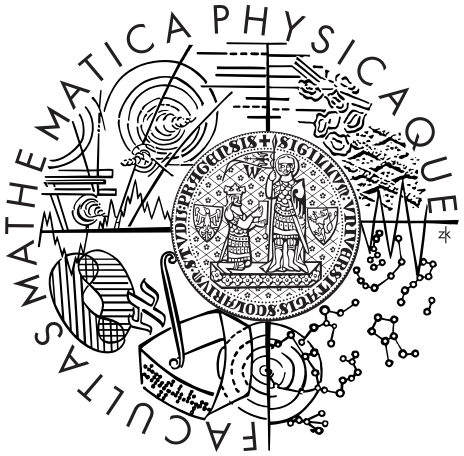
Charles University in Prague
Faculty of Mathematics and Physics

BACHELOR THESIS



Viktorie Vášová

3D Computer Vision on the Android Platform

Name of the department or institute

Supervisor of the bachelor thesis: Mgr. Lukáš Mach

Study programme: General Computer Science

Prague 2013

Dedication.

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date ..... signature of the author

Název práce: 3D počítačové vidění pro platformu Android

Autor: Viktorie Vášová

Katedra: Informatický ústav Univerzity Karlovy

Vedoucí bakalářské práce: Mgr. Lukáš Mach

Abstrakt:

Klíčová slova: 3d počítačové vidění, problém korespondence, platforma Android

Title: 3D Computer Vision on the Android Platform

Author: Viktorie Vášová

Department: Computer Science Institute of Charles University

Supervisor: Mgr. Lukáš Mach

Abstract:

Keywords: 3D computer vision, image correspondence, Android platform

Contents

Introduction

In the recent years, many researchers have been interested by the task of Computer Vision. Particularly the problem of 3D reconstruction is being investigated a lot. At this moment there is a great number of algorithms to solve problems in this area. Most of the approaches depend on the kind of input that is available (a set of pictures – determining is also how many pictures are taken, a video stream, etc.) and also on the output that we expect.

The work of many researches resulted in several online applications such as PhotoSynth by Microsoft or Google StreetView that is used by millions of people nowadays.

Meanwhile we could observe a large progress of telecommunication devices. In recent years, it became very common to own a smartphone. Especially mobile phones with Android platform are very popular. A built–in camera and large amount of applications is typical for such kind of telephone.

The goal of this work is to explore ways how to connect these two phenomena and to create an Android application that takes a set of pictures and visualizes the result of the reconstruction of the depth information. Due to the ambiguity of solving the task, we have to be aware of the fact that it is possible, that our application will be limited to only particular type of scenes.

The first part of this work analyses the problem, describes available software and gives an overview of programming libraries and languages that were used. Secondly, we focus on the theoretical basics and introduced approaches connected to this topic. The next section is denoted to the implementation of our application and finally we evaluate and benchmark our work.

1. Overview

Many years of researches resulted in several applications. In this part we will give an overview of available software dealing with the analysis of depth information and 3D reconstruction. In the second part of this chapter, available programming languages and libraries considered for our work, are discussed.

1.1 Currently Available Software

One of the first applications that were used to create a 3D model from a set of pictures of an object was Photosynth designed by Microsoft in cooperation with University of Washington. The algorithm is based on pattern recognition and generates a 3D model of a photographed object including the point cloud. After releasing the application there were available only projects generated by Microsoft or BBC and later a cooperation with NASA was started. Until two years later the version for public was released so users could upload own images to create a 3D model.

In 2007, a year after releasing Photosynth, Google introduced Street View to extend Google Maps and Google Earth. At first, this additional application provided a panoramic views of cities in the USA, but soon it expanded to other places in the world.

Autodesk, an American corporation focused on 3D design software, released modelling application Autodesk 123D recently. There are several additional tools available. One of them is 123D Catch that creates 3D model from a set of pictures taken from different view angles. This software is compatible with Autodesk 123D application, so it is advisable idea for designers who want to work with real–world objects in the virtual scene. The program is available for these operating systems: Windows, Mac OSX and iOS. It seems that for creating such a 3D model it is necessary to follow detailed instructions how to shot the pictures. In most cases the process of building model fails because of wrong set of images. An error can occur when pictures are blurred, the background is not solid or the amount of photos is not sufficient.

If we evaluate accessibility of the software for mobile phones, Google Street View is running on every type of mobile platform without any larger errors. There is a version of Photosynth for Windows phones and iOS operating system. As we already mentioned, Autodesk developed a version for iOS as well. But apparently, we miss applications developed for Android platform.

1.2 Currently Available Libraries

The problem considered in this work is getting an image information from a set of pictures and its processing afterwards. To be able to program a software dealing with a task from the area of computer vision, it is necessary to be familiar with a library that supports work with images. That kind of functions offers OpenCV library.

