REPUBLIKA SLOVENIJA
MINISTRSTVO ZA IZOBRAŽEVANJE, ZNANOST IN ŠPORT

JAVNI SKLAD REPUBLIKE SLOVENIJE ZA RAZVOJ KADROV IN ŠTIPENDIJE

Naložba v vašo prihodnost
OPERACIJO DELNO FINANCIRA EVROPSKA UNIJA
Evropski socialni sklad

A creative way to practical knowledge

The study and comparison of advanced computer vision methods for modeling feet in real-world environments

TEHNICAL REPORT LUVSS-4/2014
LJUBLJANA, 2014

Authors: Domen Rački[1], Matjaž Majnik[1], Sandi Gec[1], Matjaž Hegedič[1], Kristjan Žarn[1], Matej Kristan[1], Janez Perš[2], Damir Omrčen[3], Danijel Skočaj[1]

[1]*University of Ljubljana, Faculty of Computer and Information Science*
[2]*University of Ljubljana, Faculty of Electrical Engineering*
[3]*UCS d.o.o., Sinja Gorica 106B, 1360 Vrhnika*

University of Ljubljana
Faculty of Computer and Information Science

The project is partly funded by the European Union through the European Social Fund. The project is implemented under the Operational Programme for Human Resources Development for the period 2007-2013, Priority axis 1 ”Promoting entrepreneurship and adaptability, and priority guidelines” 1.3. ”Scholarship Scheme”, within the approved operations ”A creative way to practical knowledge”.

Contents

Chapter 1

Introduction

In recent years the sales of footwear over the Internet has experienced significant growth. The key drawback of buying footwear over the internet is choosing the right shoe size without the ability of physically trying the footwear. The company UCS has developed a recommendation system for advising in footwear purchasing, but the system itself is restrained by its need for optical foot measuring hardware, which is only available in stores. For a better user experience, the user has to be able to use the recommendation system at home.

At home, nowadays almost everyone has a powerful computer and a camera in a smart phone or tablet format. Because of the prevalence and the capacity, these devices are extremely suitable platforms for automatic measurements. These range from devices to aid visually-impaired [17], scene reconstruction [21], augmented reality [28], mobile text translation [22] to visual landmark identification [6], and even greater use of mobile vision applications is to be expected in the future. Unlike the well-established machine vision systems, which perform related tasks, such devices are much more heterogeneous and undesignated. The developer of such a system has no affect on the quality of the camera, the lighting conditions or position of the camera. It is also not possible to demand, that the user performs complex calibration procedures or environment adjustments, which would facilitate the process of image processing and increased accuracy in results.

1.1 Motivation

Because of the above-described highly unpredictable conditions and the variety of available hardware, it is necessary to use highly robust algorithms, and consider all this unpredictability in the development process. The aim of this project is to improve existing methods with advanced computer vision technologies, to solve the problem of automatic feet modelling, and to determine the suitability of the latest mobile devices for such advanced computer vision algorithms.

1.2 Outline

This technical report addresses the suitability of mobile devices and platforms for advanced computer vision methods for modelling feet in real-world environments. A diagram overview is shown in figure 1.1. The technical report is structured as follows. In the second chapter an overview of the mobile application is given. The main goal of the application is to simplify the process of obtaining images with all necessary parameters for further processing. In the third chapter the reference object detection is described, followed by chapter four, i.e., the camera pose estimation. This is required in order to compute a transformation from camera to real world coordinates. Chapter five describes the evaluation of the estimated camera pose, where we determine the estimation error of computed and measured parameters, to determine the suitability of the camera pose estimation method. Chapter six gives an overview of suitable segmentation methods with focus on heterogeneous socks. Chapter seven describes the space carving method and chapter eight the structure from motion method for 3D object construction, based on the segmentation output of the segmentation methods. We finish with conclusions in chapter nine.

