization, both the stage of the visualization of initial data ("scientific visualization" in the general sense as mentioned above) and the visual analysis of the obtained results are considered as a single integrated process. Initial data analysed with the scientific visualization method can be of various nature. The goals of the data analysis can be different as well. As a result, various graphical representations can be involved. This entire process gives an opportunity to utilise enormous potential abilities of the spatial imaginary thinking of the researcher in data analysis.

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XIX, LNCS 7870, pp. 131–142, 2013.

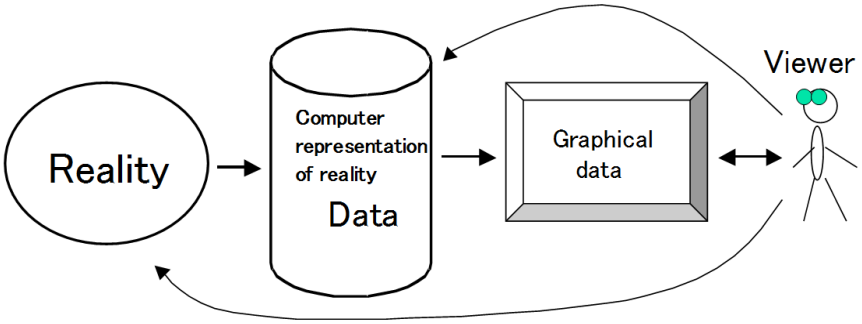


Fig. 1. The method of scientific visualization

Further evolving of this method along with extending its scope makes the development of its scientific and engineering basis worth pursuing. In this paper, some generalization of practical results is proposed on the basis of the authors' experience in solving various data analysis problems using the method of scientific visualization.

2 Some Theoretical Generalizations

1. First of all, we consider an answer to the following question (which seems a rhetorical one at the first glance) is worth clarifying - **what should be understood in the general case under scientific data and scientific data analysis in the context of visualization?**
   The distinguishing feature of modern scientific research is wide use of computers, which is true for both experimental and theoretical research. When discussing theoretical research, we mean the modern style of such research called a computational experiment, when a mathematical model of a real or imaginary object is analysed using a computer. Experimental research of material objects is quite close to mathematical research of abstractions with computers. Here, mathematical abstractions serve as a subject of research and a specific in-situ experiment takes place.
   When conducting such research, various data is obtained describing the object under study. **This scientific data is numerical data represented in the computer memory in various forms.** This is typically ordered finite sets represented in several interconnected tables, and as such they can be considered as **numerical tabular data**.
2. **Analysis of the numerical tabular data** obtained as a result of an in-situ or computational experiment, or in the process of mathematical abstraction research using a computer, is generally **preceded by preliminary processing of this data**. The goal of processing raw data is to increase its quality in the context of the following analysis. Typical processing operations include data interpolation. filtering and decimation. These operations can be applied separately or in aggregation.

As the result of processing, new numerical data is generated, which then undergoes an actual analysis. We will further refer to this generated data under analysis as scientific data or simply data.

3. The process of data analysis can be considered a solution of **the data analysis problem** with the following problem statement:

Given:
Numerical tabular data D describing the object under consideration.
Required:
Obtain conclusions C of interest to the researcher regarding the object.

4. **The solution of the above stated problem** by the method of scientific visualization means reducing this problem to the following two problems solved in succession.

The first problem is to represent the given data in some graphical form (actual problem of data visualization). This problem is solved using computer. **The second problem** is to visually analyse the graphical image(s) obtained as the result of solving the first problem. The results of this analysis are interpreted in respect to the initial data. This problem is solved directly by the researcher.

5. **The algorithm f for solving the first problem** can be represented as a superposition of the following two mappings:

$$f = f_1 * f_2, \tag{1}$$

where

$$f_1: D \rightarrow (L,O)$$

$$f_2: (L,O) \rightarrow GR,$$

and D is a set of initial numerical data in the given form,
L is a set of geometric models of imaginary spatial scenes (i.e., one or several material and/or abstract spatial objects) of a selected type,
O is a set of optical models of the spatial scene(s),
GR is a set of graphical images of the spatial scene(s).
A block-diagram of the algorithm f is presented in Fig. 2.

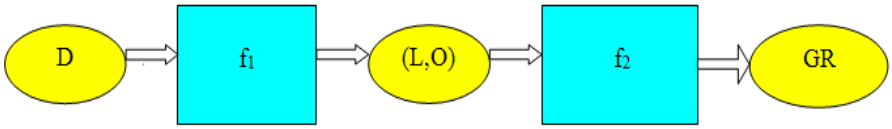


Fig. 2. Block-diagram of the algorithm f

In a particular case, the mapping f_1 can have a form f_1: D → L, where an element of the set O is defined by the researcher at the stage of the algorithm execution.

In general case, the geometric models L are collections of static and dynamic point sets with numerical parameters in the Euclidean space E^3. The optical models O represent optical characteristics of the spatial scene objects defined as dependent numerical variables on the point sets of E^3 of geometric models in the scene. Such characteristics can be light reflection and refraction coefficients, which also can depend on time.

The mapping f_2 is parameterized by such variables as the type of projection, camera location and orientation, light source parameters and spatial media, which constitute attributes of visualization.

It is worth to note that algorithms of type f of the stepwise transformation of initial data into graphical images are known as visualization conveyors (or visualization pipelines). This terminology is widely used when discussing scientific visualization in the mentioned above general sense (as making invisible visible).

The above notions can serve as a conceptual basis for the development of an algorithm of visualization of initial data. The mapping f_2 is a standard visual rendering of spatial scenes. The mapping f_1 reflects spatial modeling of initial data. When defining the mappings f_1 and f_2, one needs to specify the sets D, L, O and GR under consideration.

While the mapping f_2 is well known and assumes usage of 3D computer graphics tools, the mapping f_1 is less studied and understood. This mapping depends very much on the form of the initial data representation. For example, this data can represent numerical values of quantitative input and output variables of the studied mathematical model as well as numerical values of parameters of geometric, vector, tensor or other types of variables of this mathematical model.

Following from our experience in different application areas of scientific visualization, the mapping f1 can be represented as a superposition of two mappings s_1 and s_2:

$$f1 = s_1 * s_2 \qquad (2)$$

where

$$s_1: D \rightarrow N$$

$$s_2: N \rightarrow (L,O)$$

and N is a set of geometric models of the initial data.

In a particular case, the mapping s_2 can have a form s_2: N → L, where an element of the set O is defined by the researcher at the stage of the algorithm execution.

In contrast to geometric models of the set L, geometric models of the set N can be represented as static or dynamic geometric point sets with numerical parameters in 3D Euclidean space E^3 as well as in multidimensional geometric spaces. An example of such space is an n-dimensional Euclidean space E^n with the points represented as n-tuples of real numbers. The mapping s_1: D → N itself can in

general case include complex geometric operations being applied in one or several geometric spaces.

In addition, geometric point sets in 3D and multidimensional spaces can have point wise attributes defined by additional mappings of point sets onto real numbers, vectors or other mathematical objects.

In the simplest case, the set N is equivalent to the set L, and the mapping f_1 takes the form f_1: D → (L,O).

6. So far we have discussed the main characteristics of the first problem solution. **Let us now consider the process of solving the second problem**, namely a visual analysis of graphical images of the initial data and subsequent interpretation of the result of analysis in respect to the data, i.e., making some conclusions C regarding the object under study. Both analysis and interpretation are performed by the researcher solving the problem of data analysis.

   What does actually visual analysis of graphical data representation constitute? It is a matter of principle to understand that visual analysis in its essence consists in qualitative analysis of the spatial interpretation, which is put in correspondence to the initial data by the mapping f1: D → (L,O). This means that the obtained graphical images serve just as means for natural and convenient conveying the spatial interpretation to the researcher for its analysis followed by the interpretation of the results in terms of the initial scientific data.

   Thus, the method of scientific visualization as a method of scientific data analysis is a method of spatial modeling of this data, which allows for using enormous potential abilities of a researcher to use their spatial imaginary thinking in the process of data analysis.

   The process of the visual analysis of a graphical image is hard to precisely formalize. The efficiency of the visual analysis depends on the researcher' experience and their inclination to use spatial imaginary thinking. When observing an image, the researcher can solve three main problems:

   - analysis of shapes of spatial objects;
   - analysis of mutual spatial positions of spatial objects and their topological relations;
   - analysis of graphical attributes of spatial objects.

   As the result of solving these problems, some conclusions are made by the researcher regarding the spatial scene. As mentioned above, these conclusions are then interpreted in terms of initial data and thus conclusions are formed regarding the object under study.

   The researcher is either satisfied by the results of analysis or repeats the algorithm f or its part to obtain better results by changing the values of some parameters. For example, they can get different projection images of the spatial scene to better analyze the shapes of spatial objects in it. As the result, the process of data analysis using the method of scientific visualization becomes more complex, **iterative and interactive** one.

7. As it was mentioned above, the method of scientific visualization consists in spatial modeling of initial data. The introduced spatial scene is analyzed visually, i.e., sensory analysis of the spatial scene is performed using a specific human organ of

senses, namely a human eye. However, in the combination with human eyes, other organs of senses such as human ears can be used.

For example, when analyzing a given scalar field, a spatial scene can be put into correspondence with it, which includes a traditional isosurface (or several of them) as well as an imaginary abstract point sound source with the amplitude proportional to the scalar field value at the given point. As a result, an additional possibility of listening to the spatial scene using sound rendering appears, i.e., **extended scientific visualization** takes place in this case.

8. After **the algorithm f for solving the first problem** has been specified, **it is implemented on the computer** for the given initial data from the set D. This implementation assumes the development of an application software program in accordance with the algorithm using some programming language required by the used software tool. As we mentioned earlier, the algorithm f can be quite complex. This assumes that the implementation can involve quite sophisticated instrumental software tools supporting the entire system of concepts used by the researcher to formulate the algorithm including multidimensional and time-varying geometric modeling tools.

   Our experience shows that the programming language and software tools of the HyperFun project based on the Function Representation (FRep) of geometric objects well suite the purposes of the described instrumental tools (Pasko et al., 1995) , (Pasko et al., 2001), (Pasko et al., 2004), (Cartwright et al, 2005), (Adzhiev et al.,1996), (Adzhiev et al.,1999). We discuss the characteristics of FRep and HyperFun in the following section.

3 Function Representation and HyperFun Language

HyperFun (Pasko et al., 2004), (Cartwright et al, 2005), (Adzhiev et al.,1999) is a special-purpose programming language for geometric modeling using FRep. The system of concepts used in HyperFun includes sets of geometric objects (point sets), geometric operations and relations. FRep defines a geometric object in a multidimanesional space as whole with a single real continuous function F of several variables X (point coordinates) (Pasko et al., 1995), (Pasko et al., 2004) in the form of the inequality $F(X) \geq 0$ providing the point membership classification rule as well as an algebraic measure of the distance from the point to the object boundary. The geometric object definition is given in a multidimensional space, which allows for selecting an appropriate dimensionality for a specific application. Note that the function F can be defined in various ways including analytical equations as well as pure procedural definitions. In this sense, FRep generalizes the traditional usage of real functions in skeletal implicit modeling and Constructive Solid Geometry (CSG).