OpenCV is a cross–platform library developed by Intel. It provides large scale

of functions supporting image processing; classes for segmentation and recognition, blob detection and other 2D and 3D feature toolkits are available.

For our work is important that support for C, C++, Python and the Android platform is included in the library. OpenCV4Android offers us great equipment for image processing.

2. Related Work

In this section we mention some of the approaches that have been proposed earlier. We consider algorithms for interest points detection and description especially. It gives us an overview of already existing work.

2.1 Interest Point Detection

In 1988, Chris Harris and Mike Stephens [2] published their corner detector based on combining corner and edge detector. Although this approach is not scale–invariant, it is widely used nowadays.

However, many computer vision tasks deal with the real world input data images where objects appear in different ways depending on the scale. Due to this fact, Lindberg came out with automatic scale selection for feature detection [5].

There are several other approaches that have been introduced such as edge–based region detector by Jurie and Schmid [3] or salient region detector by Kadir and Brady [4]. But apparently they are not used that much. It seems that due better results and stability, most popular are Hessian–based detectors.

Another example of Hessian–based detector is a detector published by Herbert Bay et al. in Speeded–Up Robust Features [1]. This detector is partly inspired by SIFT algorithm, but it is several times faster and more robust against the scale transformation and the image rotation. It relies on the approximation of determinant of the Hessian matrix that can be computed using integral images. Hence, the computation is very fast (see the previous chapter).

2.2 Interest Point Description

Large number of interest point descriptors were introduced. The most common one is scale–invariant feature transform [6], SIFT for short. This algorithm was published in 1999 by a Canadian computer scientist David G. Lowe. The descriptor is computed from the intensities around the key point locations where the local gradient direction of picture intensities gives description of the local key point.

3. Used Mathematical Terminology

To make this work more integrated, we will briefly remind some mathematical concepts that are used in following chapters.

3.1 Hessian matrix

Many feature detectors and interest points descriptors use the Hessian matrix. The Hessian is a square matrix of the second order partial derivatives of a function. Assuming all second partial derivatives exist, for a given function $f(x_1, x_2, \ldots, x_n)$ the Hessian looks

$$H(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ & & \cdots & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

3.2 Integral images

Later in this work we work with the term of integral images. An integral image (also known as summed area table) allows fast and efficient computation over a rectangular area in an image. Each pixel represents the sum of all previous pixels above to the left. To be more formal, let's see the mathematical formula describing the value of a pixel, where I is an integral image and $\mathbf{x} = (x, y)^T$ is a location of a current pixel.

$$I(\mathbf{x}) = \sum_{i=0}^{i<x} \sum_{j=0}^{j<x} I(i, j)$$

(3.1)

An advantage of an integral image is that we are able to construct it with only one pass over the original image. Moreover, once we have calculated the integral image, it requires only three integer operations and four memory accesses to calculate the sum of intensities inside the region of any size in the picture (see figure 3.1).

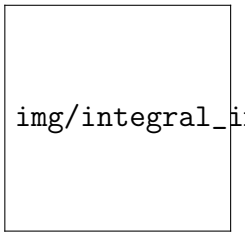


Figure 3.1: The sum of any rectangular region can be achieved by only three additions.

4. Image processing algorithms

In the previous chapter we gave an overview of related work and algorithms that were introduced. This chapter is concentrated on some of these approaches in greater detail. We will describe three techniques – SIFT and SURF algorithm for keypoints detection and then Min-cut/Max-flow graph algorithm. SIFT and SURF are the most used algorithms to detect features. Both of them are robust to scale and rotation.

Generally, feature matching algorithms first extract keypoints from a pair of photos of the same scene. If the feature detection is repeatable, most of the features should be detected in both images. Then, for each feature a descriptor is generated, typically it is a high-dimensional vector. Usually, a descriptor is affine invariant. Hence for a feature from different viewpoints should be generated the same vector. Since the descriptors are computed, we can match them between a pair of images. Due to the fact that our descriptors are vectors, we can match them according to the distance in the space, using Euclidean distance for example.

4.1 The SIFT algorithm

Scale-invariant feature transform (SIFT) was published already in 1999 by David Lowe [6]. Until now it is still very popular algorithm to generate matches for a pair of pictures. It is able to identify objects even among clutter in noisy images. SIFT feature descriptor is 128-dimensional vector invariant to scaling and orientation but only partially invariant to affine distortion and illumination changes. Hence the best results are given on the input images that do not differ much in the orientation.

The feature points are detected from the grayscale version of an input image matching centres of blob-like structures. To each feature point is assigned an information about local orientation and scale and based on these data is the descriptor constructed.