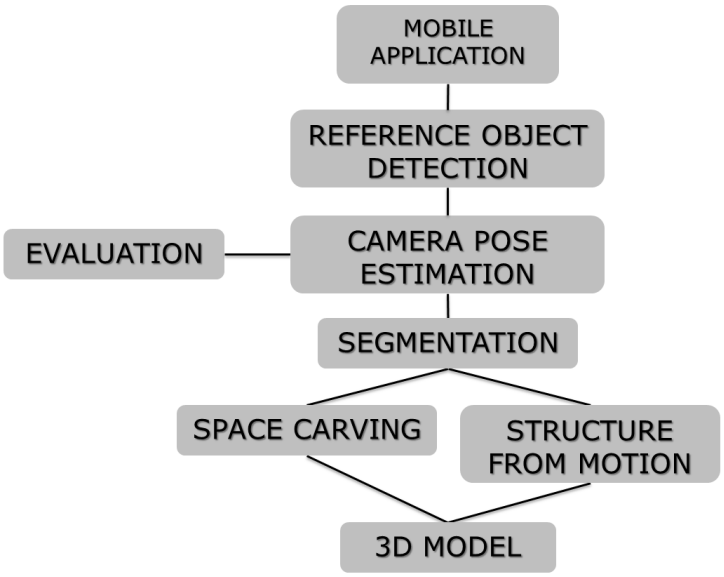


Figure 1.1: Diagram of modules and its dependencies between them.

Chapter 2

Mobile application

To simplify the process of obtaining images and testing of computer vision algorithms we developed a mobile application for the Android operating system. The application is focused on the camera sensor hardware which captures the images for processing. Besides the captured images we also obtain the camera sensor parameters (e.g. calibration matrix, field of view, etc.) and A4 paper sheet corner points with different perspective projections.

Different perspective projections are calculated using cameragen.m function developed by T. Svoboda [27]. The method is modified to obtain perspective projections around the paper with different angles. The z-axis inclination is constant of 50 degrees, the x-axis tilt varies from 0 to 360 degrees and in that way we get perspective projections that all covers entire foot model on A4 sheet. The example of a perspective projection of A4 sheet is shown in figure 2.1.

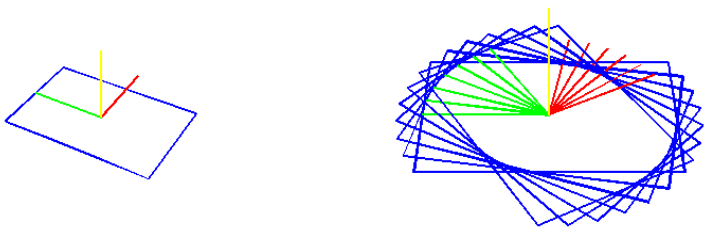


Figure 2.1: Perspective projection of A4 sheet with angle parameters $\phi = 50$ $\theta = 30$ on the left and multiple projections on the right.

The usage of the mobile application is very simple. At application start we enable or disable camera flash using check box. After reading the instructions we start capturing with "Start capture" button. Before capturing we have to level wireframe with the A4 sheet and take a picture. We repeat the capturing with all the available perspective projections, in the current solution we take 36 pictures to cover the entire model with considerable density. After capturing we have two options:

- save archive to mobile device and process it after or
- upload the archive to the server and process it with Matlab script.

Figure 2.2 shows an example how the capturing phase works from a point of view of a common user. Taking pictures can be done by touching screen or use capture button.

Figure 2.2: Example of a users point of view of the mobile application.

From a point of view of the current architecture, the solution is summarized shown in figure 2.3. In future work it can be upgraded with database which can store parameters, partial results and mesh results. In addition the results can be also sent from server to mobile application and displayed using suitable framework.

Figure 2.3: Mobile application is fundamentally used to capture and archive the data. Main taks of the server is to process the archive and display the results. Entire solution basis on client-server architecture.

Chapter 3

Reference object detection

In order to determine the size of an object in an image we need to have a reference object of known size near the object in question. Since we know the size of the reference object, we are able to determine the size of the object in question. In our case the object that needs to be measured is a foot and the reference object is an everyday A4 sheet of paper, onto which the foot is placed.

Before measuring the size of the foot we first need to detected the reference object, in our case an A4 paper sheet. We require a coarse reference object location estimation 3.1, which we obtain with the mobile application. From the coarse reference location we continue to determine the exact A4 paper sheet location in the image.