A program in HyperFun defines an algorithm of the function evaluation of a single geometric object based on the vector of point coordinates and a vector of numerical parameters. In the modeling process some finite number of pre-defined geometric primitives can be employed with the known definition and a set of numerical parameters. However, it is not entirely necessary as the user can define geometric objects

directly using equations and function evaluation procedures. This allows for the unification of several modeling styles including CSG, free form skeletal objects, sweeps, voxels, meshes and point clouds converted to real functions.

Each geometric operation in FRep has to yield a continuous real function for its resulting geometric object thus making the operation closed on the representation. Examples of basic geometric operations in FRep are set-theoretic ones, blending, offsetting, bijective mappings, projection, Cartesian product, metamorphosis and others. Examples of relations are inclusion, point membership, collision and others.

The work (Pasko et al., 2001) introduced a model of constructive hypervolumes, which allows to model heterogeneous volume objects as point sets with point wise attributes such as material, its density, colour and other physical properties (see details in (Pasko et al., 2008). HyperFun in its latest version supports constructive hypervolume modeling via defining point wise attributes in an additional input/output array. For visualization of spatial scenes with HyperFun objects with specialized rendering tools, the user can define optical characteristics of objects by selecting color, reflectance and refraction properties.

The mapping f_1 discussed above can be implemented quite naturally in HyperFun when initial data is given in the form of functional dependencies. In the case of discrete numerical data, some interpolation procedures have to be involved to obtain continuous functions. An example of the mapping f_2 is a procedure of the piecewise linear approximation (polygonization) of the FRep object surface by a set of triangles with their following projection onto the selected image plane. In (Pasko et al., 1986) a polygonization algorithm was proposed and implemented, which is used in several HyperFun rendering tools. This algorithm is free of topological ambiguities essential for the original Marching Cubes algorithm (Lorensen & Cline 1987). The trilinear interpolation inside the cubic cells and the bilinear interpolation on their faces are used to detect and connect hyperbolic arcs on the faces, which define the sides of the generated triangles.

4 Practical Example

Let us discuss and illustrate an example of the application of the method of scientific visualization for data analysis.

4.1 Problem Statement

The object under study is a computer model of the C_2H_2 molecule.

Given

A mathematical model of the scalar field of electron density of this molecule in the following form:

$$Y = \phi (x,y,z|t), \text{ where}$$

x,y,z are coordinates of points in space, Y is an electron density given in the tabular form,

$$x_1 \le x \le x_2,\ y_1 \le y \le y_2,\ z_1 \le z \le z_2;$$

t is a parameter associated with the function Y defined for the values $t=t_1$ и $t=t_2$ ($0 \le t_1 \le 1$, $0 \le t_2 \le 1$) as

$$t=t_1: Y = \phi_1(x,y,z)$$

$$t=t_2: Y = \phi_2(x,y,z).$$

Required

To analyze variations of the function Y depending on the parameter t.

We will solve this problem using the method of scientific visualization.

4.2 Solving the First Problem

According to the method described above, the algorithm f of solving the first problem is a superposition of two functional mappings:

$$f = f_1 * f_2$$

Mapping f_1 includes the following steps.

Let us define the geometric model of the spatial scene.

1. Let us introduce interpolation functions $\tilde{\phi}_1(x,y,z)$ and $\tilde{\phi}_2(x,y,z)$ corresponding to the tabulated functions $\phi_1(x,y,z)$ и $\phi_2(x,y,z)$. Let us also introduce a function of four variables $\Psi(x,y,z,t) = (1-t)*\tilde{\phi}_2+t*\tilde{\phi}_1$. The geometric interpretation of this function is the hypersurface G_5 in the Euclidean space E^5 with Ψ as the defining function. This geometric interpretation defines the mapping s_1: D → N discussed earlier. Thus, we use here s_1 as a superposition of s_{11} and s_{12}, where s_{11} is a mapping of tabular functions $\phi_i(x,y,z)$ onto the set of interpolation functions $\tilde{\phi}_i(x,y,z)$, and s_{12} is the known from analytical geometry mapping of the set of real functions of four variables onto the set of geometric objects (point sets) of the Euclidean space E^5.
2. By assigning a constant value $t=t_i$ we can put the given hypersurface G_5 in E^5 into correspondence with a hypersurface G_4 with the definition $Y = \phi_i(x,y,z)$. This can be interpreted as a geometric operation of intersection between the hypersurface G_5 and the hyperplane $t=t_i$ followed by projecting the result of the intersection onto the space E^4. The hypersurface G_4 in its turn can be put into correspondence with a collection of isosurfaces C_j in the space E^3 by selecting level values c_j for the function Y. This can be interpreted as a geometric operation of intersection of the hypersurface G4 with hyperplanes $Y = c_j$ followed by projecting of the results onto E^3. Such a procedure defines the above mentioned mapping s_2: N → L, which in this case maps hypersurfaces in E^5 onto sets of isosurfaces in E^3.

3. The obtained collection of isosurfaces is considered a geometric model of an imaginary spatial scene, which is introduced according to the initial mathematical description of the scalar electron density field as well as the mappings s_1 and s_2.
4. Let each of the level values c_j of the function Y correspond to some color of the isosurface selected from some color scale. Thus, we have completely specified the mapping f_1.

Mapping f_2

Rendering of isosurfaces can be implemented through their approximation by triangle meshes with the following mesh rendering, or through a ray-tracing procedure directly defining color for each pixel. For simplicity we can use default rendering parameters for the implementation of the mapping f_2.

Thus we have completely defined the mapping $f = f_1 * f_2$.

4.3 Solving the Second Problem

By visually analyzing the obtained images of isosurfaces we make conclusions about the changes in the shapes of isosurfaces according to the changes of the parameter t. These conclusions are interpreted then in terms of the properties of the scalar field being analyzed.

4.4 HyperFun Program Implementing the Algorithm f1 and Results of Its Work

This HyperFun model defines the hypersurface G_5, its cross-section $t=t_i$ projected on G_4, and a union of 3D volumes with zero-level isosurfaces.

```
my_model(x[3], a[1],s[1])
{
array xt[4];
array xt2[3];
array c[1];

s[1]   = 0.0;
c[1]= 0.0;
-- defining parameter t=ti;
t=0;

-- defining point coordinates in space E4:
{xt1,xt2,xt3,t}
xt[1]=x[1];
xt[2]=x[2];
xt[3]=x[3];
xt[4]=t;
```

```
-- defining point coordinates in space E3: {xt1,xt2,xt3 }
xt2[1]=x[1];
xt2[2]=x[2];
xt2[3]=x[3];


while (t<=5) loop
-- assigning constant values t=ti
xt[4] = t*0.2;
-- shift of each hypersurface G4 along X axis
xt2[1]=x[1]+3*t;

-- interpolation function of the tabular data
-- in file t1->Y=ϕ1(xt2[1],xt2[2],xt2[3])
sp1=rscalars(1,xt2,"ed1.txt");

-- interpolation function of the tabular data
-- in file t2->Y=ϕ2(xt2[1],xt2[2],xt2[3])
sp2=rscalars(2,xt2,"ed2.txt");

-- defining the projected cross-section of hypersurface
G5
-- with defining function of 4 variables by a hyperplane
t=ti
sp3=(1-xt[4])*sp1+xt[4]*sp2;
if (t = 0) then sp4=sp1;
else
sp4=sp4 | sp3;  -- union of all projected cross-sections

endif;

t=t+1;
endloop;

my_model = sp4;
}
```

To obtain good quality images when rendering isosurfaces, the VTK library was used, which includes a procedure of rendering a group of isosurfaces C_j in E^3 with the given function Y and a set of level values c_j. The images in Fig. 3 were rendered using this VTK procedure for the above HyperFun model and several values c_i to get images of several isosurfaces for each given value t_i.

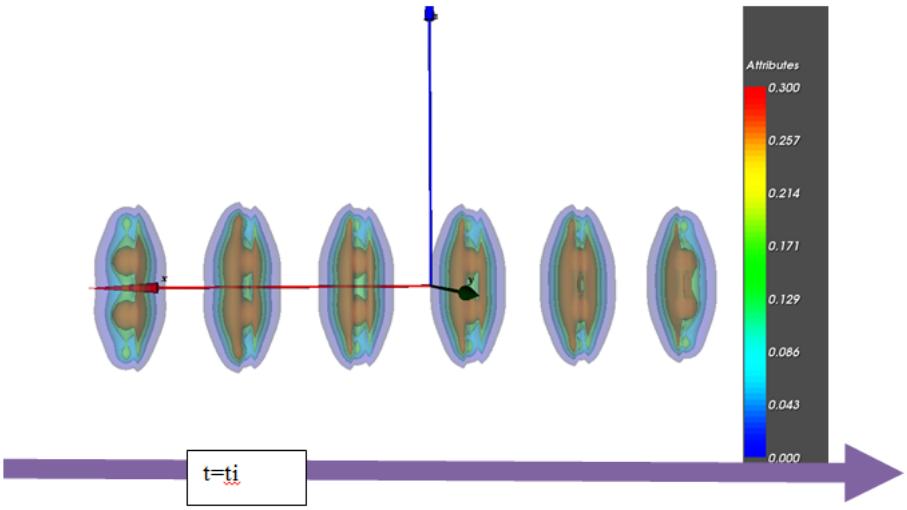


Fig. 3. Nested isosurfaces of the scalar electron density field for different values of parameter t

Fig. 3 shows images of isosurfaces of the scalar electron density field, which allow for visual analysis of the spatial scene and interpretation of the analysis results in terms of initial data, thus forming conclusions regarding the properties of the electron density field itself.

In particular, while analysing the shapes of isosurfaces and their colors for each parameter value t_i the user makes conclusions regarding the variations in the values of the electron density field on the given parameter interval $0 \leq t \leq 1$. Fig. 4 shows projections of the same spatial scene with different camera parameters of rendering.