First step to reach the set of keypoints is creating a scale-space of an image. A scale-space is a series of blurred images derived from convolution of the image with Gaussian filter. The original image is blurred several times with larger and larger Gaussian kernel, then it is downsized and this process is applied again to the subsample. Since we have a scale-space, we subtract neighbouring images and like this generate a set of differences of Gaussians (DoG). A DoG image can be formulated as

$$D(x, y) = G_{\sigma_1} * f(x, y) - G_{\sigma_2} * f(x, y)$$

where $f(x, y)$ is the original image, $*$ presents convolution, σ_i is the scale and G_{σ_i} is Gaussian kernel. Hence

$$D(x,y) = \left[\left(\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(-\frac{x^2+y^2}{2\sigma_1^2}\right)\right) - \left(\frac{1}{\sqrt{2\pi\sigma_2^2}}\exp\left(-\frac{x^2+y^2}{2\sigma_2^2}\right)\right)\right] * f(x,y)$$

These convolved images are grouped by octave, where an octave corresponds to doubling value of scale σ.

Now we can define candidate keypoints as maxima and minima of the result of difference of Gaussian that occur at multiple scales. That are exactly the areas we are looking for - spots of high contrast. This is done by comparing each pixel with its neighbouring pixels and also with corresponding pixels in neighbouring DoG image. If the pixel value is the largest or the lowest among all compared pixels, it is selected as a candidate feature point.

Usually, the detection of scale-space extrema results is a large amount of candidate keypoints. However, some of them are not stable and they need to be discarded. For each candidate we do the interpolation of nearby data to estimate its position. It is done using quadratic Taylor expansion of the Difference-of-Gaussian function around point:

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x^T}\frac{\partial^2 D}{\partial \mathbf{x^2}}\mathbf{x}$$

where $\mathbf{x} = (x, y, \sigma)$ is the offset from the point. If the offset value is less than 0.03, it indicates low contrast and a high probability of affection by noise or of corresponding to edges and the candidate is discarded. Otherwise we keep the keypoint.

The next step is to describe the nearby area of each keypoint to generate the descriptor. To achieve the invariance to rotation, the value of the gradient that establishes local orientation and scale of the keypoint is used. Firstly we compute gradients for pixels around the location of the keypoint. The nearby area is divided to 4x4 subregions and for each subregion is computed a histogram of 8 gradient values. The peaks of the histogram presents the dominant orientations. The orientations corresponding to the highest peak or peaks that are within 80% of the highest peak are assigned to the feature.

We create the descriptor using the gradient magnitudes of these regions. For a feature there are sixteen subregions, each with eight values of a histogram, that results in a 128-dimensional vector.

All extracted keypoints are stored in a database; the last step is matching them. Typically, we do this by taking one feature and looking for the nearest (in the meaning of Euclidian distance) vector descriptor in the second image.

4.2 The SURF algorithm

Another popular algorithm detecting features is SURF (Speeded Up Robust Features). It was presented by Herbert Bay et al. [1] in the year 2006 and it is partially inspired by SIFT described above. SURF is based on computation of Haar wavelet responses, Hessian-matrix approximation and using integral images.

To build a robust descriptor we need to detect keypoints that are invariant to scale thus we create a scale-space. That is because often is required to detect objects in different distances. A scale-space is usually implemented as an pyramid of subsequently smoothed images by Gaussian filter as was described in previous section concerned with SIFT algorithm. In this approach integral images are used, see previous chapter to get detailed information about integral images. So we do not have to iteratively apply the same filter, but we can reach the scaling by fast computing of integral image where the speed is independent to the size of filter.

The detection of interest points is done by using basic Hessian-matrix and its approximation. The Hessian for a point $\mathbf{x} = (x, y)$ in an image I at scale σ is defined as

$$H(x, \sigma) = \begin{pmatrix} L_{xx}(x, \sigma) L_{xy}(x, \sigma) \\ L_{yx}(x, \sigma) L_{yy}(x, \sigma) \end{pmatrix}$$

where L_{xx} presents the convolution of the Gaussian second order derivative with the image:

$$L_{xx}(x, \sigma) = \frac{\partial^2}{\partial x^2} g(\sigma) * I$$

and similarly for L_{xy} and L_{yy}

$$L_{xy}(x, \sigma) = \frac{\partial^2}{\partial x \partial y} g(\sigma) * I, L_{xx}(x, \sigma) = \frac{\partial^2}{\partial y^2} g(\sigma) * I.$$