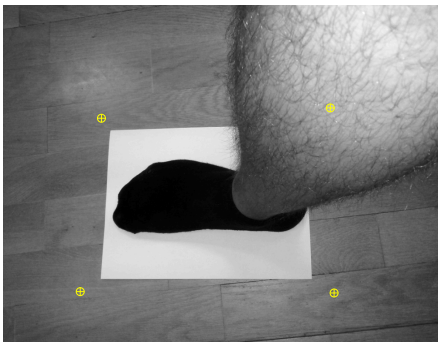
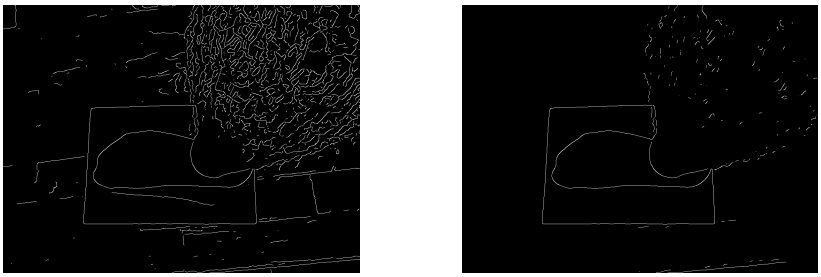


Figure 3.1: Input image I with coarse reference object location estimation, marked by yellow circles with plus signs.

3.1 Edge detection and edge filtering

Edges are detected on the input image using the Canny edge detection algorithm [4]. The output image E of the Canny edge detector is shown in Figure 3.2a. Weak edges are filtered out using a mask M, obtained by thresholding the gradient magnitude image G of the input image with the following formula:



(a) (b)

Figure 3.2: Figure 3.2a shows the output image E of the Canny edge detection algorithm, whereas Figure 3.2b shows the edge image E with removed weak edges.

$$M = G > (mean(G) + std(G))$$

The gradient magnitude image G and mask M are shown in Figure 3.3, wheres Figure 3.2b shows the edge image with removed weak edges.

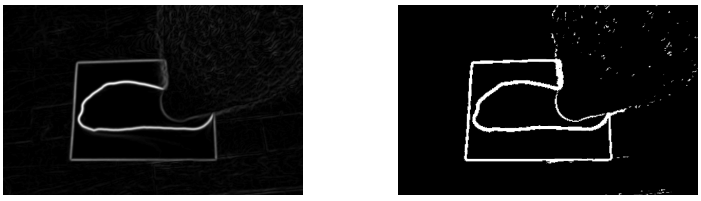


Figure 3.3: Gradient magnitude image G of the input image (left), and the mask M (right) obtained from thresholding G.

3.2 Gradient direction filtering

To further omit noisy edge influence on the line detection process we apply additional gradient direction filtering to the edge image E. The direction filter threshold is set by the following equation:

$$\delta = d * \pi/180$$

where d specifies the allowed deviation in degrees from horizontal and vertical lines.

The masks shown in Figure 3.4 are obtained by the following equations:

$$L = |G| > (\pi - \delta) \tag{3.1a}$$
$$R = |G| < \delta \tag{3.1b}$$
$$D = G > (\frac{\pi}{2} - \delta) \wedge G < (\frac{\pi}{2} + \delta) \tag{3.1c}$$
$$U = G < -(\frac{\pi}{2} - \delta) \wedge G > -(\frac{\pi}{2} + \delta) \tag{3.1d}$$

Half of the mask L, R, D and U is set to zero as shown in Figure 3.5 and all masks are merged to obtain the final gradient direction filtering mask GM 3.7. GM is applied to E, to further reduce inconsistent edges and obtain the edge image as shown in Figure 3.7. The last step in edge filtering consist of utilizing the coarse reference object location estimation points. The points are "inflated" to a region of interest (ROI) as shown in figure 3.7a, which is then applied to E, to filter out additional edges that lie outside of the ROI, thus obtaining the final edge image E' 3.7b.

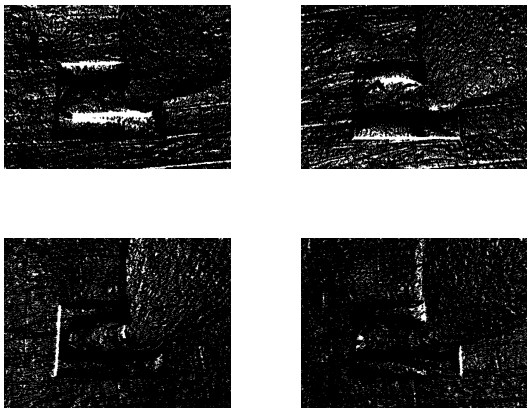


Figure 3.4: Gradient direction binary masks. The upper left mask U is obtained by 3.1d, upper right mask D by 3.1c, bottom left mask L by 3.1a and bottom right mask R by 3.1b.