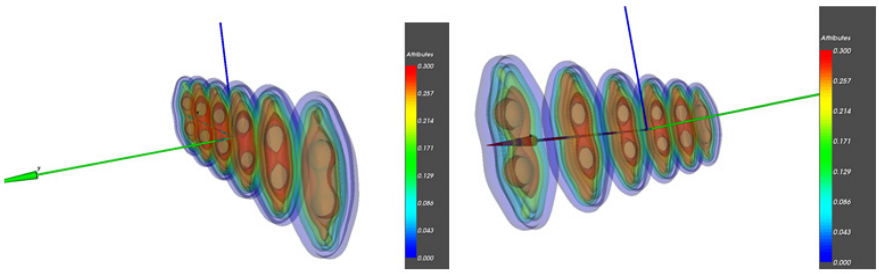


Fig. 4. Different camera positions in rendering isosurfaces of the electron density field

5 Conclusions

We introduced several theoretical generalizations obtained from practical experience of solving various problems of data analysis with the method of scientific visualization. We would like to underline that from our point of view the introduction of mappings s_1 and s_2 is the matter of principle in spite of the fact that in many problems of data analysis these mappings are trivial. With the increasing data complexity and the growth of data dimensionality, the importance of rigorous mapping of data onto geometric models will undoubtedly grow. Otherwise, solving the second problem of visual analysis of graphical images and interpreting the results in terms of initial data will be increasingly difficult.

We believe that the introduced generalizations will be useful for practitioners in data analysis and for developers of future scientific visualization systems. We continue our research in this direction to provide systematic views on specifics of each presented mapping.

References

1. Adzhiev, V., Cartwright, R., Fausett, E., Ossipov, A., Pasko, A., Savchenko, V.: HyperFun project: a framework for collaborative multidimensional FRep modelling. In: Proceedings of Implicit Surfaces 1999, Eurographics/ACM SIGGRAPH Workshop, pp. 59–69 (June 1999)
2. Adzhiev, V., Pasko, A., Savchenko, V., Sourin, A.: Modeling shapes using real functions. Open Systems 5(19), 14–18 (1996)
3. Cartwright, R., Adzhiev, V., Pasko, A., Goto, Y., Kunii, T.: Web-based shape modelling with HyperFun. IEEE Computer Graphics and Applications 25(2), 60–69 (2005)
4. Pasko, A., Adzhiev, V., Comninos, P. (eds.): Heterogeneous Objects Modelling and Applications. LNCS, vol. 4889, 285 p. Springer, Heidelberg (2008)
5. Lorensen, W., Cline, H.: Marching Cubes: A high resolution 3D surface construction algorithm. Computer Graphics 21(4), 163–169 (1987)
6. Pasko, A., Adzhiev, V.: Function-based shape modeling: mathematical framework and specialized language. In: Winkler, F. (ed.) ADG 2002. LNCS (LNAI), vol. 2930, pp. 132–160. Springer, Heidelberg (2004)
7. Pasko, A., Adzhiev, V., Sourin, A., Savchenko, V.: Function representation in geometric modelling: concepts, implementation and applications. The Visual Computer 11(8), 429–446 (1995)
8. Pasko, A., Adzhiev, V., Schmitt, B., Schlick, C.: Constructive hypervolume modelling. Graphical Models 63(6), 413–442 (2001)
9. Pasko, A., Pilyugin, V., Pokrovsky, V.: Geometric modeling in the analysis of trivariate functions. Computers and Graphics, vol 12(3/4), 457–465 (1988)

Pose Refinement of Transparent Rigid Objects with a Stereo Camera

Ilya Lysenkov and Victor Eruhimov*

Itseez,
603000, Korolenko 19b, Nizhny Novgorod, Russia
{ilya.lysenkov,victor.eruhimov}@itseez.com

Abstract. We propose a new method for refining 6-DOF pose of rigid transparent objects. The algorithm is based on minimizing the distance between edges in a test image and a set of edges produced by the training model with a specific pose. The model is scanned with a monocular camera and a 3D sensor such as a Kinect device. The pose is estimated from a monocular image or a stereo pair. The method does not require a CAD model of the object. We demonstrate experimental results on a set of kitchen items essential for any home and office environment.

Keywords: pose estimation, localization, transparent objects.

1 Introduction

Perception for personal robotics is a wide and important application of computer vision. A personal robot is expected to efficiently interact with the environment. In particular, it has to be able to detect a specific object in a scene and find its pose for grasping and manipulation. Recent advances in object recognition and pose estimation [1] demonstrate good results with a monocular camera for textured objects. SIFT features are used to find similarities between training and test textured image patches and then geometric validation is used to filter out false matches. Since the training set contains 3D coordinates of all features, pose estimation in this approach is done by solving a PnP problem on SIFT matches. However if an object has few textured features, local descriptors will produce few matches and detection will fail. Moreover, if only a small part of the object is textured, it will be detected but there may be a substantial error in the pose estimation. Also, this method does not work with transparent objects.

Both textureless and transparent objects such as cups, dishes, staplers etc. are an essential part of home and office environment. The problem of estimating the pose of such objects is important for personal robotics. While recent developments in structured light sensors such as Kinect shows promising results in finding the pose of textureless objects, this type of technology does not work with specular and transparent surfaces. Our work in this paper is largely influenced by the methods for textureless objects coming from industrial robotics [2]

* This work was supported by Willow Garage.

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XIX, LNCS 7870, pp. 143–157, 2013.

that use a CAD model of an object to estimate its pose from a monocular camera by projecting the model to a test image and comparing object features with image edges. While CAD models of manipulated objects in industrial settings are usually available anyway, CAD models of all objects in the personal space are hard to capture.

We present an algorithm for refining 6-DOF pose of a transparent object using edge features. The method does not require a CAD-model, it needs a 3D scan of an object including a point cloud and images registered to each other. We show that the method can be used for accurate pose estimation of transparent rigid objects.

2 Related Work

Transparent objects are very challenging objects in computer vision because their appearance in an image largely depends on a background. Also it is hard to capture a 3D model or a point cloud for transparent objects due to limitations in technologies of existing 3D sensors and because reconstruction of transparent objects is still a very hard problem [3].

The algorithm for detection and reconstruction of unknown transparent objects was proposed in [4]. The algorithm uses two views of a test scene captured by a ToF camera. The algorithm is insensitive to changes in illumination and it was applied for grasping of isolated transparent objects by a robot. Grasping was successful in 41% of reconstructed objects and failed attempts are explained by errors in objects reconstruction and pose estimation.

The algorithm for pose estimation of transparent objects from two views of a test scene was proposed in [5]. Accurate pose estimation was achieved but the objects are required to stay on a table plane and they should be separated from each other. So the algorithm is not able to estimate 6-DOF pose.

Kinect sensor is used for pose estimation and recognition of transparent objects in [6]. However, results are reported only in case when objects are assumed to stay on a table plane. So accuracy in case of 6-DOF pose estimation is unclear.

Specularities are important features when working with transparent objects and there are very promising approaches to pose estimation [7], [8], [9] using this cue. However, these algorithms of pose estimation require a triangulated mesh or a CAD model of an object and they were evaluated with textureless objects only.

Texture features like SIFT are not suitable when working with textureless and transparent objects because such objects don't have their own texture. Computer vision research [10] and psychological studies [11] show that edges and contours of objects are important features and they can be used successfully for the object recognition problem. For example, humans can recognize objects from rough pencil sketches although texture is missing. This cue is available both for transparent and textureless objects and it makes the problem of pose estimation of transparent objects related to pose estimation of textureless objects.

The problem of untextured pose estimation has a long history in computer vision. See [12] for a detailed overview of the 2D-3D pose estimation problem.

[13] shows that it is possible to estimate a pose of a textureless object by using single-view object detection algorithms. However the 2D object representation used in this method is viewpoint-dependent, so a set of detectors has to be trained for different viewpoints. Running all detectors is infeasible in the general case so pose clustering is used [14], [13], [15], first to make a rough estimation of the pose and then refine it by running a smaller set of detectors. The pose corresponding to the most confident detector is returned as an estimation of the object pose. But the accuracy of this estimation is bounded by the number of detectors that also defines the computational cost.

General multi-view approaches and a 3D model of an object are required to balance between the computational cost and the pose estimation accuracy. The idea to use a 3D representation of an object for recognition is going back to early computer vision of 70's and 80's, see, for example, [16]. Approaches [17], [2], [18] utilize this idea and they can estimate a pose of a textureless object quite accurately. Algorithms [17], [18] find the closest training pose and run a local optimization of it using a CAD model of an object. High-quality CAD-models are hard to obtain and although there are some CAD-models of typical household objects (like a cup or a bottle), models are not available for all specific objects that robots need to grasp in a household environment.

Our approach to pose refinement step is similar to [17], [18] and also based on edges cue. However, it does not require a CAD model and it is able to estimate 6-DOF pose of transparent objects.

3 Proposed Approach

To solve the considered problem we divide it to following tasks:

1. Create a 3D model which allows to generate object edgels (points on edges) for different poses. Our model contains a 3D object model and a 3D edge model. The 3D object model is a point cloud of the whole object and it is used to generate silhouette edges. The 3D edge model is a point cloud with points on surface edges, that is edges created by depth discontinuities or texture.
2. Determine a cost function which estimates dissimilarity between generated edgels and the observed test data and then minimize the cost function by varying parameters that determine pose of the object.

We will address all of these steps in the following subsections.

3.1 Creation of the 3D Model

There are no stable ways to estimate depth or produce point clouds for transparent objects [3]. So we take a copy of the object, paint it with a color and use the painted object in the model creation pipeline.

The 3D object model is created automatically from the train data. We scan each object on a planar surface with a Kinect device. Two fiducial markers

consisting of grids of circles are placed in the field of view to provide accurate registration of frames. Depth map from Kinect allows us to segment the plane and calculate the object mask in each image.

We illustrate the algorithm of the surface edge model creation using a textureless object that has many surface edges (Fig. 1). First, we extract 3D points that correspond to surface edges in each frame, then we register point clouds from different frames, and, finally, we build a surface edge model.

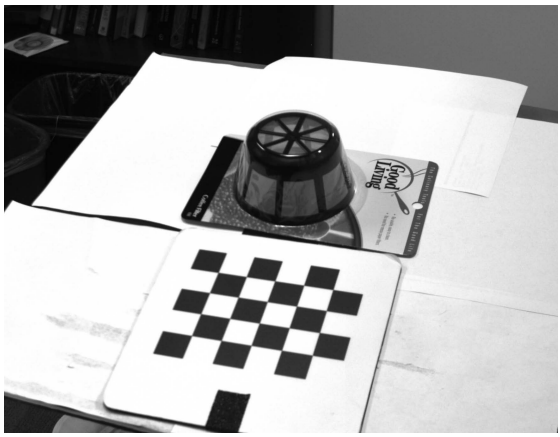


Fig. 1. Example of an object to be modeled

Detecting Edges in Each Frame

1. Find edges on each image of the object using Canny edge detector [19] . Then find edges of the object by intersecting the detected edges with the object mask.
2. Select the points from the 3D cloud that correspond to image edges. Our point cloud is interpolated to the size of the train image and so there is a bijection between 3D points and image pixels. As a result we get a 3D edge model for each training image.