Since the Gaussian is considered as over-rated, in this approach we use box-filters instead and obtain speeded-up variation. Box filter approximation of Gaussian can be computed very quickly using integral images. Size of a box filter correspond to the scale size of Gaussian, furthermore due to way of computing integral images, the calculation time is independent to the scale size. These approximations are computed for every scale of the scale-space and those points that simultaneously are local extrema of both the determinant and trace of the Hessian matrix are chosen as candidate keypoints. The trace of Hessian matrix is identical to the Laplacian of Gaussians (LoG). If we denote those approximating box filters as D_{xx}, D_{yy} and D_{xy}, we get the determinant value

$$det(H) = D_{xx} D_{yy} - \omega^2 D_{xy}^2.$$

SURF descriptor is a high-dimensional vector consisting of the sum of the Haar wavelet response around the point of interest. The vector describes the distribution of the intensity within the neighbourhood of the keypoint. At first, to be invariant to image rotation, for a feature dominant orientation is established. The Haar wavelet responses in x and y direction are calculated and they are

summed within a sliding orientation window of size $\frac{\pi}{3}$ afterwards. Again, the responses can be computed with the aid of the integral image. Summed responses then yield a local orientation vector. The longest such vector lends the orientation of the feature.

The first step to extract the descriptor is constructing a square region around the keypoint aligned with dominant orientation. The size of the square area is $20s$, where s is the scale where the keypoint was detected. We split the window into 4 x 4 regular square subregions and for each of them we compute Haar wavelet responses d_x and d_y at 5 x 5 evenly spaced sample points inside. The sum of the responses for d_x and d_y and their absolute values $|d_x|$, $|d_y|$ separately results in a descriptor $\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$ for each subregion. Concatenating this for all 4 x 4 subregions gives 64-dimensional vector descriptor.

Matching SURF descriptors between two images can be speeded-up by including the trace of Hessian (Laplacian) to the process of finding corresponding keypoints. We exploit the sign of Laplacian that distinguishes bright blobs on dark background from the reverse situation. In the matching stage we compare only features if they have the same sign – type of contrast. This costs no more time for computation since we already computed it during the previous stage.

4.3 SIFT vs. SURF

Both SIFT and SURF are popular feature descriptors based on the distribution of intensity around the interest point. SIFT used to be evaluated as the most robust and distinctive descriptor for feature matching. However, SURF has been shown to have similar performance to SIFT, but being much faster. The results of research in [1] point out the fast computation of a descriptor. SURF describes image three times faster than SIFT. The key to the speed is using integral images. Also, the SURF descriptor is 64-dimensional vector in the comparison to 128 integers describing SIFT feature. Hence the SIFT matching costs more calculation time. Furthermore, due to the global integration of SURF-descriptor, it stays more robust to various perturbations than the SIFT descriptor.

SURF offers invariance to rotation, handling image blurring, but SIFT is still more robust in illumination changes and viewpoint changes.

4.4 Maximum Flow Graph Cut Algorithm

In computer vision there is a general task of stereo correspondence. It is a problem of discovering the closest match between points of two pictures typically taken from different positions. Presumably, rectification of the images considerably simplifies the situation. Sometimes this constraint can be done. Once the correspondences are solved, we obtain a disparity map and it can be exploited to reconstruct the positions and distances of the cameras.

Generally, there are two approaches to find the corresponding points. One of them is to detect features and interest points in an image and then find corresponding points in the other one. Only these high distinctiveness points are matched, hence it is called sparse stereo matching. The second way, dense stereo matching, is to match as many pixels as possible.

In 1999 Sébastien Roy published an algorithm [7] [8] to solve the stereo correspondence problem formulated as a maxim-flow problem in a graph. This approach does not use of epipolar geometry as many other do. It solves efficiently maximum-flow and gives a coherent minimum-cut that presents a disparity surface for the whole image. This way provides more accurate depth map then the classic algorithms and the result is computed in shorter time in addition.

5. Developing for Android

5.1

5.2

6. OpenCV

6.1 OpenCV4Android

6.2

7. Implementation

7.1

7.2

Conclusion

Bibliography

[1] H.Bay, A. Ess, T. Tuytelaars, and L. Van Gool. *Speeded–Up Robust Features (SURF).* In ECCV, 2006.

[2] C. Harris and M. Stephens. *A combined corner and edge detector.* In Proceedings of the Alvey Vision Conference, pages 147 – 151, 1988.

[3] F. Jurie and C. Schmid. *Scale–invariant shape features for recognition of object categories.* In CVPR, volume II, pages 90 – 96, 2004.

[4] T. Kadir and M. Brady. *Scale, saliency and image description.* IJCV, 45(2):83 – 105, 2001.

[5] T. Lindeberg. *Feature detection with automatic scale selection.* In International Journal of Computer Vision, 30(2):79 – 116, 1998.

[6] D. Lowe. *Object recognition from scale–invariant features.* In CCV, 1999.

[7] S. Roy. *A Maximum-Flow Formulation of the N-camera Stereo Correspondence Problem.* Computer Vision, 1998. Sixth International Conference on.

[8] S. Roy. *Stereo Without Epipolar Lines: A Maximum-Flow Formulation.* In International Journal of Computer Vision, 1999.

List of Tables

List of Abbreviations

Attachments