3.3 Line detection, filtering and fitting

We use the Hough transform algorithm [7] to detect lines on image E' as shown in Figure 3.8a. Nonmaxima suppresion is performed on all detected lines so for each A4 paper edge, the closes line to the edge points is kept as shown in Figure 3.8b. The remaining Hough lines are fitted to the edge points by means of least squares. Only edge points that don't line on other lines are considered. The points are sorted by distance from the detected Hough line, and γ percent farthest points are discarded in order to omit outlier influence on the least squares fitting method. After the fitting process, line intersections are computed and thus we obtain four points representing the exact reference A4 paper sheet's location in the image, as shown in Figure 3.8c.

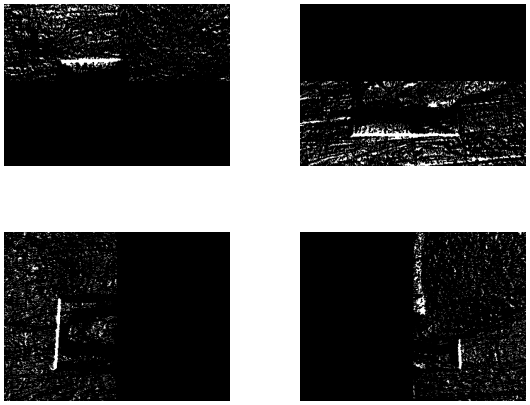
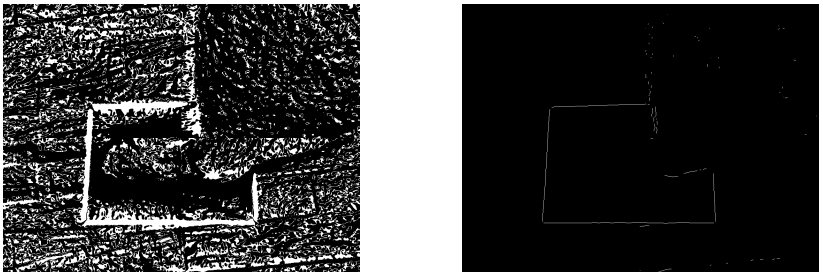
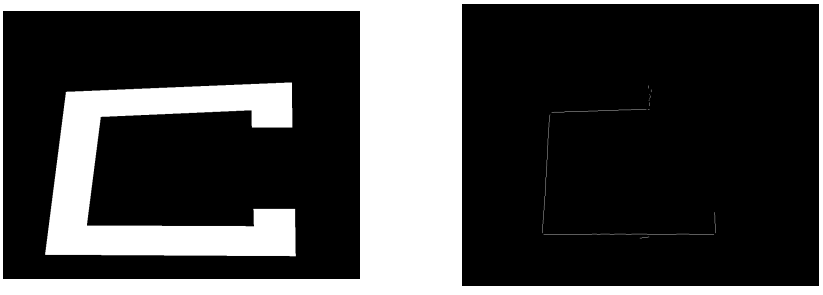


Figure 3.5: Halfed masks L, R, D, U. For mask L the right half is set to zero, for mask R the left half is set to zero. For mask D the bottom side is set to zero, and for mask U the upper side is set to zero.



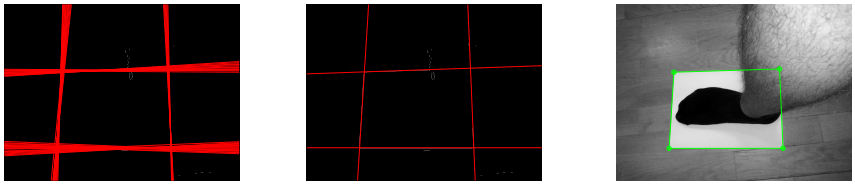
(a) (b)

Figure 3.6: Figure 3.6a shows the merged masks L, R, D, U, i.e., the final gradient direction filtering mask GM. Figure 3.6b shows the mask GM applied to edge image E.



(a) (b)

Figure 3.7: Figure 3.7a shows the Region of interest mask ROI, whereas Figure 3.7b shows the Final edge image E', obtained by applying the ROI mask to E.