Registering Point Clouds

1. Transform all models to the same coordinate system associated with the first frame, using the poses from the fiducial markers. The corresponding points from different frames would coincide with each other in the ideal case but there are always some deviations in practice due to noise (see the Fig. 2A).
2. Register transformed point clouds. There is a classic and widely used algorithm Iterative Closest Point (ICP) for registration of two point clouds [20], [21]. Global approaches like [22] are used for registration of multiple point clouds because they can distribute registration error between all point clouds

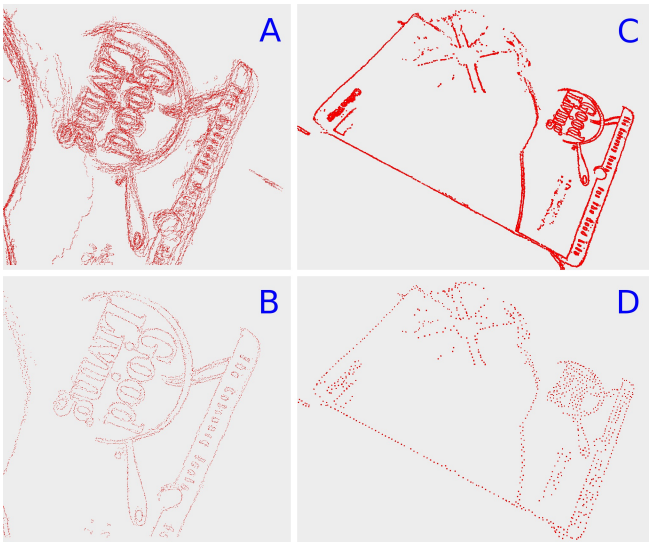


Fig. 2. Creation of the surface edge model. (A) All train point clouds transformed to the same coordinate system. (B, C) Refined and denoised point cloud with k-partite matching and robust statistics. (D) Downsampled point cloud which approximates the full model well.

evenly. We have a good initial alignment of point clouds using the poses from the fiducial markers so we have used more simple global algorithm [23] with LM-ICP [24] to register pairs of point clouds.

Creating a Surface Edge Model

1. Partition all transformed points into groups where each group corresponds to the same point of the object. This allows to get more accurate coordinates of the object point by its noised observations. Partitioning is done by solving the problem of k-partite matching which is a generalization of the bipartite matching for the case of k-partite graphs. It is known to be an NP-hard problem [25] so we used a heuristic algorithm based on [26] (see appendix for details).
2. For each group compute accurate coordinates of the model point using robust estimation of location [27] e.g. the minimum covariance determinant estimator (MCD) [28]. The constructed model is given in the Fig. 2B and it represents edges of the object much better than transformed point clouds in Fig. 2A.
3. Downsample the constructed 3D edge model. The 3D edge model of the whole object is given in the Fig. 2C. It contains many close points that don't give additional information. So we keep 10% of points to lower computational costs of further processing. It is done by a trivial adaptation of the

Douglas-Peucker algorithm [29] for this task. The downsampled model is given in the Fig. 2D.

It is important to note that as a result of the k-partite matching the silhouette edges, which presence depends on a point of view, will be automatically filtered out as they will not have correspondences in different frames.

We group all surface edge points into contours by proximity. 3D orientation of a contour at a point can be estimated as direction of the tangent vector to the 3D contour at this point. We do this by generalizing [30] to the 3D case by means of multi-dimensional robust statistics [27]. When the model is transformed in 3D space, orientations of points are transformed as usual 3D points. We use the contours to calculate the orientation in each projected edgel that can be used in the cost function.

The algorithm for constructing a silhouette model is similar. We register dense point clouds that we obtain from a Kinect by using the same algorithm (ICP registration with the initial pose from the fiducial markers). The coordinate system origin is placed into the mass center of the joint point cloud.

In order to guarantee that poses that are close to each other are produced by close rotation and translation vectors, we place the coordinate system origin into the center of mass for each of the objects.

3.2 The Cost Function

The cost function is defined by comparing detected test image edges with projections of 3D surface and silhouette edges that depend on the object pose. Given a rotation and translation of the object, we transform the point cloud into the test camera reference frame. Surface edges are projected into the image and they give us 2D surface edges because transparent objects don't have self-occlusions. In order to get silhouette edges, we project a dense point cloud into a test image, apply several closing operations to the resulting set of pixels and find the borders of the connected components. These borders constitute silhouette edges.

Now we want to construct a cost function that compares two sets of edges in an image. Let $E = \{e_j\}$ be a set of pixels that belong to edges of a test image, $T = \{t_i\}$ is a set of the model points projected into the image plane. One of the most popular cost functions is Chamfer Matching (CM):

$$d_{CM}(E,T) = \frac{1}{|T|} \sum_{t_i \in T} \min_{e_j \in E} ||t_i - e_j||, \tag{1}$$

where $||\cdot||$ is Eucledian norm. However, mean is not a robust statistic because a single outlier can affect the final value severely. Edge detection is an unstable operation that produces a lot of variation, especially in edge endpoints. In order to overcome this issue we use the Partial Directed Hausdorff (PDH) distance [31]:

$$d_H(E,T) = K^{th}_{t_i \in T} \min_{e_j \in E} ||t_i - e_j|| \ . \tag{2}$$

Here $K^{th}_{t_i \in T}(\mathbf{X})$ is the Kth ranked value in the sorted set $\mathbf{X}$. Throughout the paper we use $K = 0.8|T|$, where $|T|$ is the number of elements in T. However, this distance can be set to zero by placing the object infinitely far away from the camera. It means the global minimum will be achieved in the incorrect pose for this distance. So we introduce a Normalized PDH (NPDH) distance:

$$d_H(E,T) = \frac{1}{\sqrt{\det C}} K^{th}_{t_i \in T} \min_{e_j \in E} ||t_i - e_j||, \tag{3}$$

where C is the covariance matrix of the projections of the point cloud into the image.

Both CM and PDH distances are known to behave incorrectly in clutter. Oriented Chamfer Matching (OCM) [10] is known to handle clutter better. However, it is more computationally expensive, so we use NPDH throughout the paper.

The PDH cost function is computed separately for surface and silhouette edgels. The resulting distances are added with different weights: 2/3 for surface and 1/3 for silhouette edges. The weight for surface edges is higher because surface edges are more stable: they are constructed by fusing edgels from many training frames, so we know that each surface edge is found robustly by the edge detector.

The cost function 3 is minimized by the global optimization algorithm DIRECT [32] from the NLopt library [33] by varying the 6 parameters defining pose of the object: a translation and rotation vectors.

4 Experimental Results

4.1 Transparent Objects

The algorithm was tested on the base of 5 transparent objects. We take 5 pairs of kitchen items and paint one object in each pair in white color to make it opaque because there are no reliable way to scan a transparent object [3]. We use the painted object to scan it with Kinect to create object models. Each training sequence contains 12 frames with different poses of the table relative to Kinect. The asymmetric circles pattern from the OpenCV library was used as the fiducial marker to estimate poses between frames. To create test data we used the corresponding transparent objects captured by a calibrated stereo pair of Canon EOS 40D cameras from a distance about 1 meter. Images were resized to resolution of about one megapixel (1166x778). Each test object is placed exactly as the corresponding training object relatively to the fiducial marker, so we know the ground truth.

The objective of these experiments it to investigate how accurate the initial guess about the object pose should be for the algorithm to produce a stable correct result. We ran the algorithm with many different initial guesses generated randomly. In particular, the correct pose was translated in random direction on the specified distance d and rotated in random direction on the specified angle α. Each experiment with specific values of d and α was repeated 50 times and

we take 27 different combinations of these values. All objects in the test base have rotation symmetry and this was taken into account when evaluating pose returned by the algorithm but this knowledge was not used by the algorithm itself.

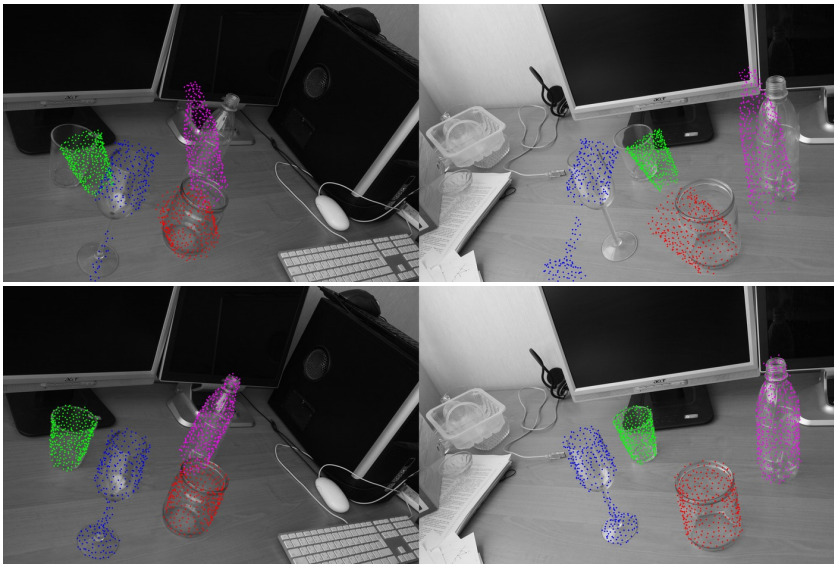


Fig. 3. Images from a stereo pair with the projected poses found with the algorithm, initial (upper row) and refined (bottom row)

The example of the results is given in Fig. 3. Points of objects' models are colored. They are projected into the image plane using initial hypothesis of objects' poses and poses refined by the algorithm. Initial poses are quite far away from correct poses. However, final poses are accurate enough for grasping.

We run the algorithm on all 5 objects to see how often the algorithm returns a correct pose. We consider the pose estimation successful if the difference between the returned and correct poses is less than 2 cm in translation and 10 degrees in rotation. Fig. 4 shows the statistics for all 5 objects. The percent of runs when the algorithm succeeded is plotted on the y-axis. The chart shows that if the initial translation error is less than 2 cm, we can successfully reconstruct the pose in more than 80% of the cases. Black area in Kinect depth map that corresponds to specular and transparent surfaces can give us a hypothesis about the object location. This information can be used to generate a good initial guess about the translation vector. If the initial error of the translation vector is 2cm, the rate of successful reconstructions (averaged over all angles) is 88%, if

the initial translation error is 5cm, then the rate of successful reconstructions is 77%. Note that part of the error comes from poses that are upside-down to the ground truth: since many objects are close to cylindrical shape, the final result can put the top of the glass to the bottom.

See more examples at Fig. 5. Also see Fig. 6 for example of the algorithm failure. The algorithm returned the pose which is upside down of correct one because the object has nearly cylindrical shape.

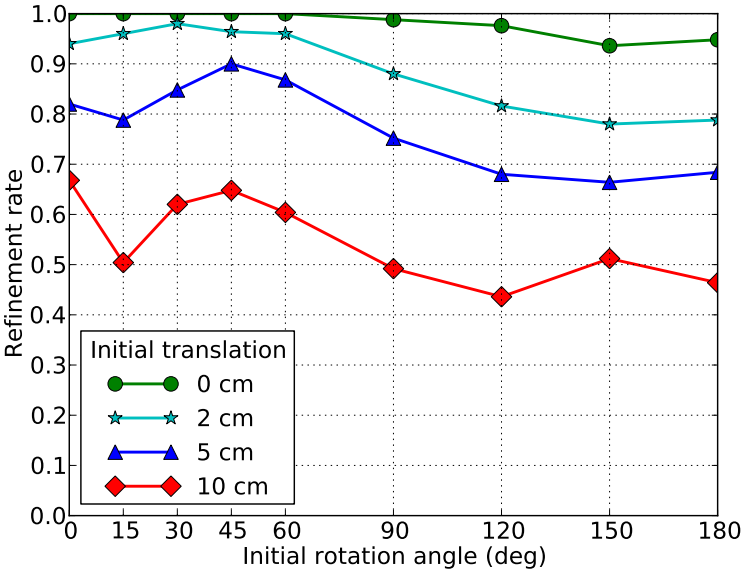


Fig. 4. Statistics of the algorithm working on rigid transparent objects. Pose can be refined successfully if an initial pose is not very far from the correct pose. The algorithm is robust to incorrect initial rotation but it is more sensitive to initial translation.

The proposed algorithm can refine poses of transparent objects in some cases, but it has several limitations. The approach demands good initial hypothesis of the object pose, otherwise the search for the global minimum takes too much time. The algorithm is unstable in clutter e.g. if the object is surrounded by other objects. But in the case of low clutter the algorithm works with sufficient speed and quality to be applied for pose refinement of rigid transparent objects.

Another limitation of the proposed method is using a calibrated stereo pair for generating test images instead of a single monocular camera. The main obstacle for a monocular camera is ambiguity that cannot be resolved from a single image without additional assumptions or priors.

There exist significantly different poses that have very good projections to a test image and it is specificity of transparent objects. For example, two different poses of an opaque object are shown in the Fig. 7 and there is no ambiguity

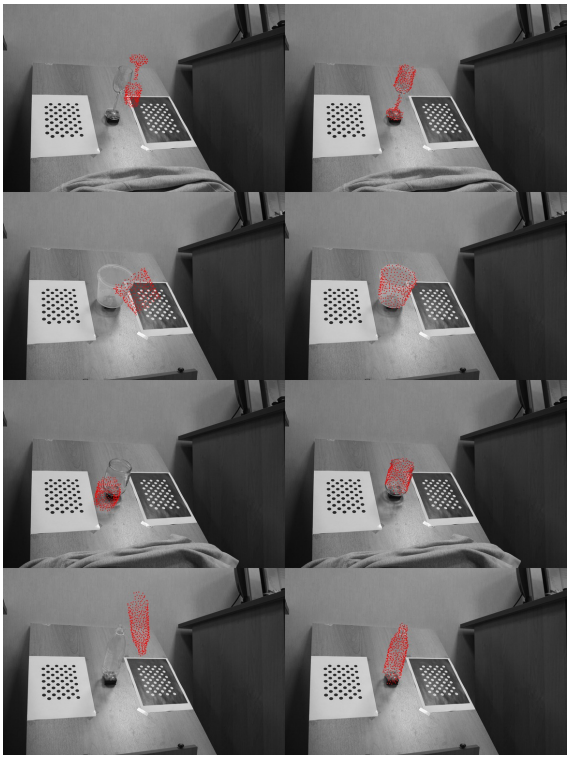


Fig. 5. Examples of successful pose refinement for different objects. Left images are initial poses and right images are refined poses. Only one image from the stereo pair is shown.

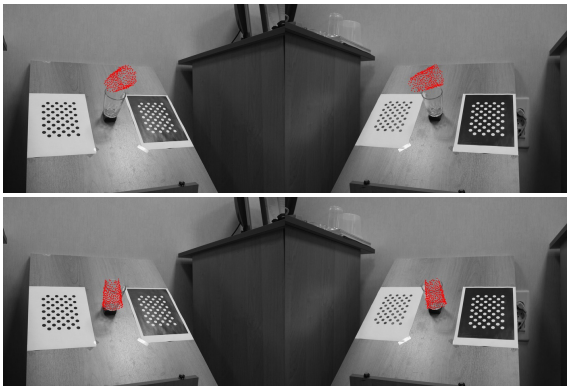


Fig. 6. Example of the algorithm failure due to cylindrical shape of the object. Two images from a stereo pair are shown: initial pose (upper row) and refined pose (lower row).

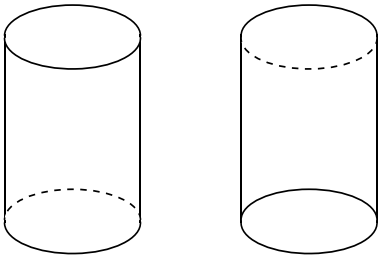


Fig. 7. Two different poses of an opaque object. There is no ambiguity between them because different edges are visible in different poses.

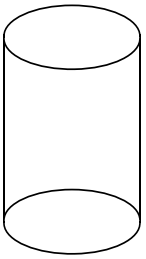


Fig. 8. Ambiguous projection of a transparent object. Two plausible poses of the object are possible because all edges are visible on the same image.

between them. However, if the same object is transparent then there are two different plausible interpretations of the same projection (Fig. 8) because transparent objects don't have self-occlusions and all edges are visible.

We evaluated the algorithm with a monocular camera on the same dataset using only left images of our stereo test set. The statistics of pose estimation is shown in the Fig. 9. One can see that there is a significant degradation of accuracy compared to the stereo case.

4.2 Textureless Objects

The proposed algorithm can be used for pose refinement of opaque objects too. An example of the algorithm working is shown in the Fig. 10. The corresponding poses of the object are used to project model points onto the images for visualization. You can see that the initial pose is far from correct because edges of the object are not projected onto edges of the image. However, the proposed algorithm refines this pose and gives accurate pose estimation suitable for robotic grasping.

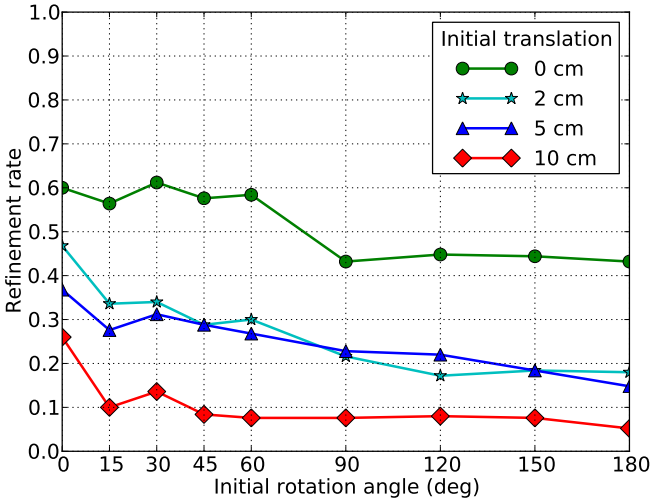
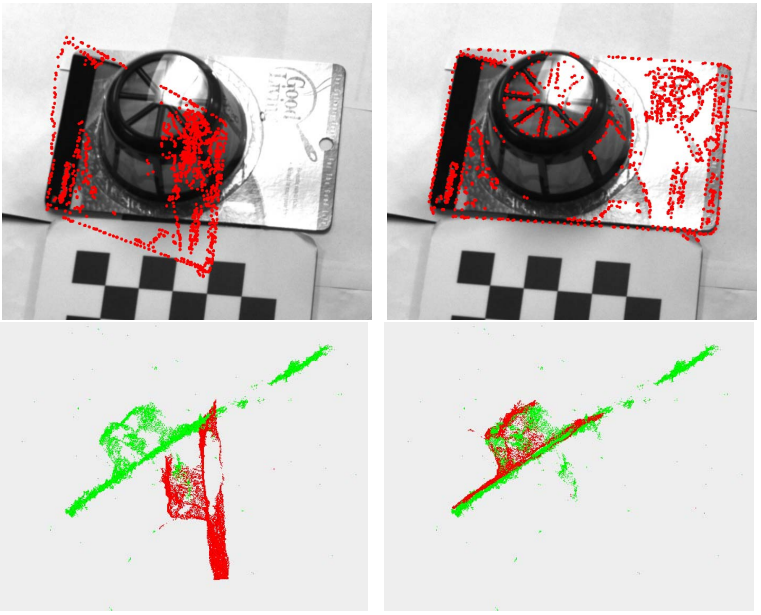


Fig. 9. Statistics of the algorithm working when using a monocular camera only. The results degrade significantly comparing to the stereo camera due to inherent ambiguity of pose estimation of transparent objects from a single view.



(a) Initial pose (b) Refined pose

Fig. 10. Initial and refined pose of an opaque object. The first row shows the surface model projected onto the images, the second row shows corresponding point clouds. The edges and the point cloud of the object are far from correct location initially but they move to correct position after the algorithm working.

5 Conclusion

The paper presents the algorithm for refining the 6-DOF pose of transparent objects. Our method only requires a calibrated stereo pair during the online stage. Given an initial estimate that has an error in translation less than 5cm, the rate of accurate pose estimations is higher than 75%. The method allows to grasp transparent objects without using expensive sensors such as TOF cameras.

Appendix: k-Partite Matching

Bioinformatics has an important problem of finding correspondences between protein-protein interaction networks [34]. This problem resembles our problem of point cloud registration because we need to find correspondences between points from different point clouds to refine locations of individual points. In bioinformatics this problem can be successfully solved by treating it as k-partite matching problem as proposed in [26]. However, in our case we have a lot of outliers and strong noise which creates significant difficulties for that algorithm. Based on [26] we developed a more suitable algorithm for our problem which uses robust statistics explicitly to be not affected by noise (Algorithm 1).