(a) (b) (c)

Figure 3.8: Detected Hough lines on edge image E' are shown in Figure 3.8a. Figure 3.8b shows Hough lines after nonmaximum suppression and Figure 3.8c shows the final detected A4 papers sheet.

Chapter 4

Camera pose estimation

Some of the methods used in this project require that we determine the camera pose (position and orientation) in the world coordinate system as shown in Figure 4.1. For that we need to know the rotation and translation, mapping points from the camera coordinates to the world coordinates.

4.1 Position and orientation

Let R represent the rotation matrix and T the translation vector. The relationship between the two systems looks as follows:

$$X_{camera} = R * X_{world} + T$$

The rotation matrix R trivially represents the camera orientation, but the same cannot be said for its position. We know that in accordance with the above formula, this equation must also hold:

$$0 = R * position + T$$

Where $position$ represents camera position in world coordinates. We can then calculate it by re-arranging the equation like this [11]:

$$position = R^{-1} * (-T)$$

4.2 Estimating rotation and translation from a homography

We obtain the rotation matrix and translation vector either through the camera calibration process or, more commonly, estimate it from a homography mapping the pixels on the image plane to points in the world. Note that the camera's internal parameters must be known (it must be calibrated) for this procedure. Let H represent the homography matrix and K represent the camera calibration matrix. In the first step we obtain the auxilliary matrix B from both:

$$B = K^{-1} * H$$

Let's label B's columns as following: $B = [r_1\ r_2\ t]$. Ideally, r_1 and r_2 would be the same lengths, but in practice they are not. Let's take their length average λ, which is a scalar:

$$\lambda = \frac{||r_1|| + ||r_2||}{2}$$

We can now estimate the translation vector T:

$$T = \lambda * t$$

For the rotation matrix, we also need a third column, which is simply a cross-product of the first two: $r_3 = r_1 \times r_2$. Finally, we combine these three columns in a matrix and scale it with λ to make orthonormal - the result is an estimated rotation matrix R:

$$R = \lambda * [r_1\ r_2\ r_3]$$

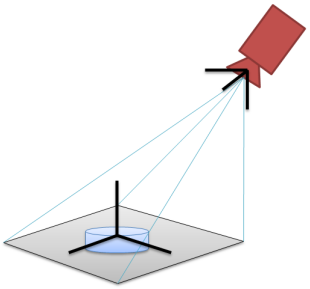


Figure 4.1: Reference object, i.e., local coordinate system and camera, i.e., world coordinate system.

Chapter 5

Evaluation

When designing machine vision applications for mobile devices, we have very little control over the hardware (camera included) our application has to run on. We need to account for a plethora of different mobile phone and tablet cameras, primarily designed for social networking. In this case the robustness of our machine vision methods faces greater challenges than it would if we used purpose-built industrial cameras. To combat this, it is helpful to be able to estimate and model errors that are a result of mobile device cameras' shortcomings.

We have therefore devised a methodology for a evaluation of cameras on mobile devices and have conducted a pilot evaluation on a very small sample of devices, with plans to expand it to a much larger sample of smartphones and tablets currently on the market.

5.1 Calibration board

We have designed a calibration board to be used in the evaluation procedure. It features an A4-sized white rectangle (a "paper sheet") printed on a gray background (Figure 5.1). This helped us minimize any potential errors resulting from bending or crumpling if we used a real A4 sheet of paper. Finally, an asymmetrical pattern of 44 spots is printed on the paper sheet. This pattern is used in camera calibration as well as evaluation. It is asymmetric to remove any orientation ambiguities.

To ensure the reliability of the entire camera calibration and evaluation procedure, precise digital print with an error margin of at most 0,1 mm was used to produce the board.

5.2 Procedure and results

With each mobile device, a series of 21 images of the calibration board was taken from various angles and distances, as shown in Figure 5.2. Calibration was then performed on the series, giving us the calibration matrix and radial distortion coefficients for the camera, and a rotation matrix/translation vector for each image.

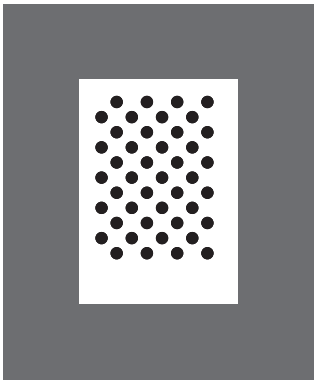


Figure 5.1: The calibration board, with an A4-sized rectangle, a 10 cm dark gray margin and a calibration pattern.