Algorithm 1. Creating a surface edge model

Require: k-partite graph G
Ensure: Edge model
  for all vertex v of the graph G **do**
    Find the nearest vertex to v in each part of the graph G
    Sort edges between v and found vertices by weight
    Compute sum of h edges with minimum weight as robust estimation of scatter
    Put v in a priority queue with priority equal to the computed sum
  end for
  while the priority queue is not empty **do**
    Take the top vertex from the queue
    Compute scatter estimation as before
    if scatter estimation is not changed **then**
      Compute robust estimation of a centroid using this vertex and the nearest vertices from different parts of the graph
      Add the centroid into the model
      Remove the vertex and these nearest neighbors from the queue
    else
      Put the vertex into the queue with updated estimation of scatter
    end if
  end while

References

1. Collet, A., Berenson, D., Srinivasa, S., Ferguson, D.: Object recognition and full pose registration from a single image for robotic manipulation. In: IEEE International Conference on Robotics and Automation (2009)
2. Rosenhahn, B., Brox, T., Weickert, J.: Three-dimensional shape knowledge for joint image segmentation and pose tracking. International Journal of Computer Vision 73(3), 243–262 (2007)
3. Ihrke, I., Kutulakos, K.N., Lensch, H.P.A., Magnor, M., Heidrich, W.: State of the Art in Transparent and Specular Object Reconstruction. In: STAR Proceedings of Eurographics, pp. 87–108 (2008)
4. Klank, U., Carton, D., Beetz, M.: Transparent Object Detection and Reconstruction on a Mobile Platform. In: IEEE International Conference on Robotics and Automation (2011)
5. Phillips, C., Derpanis, K., Daniilidis, K.: A Novel Stereoscopic Cue for Figure-Ground Segregation of Semi-Transparent Objects. In: 1st IEEE Workshop on Challenges and Opportunities in Robot Perception (2011)
6. Lysenkov, I., Eruhimov, V., Bradski, G.: Recognition and pose estimation of rigid transparent objects with a kinect sensor. In: Robotics: Science and Systems Conference (2012)
7. Lagger, P., Salzmann, M., Lepetit, V., Fua, P.: 3d pose refinement from reflections. In: IEEE Conference on Computer Vision and Pattern Recognition. IEEE (2008)
8. Chang, J., Raskar, R., Agrawal, A.: 3d pose estimation and segmentation using specular cues. In: IEEE Conference on Computer Vision and Pattern Recognition. IEEE (2009)
9. Netz, A., Osadchy, M.: Using specular highlights as pose invariant features for 2d-3d pose estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (2011)
10. Shotton, J., Blake, A., Cipolla, R.: Multiscale categorical object recognition using contour fragments. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1270–1281 (2007)
11. Biederman, I., Ju, G.: Surface versus edge-based determinants of visual recognition. Cognitive Psychology 20(1), 38–64 (1988)
12. Rosenhahn, B.: Pose estimation revisited. PhD thesis, Universität Kiel (September 2003)
13. Hinterstoisser, S., Lepetit, V., Ilic, S., Fua, P., Navab, N.: Dominant orientation templates for real-time detection of texture-less objects. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 2257–2264 (2010)
14. Gavrila, D.: A bayesian, exemplar-based approach to hierarchical shape matching. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1408–1421 (2007)
15. Reinbacher, C., Ruther, M., Bischof, H.: Pose estimation of known objects by efficient silhouette matching. In: IEEE International Conference on Pattern Recognition, pp. 1080–1083 (2010)
16. Lowe, D.: Three-dimensional object recognition from single two-dimensional images. Artificial Intelligence 31(3), 355–395 (1987)
17. Liu, M., Tuzel, O., Veeraraghavan, A., Chellappa, R., Agrawal, A., Okuda, H.: Pose estimation in heavy clutter using a multi-flash camera. In: IEEE International Conference on Robotics and Automation (2010)

18. Ulrich, M., Wiedemann, C., Steger, C.: CAD-based recognition of 3d objects in monocular images. In: International Conference on Robotics and Automation, vol. 1191 (2009)
19. Canny, J.: A computational approach to edge detection. Readings in Computer Vision: Issues, Problems, Principles, and Paradigms 184, 87–116 (1987)
20. Besl, P., McKay, N.: A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence (1992)
21. Zhang, Z.: Iterative point matching for registration of free-form curves and surfaces. International Journal of Computer Vision 13(2), 119–152 (1994)
22. Pulli, K.: Multiview registration for large data sets. In: IEEE Second International Conference on 3-D Digital Imaging and Modeling, pp. 160–168 (1999)
23. Bergevin, R., Soucy, M., Gagnon, H., Laurendeau, D.: Towards a general multi-view registration technique. IEEE Transactions on Pattern Analysis and Machine Intelligence 18(5), 540–547 (1996)
24. Fitzgibbon, A.: Robust registration of 2D and 3D point sets. Image and Vision Computing (2003)
25. Hazan, E., Safra, S., Schwartz, O.: On the hardness of approximating k-dimensional matching. In: Electronic Colloquium on Computational Complexity, TR03-020 (2003)
26. Singh, R., Xu, J., Berger, B.: Global alignment of multiple protein interaction networks. In: Proc. Pacific Symp. Biocomputing, vol. 13, pp. 303–314. Citeseer (2008)
27. Hubert, M., Rousseeuw, P., Van Aelst, S.: High-breakdown robust multivariate methods. Statistical Science 23(1), 92–119 (2008)
28. Rousseeuw, P.: Multivariate estimation with high breakdown point. Mathematical Statistics and Applications 8, 283–297 (1985)
29. Douglas, D., Peucker, T.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information and Geovisualization 10(2), 112–122 (1973)
30. Matas, J., Shao, Z., Kittler, J.: Estimation of curvature and tangent direction by median filtered differencing. In: Braccini, C., Vernazza, G., DeFloriani, L. (eds.) ICIAP 1995. LNCS, vol. 974, pp. 83–88. Springer, Heidelberg (1995)
31. Huttenlocher, D., Klanderman, G., Rucklidge, W.: Comparing images using the hausdorff distance. IEEE Transactions on Pattern Analysis and Machine Intelligence, 850–863 (1993)
32. Jones, D., Perttunen, C., Stuckman, B.: Lipschitzian optimization without the Lipschitz constant. Journal of Optimization Theory and Applications 79(1), 157–181 (1993)
33. Johnson, S.G.: The nlopt nonlinear-optimization package, `http://ab-initio.mit.edu/nlopt`
34. De Las Rivas, J., Fontanillo, C.: Protein–protein interactions essentials: key concepts to building and analyzing interactome networks. PLoS Computational Biology 6(6), e1000807 (2010)

Analysis of Space-Time Flow Structures by Optimization and Visualization Methods

Alexander Bondarev

Keldysh Institute for Applied Mathematics RAS, Russia, Moscow
bond@keldysh.ru

Abstract. This paper presents an approximate approach to analysis of space-time structures for unsteady problems in CFD (computational fluid dynamics). The approach is based on the solution of optimization problem combined with methods of data visual presentation. This approach is intended for fast approximate estimation of unsteady flow structures dependence on characteristic parameters (or determining parameters) in a certain class of problems. For some cases such approach allows to obtain the sought-for dependence in a quasi-analytical form. Having natural internal parallelism the approach is very suitable for parallel computations.

Keywords: space-time structures, optimization, inverse problems, visualization methods.

1 Introduction

Time-dependent processes in CFD problems are often accompanied by the presence of changeable space-time structures (STS) in the flow, such as separation zones, circulating flows, vortex bursts, etc. These structures cause many undesirable effects in practice: reduced lift, airframe and control vibrations. STS can appear and disappear defining the flow pattern and quantitative characteristics of the flow field. Simulating these changeable structures is therefore an important aspect of CFD.

Nowadays, modern computer hardware and mathematical methods allow one to simulate practically any time-dependent physical process in CFD and to obtain corresponding field of physical values. Calculating thoroughly the flow field one can obtain a beautiful picture of STS transformations. Nevertheless it is evidently insufficient for practical aims. In practice the phenomenon (physical effect) by itself is not the main point of interest. For practical engineering it is more interesting to know the circumstances leading to the phenomenon appearance, i.e. how this appearance depends on the problem characteristic parameters, such as Mach number, Reynolds number, Prandtl number etc. To define such dependence one should solve the problem in optimization statement, which is based on multiple calculations of the inverse problem.

This paper presents an approximate approach to analysis of space-time structures. This approach is intended for fast and rough estimation of unsteady flow

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XIX, LNCS 7870, pp. 158–168, 2013.

structures dependence on characteristic parameters in a certain class of problems. The approach is based on the solution of optimization problem combined with methods of data visual presentation. Visualization methods are applied to the solution of optimization problem. The solution is obtained in a form of multidimensional array. According to classification described in [1], such approach can be referred to *data analysis methods.*

There are two high-priority tasks for modern parallel computations: multidisciplinary problems and inverse problems. From this point of view the described below approach is very promising because it can be applied to a very wide range of time-dependent processes for various practical applications. Using the theory of optimization problems solution the approach is a modification of method [2] (parametric space analysis).

As it is shown below for concrete examples, this approach allows obtaining the sought-for dependencies in a form of quasi-analytical expressions.

2 Inverse Problems and CFD Applications

Numerical computation of the inverse problems in CFD is enough difficult. One should calculate 4D problems (3D+time) in a *variational statement.* It requires using high-performance computers. The separate difficulty is to visualize the inverse problems solutions for multidimensional case. There is a significant lack of concepts and tools in scientific visualization now. Nevertheless, the rapid development of computing technologies and hardware allows one to solve this class of numerical simulation problems.

A wide range of CFD problems can be solved using the concept and statement of inverse problems. Typically, the practical engineering problem is to choose the desired variants from the set of admissible variants. This can be the choice of a geometric shape (minimal drag), the choice of flow control (maximal mixing), etc. According to [3,4], the inverse problems are classified as boundary searches, coefficient searches, retrospective inverse problems, optimization problems. In general, for practical goals inverse problems are formulated as follows. One should find the determining parameters for which a phenomenon of interest occurs in a certain class of problems. It shouldn't depend on the details of the phenomenon appearance. It can be quantitative appearance of the phenomenon (some variable reaches definite value) or qualitative appearance (vortex formation, flow separation, any other STS changing).

Let's consider formalized statement of the inverse problem in a general form.

The numerical solution of the chosen CFD problem is elaborated during the computation process. The solution is defined by the finite set of determining parameters (or characteristic parameters) of the problem. These determining parameters can be divided into three main groups:

$A = (a_1, ..., a_n)$ - the parameters determining physical properties and mathematical model of problem;

$B = (b_1, ..., b_m)$ - the parameters determining the numerical method;

$C = (c_1, ..., c_l)$ - the parameters determining organization of the calculation process.

All these parameters form the numerical solution

$$F = F(A, B, C) = F(a_1, ..., a_n, b_1, ..., b_m, c_1, ..., c_l) \tag{1}$$

as a result of computation. So, the solution is based on the chosen mathematical model, numerical method and the way of calculation process organization.

We consider *the event functional* $J(F(A, B, C))$. Just as logical variable so this functional has two values:

$J(F(A, B, C)) = 1$ - if the event of interest occurs (independently on the event details), or

$J(F(A, B, C)) = 0$ - if the event of interest doesn't occur.