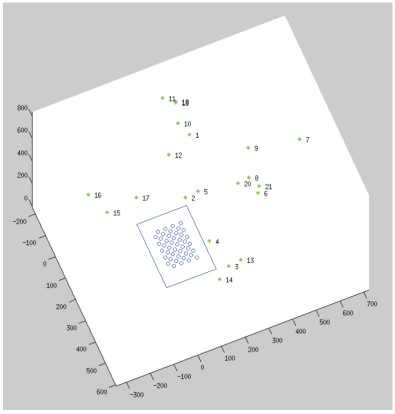


Figure 5.2: Different camera positions in real world coordinates.

We treat the calculated camera poses as ground-truth for further error estimation, alongside the known calibration pattern positions and measurements.

For the actual error estimation we then detected the paper sheet on each image and estimated a homography from the image plane to the paper plane. Using a simple blob detector, the pattern of spots was detected. We estimated the camera pose from the homography and also used it to estimate the position of each spot on the paper sheet. For each image measured the errors in estimation of the following parameters:

A) camera orientation

B) camera position

C) position of each of the 44 points of the pattern

D) total width of the pattern

E) total height of the pattern

	Apple iPad Air		Motorola Moto X		Samsung Galaxy Nexus	
	original	undistorted	original	undistorted	original	undistorted
A	$0.580°$	$0.196°$	$0.453°$	$0.132°$	$0.416°$	$0.173°$
B	8.765 mm	2.890 mm	5.900 mm	1.276 mm	10.115 mm	3.331 mm
C	0.473 mm	0.535 mm	0.329 mm	0.386 mm	0.631 mm	0.499 mm
D	0.354 mm	0.104 mm	0.136 mm	0.127 mm	0.464 mm	0.187 mm
E	0.635 mm	0.242 mm	0.244 mm	0.130 mm	0.584 mm	0.200 mm

Table 5.1: Median errors of estimations of A) camera orientation, B) camera position, C) position of pattern spots, D) pattern width, E) pattern height. Error estimation was done both on the original series of images as well as a series where images were undistorted to compensate for radial distortion of the lens.

A fully automatized script was written in MATLAB. It relies on OpenCV [3] methods for pattern detection and calibration and our own methods for everything else, taking a series of images captured on a device as an input, then performing all steps of calibration and evaluation as described above, finally outputing the evaluation results.

The initial results of a small pilot study (see Table 5.1) show that errors on cameras of modern phones and tablets are well within acceptable margins for our use. Nevertheless, a broader evaluation of more than three dozens of currently available mobile devices is planned - to obtain device-specific results as well as broader, statistically more valid data.

Chapter 6

Segmentation

6.1 Segmentation

Image segmentation is one of the most important tasks in computer vision. The main goal of image segmentation is to group together image pixels that belong to the same entity in an image, therefore to segment the image to corresponding regions.
As part of preprocessing, we have implemented an algorithm for automatic image cropping. This algorithm receives as its input an image as captured by the user, i.e. a foot, a sheet of paper and the floor are all present on the image. An example of such an image is in Fig. 6.1. The algorithm is also supplied with the coordinates of the obtained four corners of the A4 paper sheet reference object detection. On its output the algorithm returns a cropped image without the background (i.e. only a foot and the paper present). A cropped version of the image in Fig. 6.1 is in Fig.6.2a.

Figure 6.1: An example of a user-captured image

Segmentation algorithms then obtain a cropped image with only a foot on the paper sheet, and return two sets of pixels which represent a foot segment and a background segment.
For accurate and reliable evaluation of performance of segmentation algorithms we need a correctly labeled test set. This ground truth set of images was obtained by an interactive manual-automatic segmentation which was carried out on the cropped

images, obtained in the previous step. As the result we got foot segments which we binarized to obtain binary segmentation masks, on which black color represents a foot and white color represents background (in our case a sheet of paper). An example of a binary segmentation mask is in Fig. 6.2b.

(a) Cropped image from Fig. 6.1

(b) Binary segmentation mask of Fig. 6.2a

In the rest of Section 6.1 we address the comparison of segmentation algorithms in a limited scenario with only one object on a uniform background. In our study we have considered six image segmentation algorithms and analysed their performance in a specific task of segmenting a single object on a uniform background in different settings and illumination conditions. For the results to more generally applicative, we did not commit ourselves to foot images but have instead made the comparison on the dataset of 50 different objects. The results have also been published in [29].