Presenting $J(F(A, B, C))$ in a form

$$J(F(A, B, C)) = J(a_1, ..., a_n, b_1, ..., b_m, c_1, ..., c_l) \tag{2}$$

one can formulate the inverse problem in a general form as follows. Find all the determining parameters $(a_1, ..., a_n, b_1, ..., b_m, c_1, ..., c_l)$ for which a phenomenon of interest occurs in a certain class of problems, i.e.

$$J(a_1, ..., a_n, b_1, ..., b_m, c_1, ..., c_l) = 1 \tag{3}$$

Considering the determining parameters $(a_1, ..., a_n, b_1, ..., b_m, c_1, ..., c_l)$ as a set of basis vectors, one can present the space of the determining parameters $L(a_1, ..., a_n, b_1, ..., b_m, c_1, ..., c_l)$ having $(n + m + l)$ dimensions.

Then for general case the inverse problem can be formulated as the problem of finding in this space L all the subdomains L^* where the event of interest is observed, i.e. $J(L^*) = 1$.

At the same time the problem of data filtration is solved. Setting the ranges for characteristic parameters one can not guarantee the fact of appearance of the sought-for event inside the range. So if the event does not occur for some point of space, this point is not considered.

3 Optimization Problem and Visualization

Using numerical or experimental modeling of unsteady phenomenon for practical goals in mechanics we usually know the reason of phenomenon appearance and quantitative parameter regulating this reason (control parameter)f^*_{cont}. The simulation is intended to define the control parameter dependencies on the determining parameters $(f_1, ..., f_n)$ of the problem. To obtain such dependencies $f^*_{cont}(f_1, ..., f_n)$ in a quasi-analytical or in a table form is a real practical goal of research. As a matter of fact these dependencies have been the main point of practical CFD applications last 50 years.

This paper considers a methodological approach to obtain these dependencies by means of numerical simulation. The approach can be described in general as follows.

Let's suppose that one has mathematical model of the CFD problem and reliable numerical method for solving. Then one can simulate some time-dependent

CFD process under investigation. During this simulation some event occurs. For instance, a new space-time flow structure appears. Such simulation is a computation of *direct problem*.

To study thoroughly the unsteady event one should solve the inverse problem with purpose to find the exact value of control parameter, when the event occurs.

For inverse problem solution one should multiply solve the direct problem varying the control parameter $f^*_{cont}(f_1, ..., f_n)$ until the onset of the unsteady phenomenon (physical effect) of interest. During this optimization process the set of determining parameters $(f_1, ..., f_n)$ is fixed. It is classic form of *inverse problem*. But it is the solution only for single point in multidimensional space created by determining parameters $(f_1, ..., f_n)$.

Then a set of grid points is specified for each determining parameter range. So the grid in multidimensional space is defined. The main point of interest is to find the solutions of inverse problem for each point of the grid. For this purpose the inverse problem should be solved multiply, for each point of grid. This procedure is a *parametric optimization analysis*. As a matter of fact it is a kind of standard parametric research where the inverse problem is considered for each point of parametric grid instead of direct problem.

The considered problem is a very difficult one. It requires a lot of computer resources. So it is quite natural to use parallel methods for this problem.

The problem of parametric optimization analysis has one serious advantage. It amounts to solving a set of similar small tasks. Each small task is the solution of inverse problem having a fixed set of determining parameters. So the approach of *multitask parallelism* can be applied. Using the principle "one task – one process" and having a minimal quantity of internal exchanges allows one to make very effective applications of this approach to practical problems. The possibility of rough grids using is another one serious advantage of the approach to be described.

The most effective and easy way to implement multitask parallel computations is to apply MPI (Message Passing Interface) technology [5]. Using MPI tools one can implement parallel computations according to the scheme presented in Fig. 1.

The control process creates the grid in the multidimensional space of determining parameters. Then the process forms the tasks and sends them to others processes and to itself. After task completion the control process collects the results in multidimensional array.

As a result one obtains control parameter dependence on determining parameters in general form of n-dimensional array $f^*_{cont}(f_1, ..., f_n)$.

This form is not suitable for practical goals. The most effective way of a search of the sought-for dependence in a quasi-analytical form is visual presentation of the array.

Analyzing the array one can decrease the dimensionality. For this purpose one should omit those determining parameters, which do not influence at the control parameter. If one has as a result $n \leq 3$ after such decreasing, then the rest of data can be visualized. For some cases the visualization is fast and effective

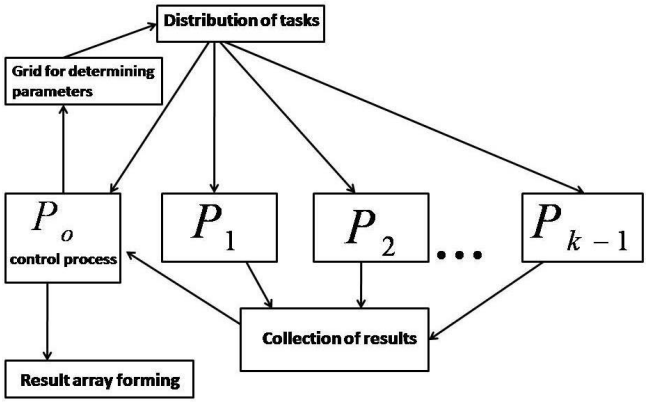


Fig. 1. Scheme of multitask parallel computations

way to obtain the dependence $f^*_{cont}(f_1, ..., f_n)$ in a form of quasi-analytical expression. To obtain such expressions it is assumed to approximate the array data (where it is possible) by simple geometric elements, such as lines, planes, parts of spheres etc.

It is very simple to do if we are dealing with the case of two determining parameters. We can approximate the surface $f^*_{cont}(f_1, f_2)$. For the case of three determining parameters we are able to build the isosurfaces. Then we can try approximating the isosurfaces by means of simple geometric elements. But for the case $n > 3$ there is an evident lack of concepts and tools for visual presentation. The creation of reliable and suitable for human acceptance visual presentations for this case is a subject for discussions.

By tradition the problem of multidimensional data visualization can be referred to the field of *Information Visualization* due to the fact that the solutions of such problems are necessary for business applications. There were some attempts to elaborate original visualization methods for multidimensional data. For instance, one can mention such approach as "Chernov's faces" [6]. The main idea of this approach is to present the values of different variables by various details of human face. Another attempt is presented in [7] for the space of events as "event tunnel". The space of events is presented in a form of 3D cylinder ("event tunnel"). Lengthwise axis presents the time; the events are presented as spheres inside the cylinder. As the distance from the point of observation grows, the sizes of spheres are reduced. Despite some success for business applications the artificial character of such visual concepts is evident.

So for common case one should try to decrease the array dimensionality up to 3 and hope the class of problems under consideration would allow such decreasing. Fortunately, for many real applications it is true, as it is shown below.

There are some well-known ways to decrease the array dimensionality.

The first way is the analysis of variances for each characteristic parameter. Characteristic parameter is considered as coordinate direction. Data variances $D_1, D_2, ...D_n$ are computed along the each direction. Then the variances should

be arranged. The direction with minimal variance $D_{\min} = \min_{i=1,n}\{D_i\}$ is rejected (compactification).

Another way is the construction of different 3D data projections for various triplets of determining parameters, as it is shown below for example of computations. If the data on projections for some direction are close to constant then this direction can be rejected.

Also it is very useful to apply PCA method (Principal Component Analysis) with purpose to decrease the number of dimensions. The principal component is a direction in multidimensional space with maximal data variance along. PCA method is based on localization of 3 principal components and data presentation using these components as new coordinate system. So this method provides the choice of a new orthogonal basis with coordinate directions corresponding to the variances arranged in descending order. For the case of data with complex topology PCA method is generalized. The generalized method is called principal manifolds method (or principal curves) [8].

Combining all these methods one can decrease the array dimensionality for many practical cases.

Described above approach combines the parallel computations of parametric optimization problem with and data analysis. The approach was applied to analysis of finite difference schemes also. This methodological approach was successfully used in [9] to optimize the computational properties of hybrid finite difference scheme applied to the solution of the supersonic far wake problem. Viscosity and turbulence were taken into account. The emergence of undesired oscillations was considered as the event to be controlled. The weight coefficient of finite difference scheme was used as the control parameter. The grid step, Mach and Reynolds numbers were chosen as determining parameters. As a result for chosen class of problems the weight parameter dependence on determining parameters was obtained in a quasi-analytical form.

4 Application Examples

The optimization approach is applied to practical analysis of unsteady circulating zones transformation. The problem of unsteady interaction of the supersonic viscous flow with jet obstacle is considered. This obstacle appears due to cocurrent underexpanded jet exhausting from the nozzle. The nozzle is placed to external supersonic viscous flow. Expanding jet propagates on the external surface of the nozzle and creates obstacle in external flowfield. The obstacle disturbs external flow and circulating zone appears ahead the obstacle. Typical flow structure is shown in Fig. 2 (a) by streamlines.

We consider a problem containing time-dependent boundary condition for underexpanded jet. Jet pressure ratio was set at the nozzle edge as time-dependent function $n = n(t) = P_a/P_\infty$ (where P_a - jet pressure, P_∞ - external flow pressure). The full system of time-dependent Navier-Stokes equations for viscous compressible heat-conductive flow is used as mathematical model. Implicit

hybrid finite difference WW-scheme is applied to solve the system of equations. This scheme has second order accuracy in time and space.

The dependence $n = n(t)$ is chosen as linear function. It allows one to set different rates of pressure ratio growth up to $n = 100$.

As a result of calculations of direct problem a new STS formation is obtained. Increasing the rate of pressure ratio growth one obtains new space-time structure in the vicinity of circulating zone ahead the jet obstacle. This new structure is shown by streamlines in Fig. 2 (b).

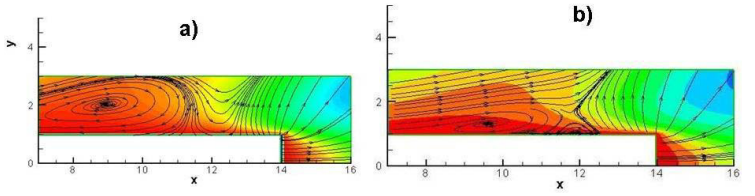


Fig. 2. Flow structure for slow pressure ratio growth (a) and for fast pressure ratio growth (b)

Let's consider the optimization approach application to analysis of unsteady event: the formation of the new space-time structure in the flow. The rate of pressure ratio growth is chosen as control parameter. The case of four determining parameters is considered. These four parameters are Mach number M_∞, Reynolds number Re_∞, Prandtl number - Pr_∞ and Sh_∞ - Strouhal number for the problem under consideration. For each fixed set of these numbers $(M_\infty, Re_\infty, \mathrm{Pr}_\infty, Sh_\infty)$ the inverse problem is solved by varying pressure ratio growth rate until the onset of the new structure formation in the flow. These characteristic numbers vary in ranges: $1.5 \le M_\infty \le 3$; $2.5 \le \lg Re_\infty \le 4$; $0.72 \le \mathrm{Pr}_\infty \le 1$; $1 \le Sh_\infty \le 2$. For each new fixed set of numbers $(M_\infty, Re_\infty, \mathrm{Pr}_\infty, Sh_\infty)$ the procedure described above is repeated.

So for each set of determining parameters $(M_\infty, Re_\infty, \mathrm{Pr}_\infty, Sh_\infty)$ one defines the exact values for crucial velocity of pressure ratio growth V^* and characteristic time t^* when the new flow structure appears. As event characteristic time $t^* = t_{ev}/t_{n=100}$ is chosen, here t_{ev} - the time when the event occurs and $t_{n=100}$ - the time when pressure ratio reaches 100.

According to the scheme presented in the previous chapter parallel algorithm is elaborated for computations. Two types of grids are chosen: 5 and 10 points for each determining parameter. It requires computing 625 and 10000 inverse problems. The computations are performed by parallel cluster K100 (Keldysh Institute of Applied Mathematics RAS, Moscow, Russia). MPI technology is applied to control parallel computations.

As a result of approach application four-dimensional arrays are obtained. These arrays contain numerical presentations of the characteristic time t^* and crucial velocity V^* dependencies on four determining parameters $(M_\infty, Re_\infty, \mathrm{Pr}_\infty, Sh_\infty)$.

According the previous chapter some methods are applied to decrease the number of dimensions.

The analysis of variances for each characteristic parameter shows the fact the variance along direction Re_∞ is much smaller than the others, i.e. $D_{\lg Re_\infty} << D_{M_\infty}, D_{Sh_\infty}, D_{\mathrm{Pr}_\infty}$. One can assume the determining parameter Re_∞ does not influence on the solution.

The construction of different 3D data projections for various triplets of determining parameters confirms this assumption.

The dependencies $V^* = V^*(M_\infty, \lg Re_\infty, \mathrm{Pr}_\infty)$ and $t^* = t^*(M_\infty, \lg Re_\infty, \mathrm{Pr}_\infty)$ are presented in Fig. 3 and Fig. 4 by isosurfaces. These isosurfaces show that characteristic time and crucial velocity does not depend on Reynolds number for chosen laminar diapason ofRe_∞.

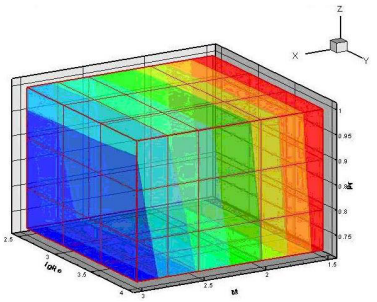


Fig. 3. Crucial velocity dependence on Mach, Reynolds and Prandtl numbers

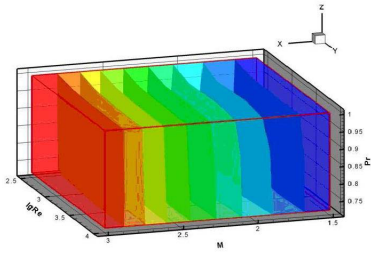


Fig. 4. Characteristic time dependence on Mach, Reynolds and Prandtl numbers

Further, the dimensionality of array can be decreased and one is able to consider 3D arrays $V^* = V^*(M_\infty, \mathrm{Pr}_\infty, Sh_\infty)$ and $t^* = t^*(M_\infty, \mathrm{Pr}_\infty, Sh_\infty)$. These dependencies are shown in Fig. 5 and Fig. 6 by isosurfaces.

Also PCA method is applied to four-dimensional array containing crucial velocity data. The presentation of crucial velocity in coordinates of principal components is shown in Fig. 7.

Using all these results and visual presentations one can make two main conclusions:

1) The dimensionality of four-dimensional array under consideration can be decreased up to 3, because characteristic parameter Re_∞ has a very small influence on the solution.

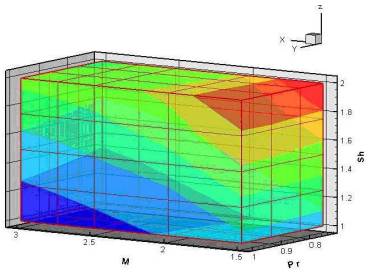


Fig. 5. Crucial velocity dependence on Mach, Prandtl and Strouhal numbers

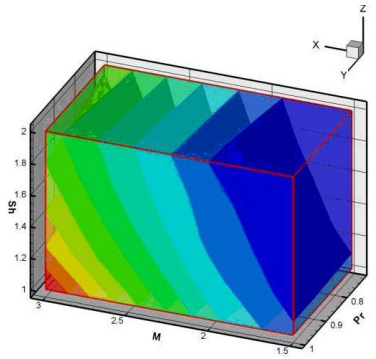


Fig. 6. Characteristic time dependence on Mach, Prandtl and Strouhal numbers

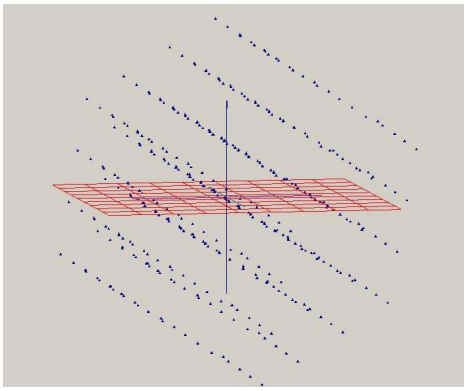


Fig. 7. Data presentation for crucial velocity in principal components

2) Analyzing the view of these visual presentations one can approximate the isosurfaces by planes. For the purpose of rough estimation the sought-for dependence can be written in a form of plane

$$AM_\infty + B\,\underset{\infty}{\mathrm{Pr}} + CSh_\infty = const.$$

It allows one to get average estimation of V^* and t^* dependencies on determining parameters as

$$V^* = V^*(M_\infty, \underset{\infty}{\mathrm{Pr}}, Sh_\infty) = -0.1M_\infty + 0.115\,\underset{\infty}{\mathrm{Pr}} + 0.24Sh_\infty \tag{4}$$

$$t^* = t^*(M_\infty, \underset{\infty}{\mathrm{Pr}}, Sh_\infty) = 0.224M_\infty - 0.04\,\underset{\infty}{\mathrm{Pr}} - 0.132Sh_\infty \tag{5}$$

These expressions describe the connection between control parameter (pressure ratio growth rate) and characteristic parameters – Mach, Reynolds and Strouhal numbers.

The way to check the accuracy of such approach is simple enough. For this purpose one should specify characteristic numbers and the rate of pressure ratio growth and then one should carry out the computation of the straight problem. The resulting space-time structure can be compared with the structure corresponding to approximate expression.

For the problem under consideration such estimation of accuracy is computed also. Maximal deviation from the approximating plane for crucial velocity does not exceed 14 percents.

5 Dicussions

In this chapter we would like to discuss the field where the described approach could be efficiently implemented.

As is known, computer simulation of 3D time-dependent CFD process is a very difficult task. The simulation requires a lot of computer resources, especially for thorough grids. Taking this into account, the implementation of parametric research for such simulation is almost impossible. Even parallel computer systems can not help much. One can find an example of this situation in [10]. In this paper the authors describe the simulations of 3D time-dependent CFD complex problems, such as the search of separation zones in the flows, internal CFD problems for engines, etc. The authors note the fact that even for direct problem solution the computation of Navier-Stokes equations with modern parallel cluster can take up to two weeks.

For such problems the researcher can not know in advance the point in the space of determining parameters where the computation would be effective. So the set of determining parameters for direct problem is chosen by researcher basing on experience. The inaccuracy of this choice leads to the loss of computer time.

At the same time these difficulties can be avoided. A simple strategy of computing can be applied with the help of preliminary optimization approach.

At the first stage of this strategy proposed in this paper approach is applied for rough grids. It allows obtaining important points in the space of determining parameters. Also in this space can be defined hidden dependencies.

At the second stage one can carry out the computation of direct problem for thorough grid. The fixed set of determining parameters for the problem is set in accordance with results of the previous stage of computation.

6 Conclusions

The optimization approach to unsteady processes analysis is considered. The approach allows carrying out fast and effective estimation of the way how the crucial points of flow structure transformation depend on determining parameters of the problem. Combining the inverse problems solutions with the visual presentations of such solutions for many real cases allows one to obtain the sought-for dependencies in a quasi-analytical form. The approach can be applied for rough grids. The approach amounts to solving a set of similar small tasks, so it can be applied also for parallel computations. The approach can be applied to a wide range of time-dependent processes for various practical applications.

Acknowledgements. This work is supported by RFBR (project number 13-01-00367A).

References

1. Bondarev, A.E., Galaktionov, V.A., Chechetkin, V.M.: Analysis of the development concepts and methods of visual data representation in computational physics. Computational Mathematics and Mathematical Physics 51(4), 624–636 (2011)
2. Sobol, I.M., Kartyshov, S.V., Kulchickaya, I.A., Levitan, Y.L.: Multicriterial optimization of mathematical models. Matematicheskoe Modelirovanie (6), 85–93 (1994) (in Russian)
3. Alifanov, O.M.: Inverse Problems for Heat Transfer. Mashinostroenie, Moscow (1988) (in Russian)
4. Beck, J.V., Blackwell, B., St.Clair, C.: Inverse Heat Conduction. Ill-posed Problems. John Wiley&Sons, USA (1985)
5. Pacheco, P.: Programming Parallel with MPI. Morgan Kaufmann, San Francisco (1997)
6. Savasere, A., Omiecinski, E., Navathe, S.: An Efficient Algorithm for Mining Association Rules in Large Databases. In: Proc. 21st Int'l. Conf. Very Large Data Bases, pp. 432–444. Morgan Kaufmann, San Francisco (1995)
7. Suntinger, M., Obweger, H., Schiefer, J., Gröller, M.E.: Event Tunnel: Exploring Event-Driven Business Processes. IEEE Computer Graphics and Applications 28(5), 46–55 (2008)
8. Gorban, A., Kegl, B., Wunsch, D., Zinovyev, A. (eds.): Principal Manifolds for Data Visualization and Dimension Reduction. LNCSE, vol. 58. Springer, Heidelberg (2007)
9. Bondarev, A.E.: Optimizing Hybrid Finite Difference Scheme with regard to Viscosity and Turbulence on the Base of Inverse Problems. In: Proc. of Conf. "High-Performance Computations in Mechanics and Physics", Moscow, pp. 39–44 (2009) (in Russian)
10. Shalaev, V.I., et al.: Application of parallel computer technologies to complex CFD problems in "VC FALT MFTI". Supercomputers 10(2), 59–64 (2012) (in Russian)

Author Index