6.2 Our scenario of comparison

In general, segmentation is an ill-defined problem. General images can be segmented in different ways, up to a different level of detail, depending on the task. In our study we limited our research to a specific, more constrained problem: segmentation of a single object placed on a uniform background. We have therefore dealt with the scenarios, where

- the background is of uniform color, without the texture and clutter,
- there is only one object on a scene, therefore no occlusions (except self-occlusions) may occur,
- the object is positioned in the center of the image and is fully contained in the image,
- the object can be of arbitrary colors and shape,
- the illumination conditions are not constrained.

The goal is well-defined: to partition an image into two regions containing pixels that belong to the object and to the background, respectively.

Such constrained settings often simplify the segmentation problem. Such an example is depicted in Fig. 6.3(a), where the background is very homogeneous and the boundary between the object and the background is clear and sharp. However, when the same object in the same position is illuminated with a strong point light source, such as in the image depicted in Fig. 6.3(b), the segmentation task becomes significantly more difficult due to nonhomogeneous background and strong shadows. Even more difficult

problem we face when the object to be segmented, or a part of it, is of a similar color as the background (e.g., Fig. 6.3(c)).

(a) (b) (c)

Figure 6.3: Examples of objects with (a) uniform illumination, (b) strong shadows, (c) problematic object color.

In machine vision applications, the problems depicted in Figs. 6.3(b) and (c) are usually avoided by assuring suitable illumination conditions and carefully choosing the color of the background. We will refer to scenarios, where this is not possible (e.g., where the illumination cannot be controlled). The goal of this study is to analyse the performance of several segmentation algorithms in such scenarios. We will briefly describe six chosen segmentation algorithms and analyse their performance in different conditions and illumination settings.

6.3 Segmentation methods

Contour evolution-based image segmentation methods are well-suited for segmenting non-textured images with only one object. In this study we compare the performance of Active contour[13], Chan Vese model[5], DRLSE[15] and GrabCut[24] under different illumination conditions. Otsu's segmentation method[20] and adaptive thresholding[25] are included for comparison reference.

Active Contour Model

Active contour model (also known as *snakes*)[13] has been proven to be a promising framework for image segmentation. The fundamental idea of the active contour is to evolve a spline curve under the influence of energy functions. Energy functions are set up in such a way that in the equilibrium position the spline curve conforms to the object boundary or to other desired features. In steady state, the internal energy term (controlling the smoothness of the spline curve) counterbalances the external energy term (edge image), thereby conforming the curve at the edges. However, the sensitivity of snakes to initialization and poor convergence have limited its use in image segmentation. To overcome these limitations, various improvements to the active contour method have been proposed in the past [30, 12].

Chan Vese Model

Level set method, based on the active contour model, involves representing contours as the zero level of an implicit, level set function[19]. The level set method overcomes the limitation of contour continuity in the active snakes model, thereby handling topological changes like splitting and merging. Chan and Vese proposed a model of active contour without edges (Chan Vese model)[5], which is based on level set evolution and Mumford-Shah segmentation techniques[18]. Unlike in active contours, the

curve evolution stopping term in the Chan Vese model is based on the minimization of Mumford-Shah functional[18]. Consequently, the Chan Vese model is capable of segmenting objects with minimal or no gradient at all. However, as the Chan Vese model is based on global image information, it fails to segment images with intensity inhomogenity. Another drawback of Chan Vese model is its slow convergence due to the step of reinitialization. Reinitialization involves assigning a signed distance function to the downgraded level set, thereby preventing any irregularities in the curve evolution.

DRLSE

Li and Xu redress the constraint of reinitialization by proposing distance regularized method for level set evolution (DRLSE)[15]. In DRLSE, the energy functional for curve evolution inherently contains the distance regularization term along with the external energy, thereby eliminating the need for reinitialization. As the distant regularization term is embedded in the level set evolution, the DRLSE method does not require reinitialization at periodic intervals, thereby making the method computationally fast and accurate. However, unlike the Chan Vese method, DRLSE fails to detect objects with weak boundaries, resulting in boundary leakage.

GrabCut

GrabCut is an interactive object segmentation method based on the application of graph cuts in conjunction with some apriori knowledge of foreground and background of an image[24]. GrabCut uses the graph cut algorithm to solve an optimization problem (Energy cost function)[2] by creating a specific weighted graph model, where each vertex corresponds to an image pixel and weights of each edge, connecting vertices, represent similarity between pixels. The main problem of GrabCut is its inability to segment images with low contrast between the foreground and background colors, as the GrabCut optimization algorithm is based on the probabilistic model for pixel color distribution.

Otsu's Thresholding

Thresholding involves classifying image pixels into categories of foreground and background pixels based on a given intensity threshold. Clearly, the main problem is to precisely detect the threshold which would give the optimal binarization result. Otsu's thresholding[20] selects the threshold value that minimizes the intra-class variance (or maximizes the inter-class variance), as a result of which the thresholding performs well only for images having bimodal intensity distribution.

Adaptive Thresholding

Fixed threshold cannot result in an appropriate segmentation when images are poorly illuminated. Local adaptive thresholding[25] solves this problem by selecting an individual threshold for each pixel based on intensity values of its neighbouring pixels. As a result, local adaptive thresholding performs much better even in the case of images for which global thresholding fails completely.

6.4 Experimental results

The performance of the above-listed six methods has been evaluated on the newly generated image dataset of 50 objects exposed to 4 different types of illumination conditions (the total of 200 images). Some of these images are shown in Fig. 6.4, while the illumination types are described below:

Type 1: Low intensity images having minimal contrast between the foreground and background colors (Fig. 6.4(a)).

Type 2: Images with sufficient lighting conditions having negligible shadows (Fig. 6.4(b)).

Type 3: Highly illuminated bright images with moderate shadows (Fig. 6.4(c)).

Type 4: Images with strong shadows having high intensity inhomogeneity (Fig. 6.4(d)).

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

Figure 6.4: Examples of dataset images (a) Type 1 images (images of low intensity), (b) Type 2 images (images with no shadows), (c) Type 3 images (bright images with moderate shadows), (d) Type 4 images (images with strong shadows).

All images were transformed to grayscale before applying the segmentation algorithms. All the contour-based methods require an initial contour to be set. We used the apriori knowledge about images (one object in the center of a uniform background) to set the initial contour automatically. A rectangular contour (with size 10 percent less than the image size) was used to initialize Active contour, DRLSE and Chan Vese methods. GrabCut algorithm was initialized by declaring a square (20% of the image size) in the image center as foreground and all the pixels outside the rectangle (size 10% less than the image size) as background. We refer to this type of initialization as unsupervised, since we did not use the ground truth information for individual images. To verify the influence of the initialization to the segmentation process we also initialized the contours by considering the ground truth segmentations. This supervised initialization

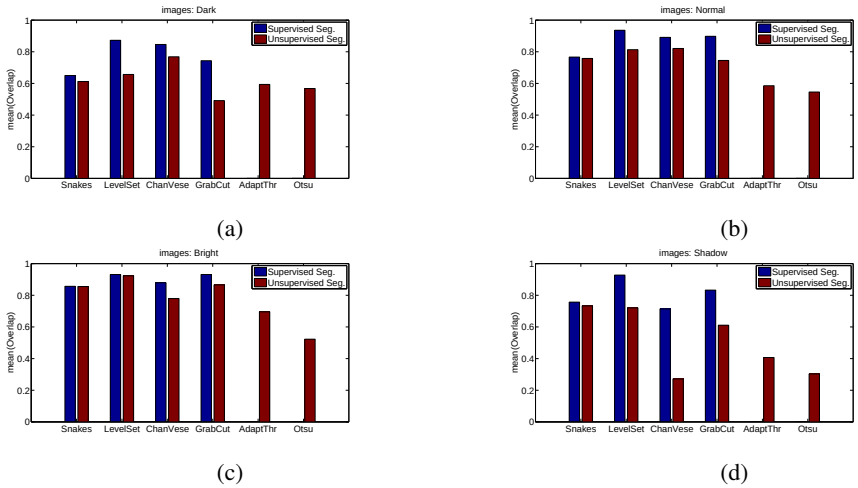


Figure 6.5: Comparison of segmentation methods (a) Type 1 images (low intensity images), (b) Type 2 images (images with no shadows), (c) Type 3 images (images with moderate shadows) , (d) Type 4 images (images with strong shadows).

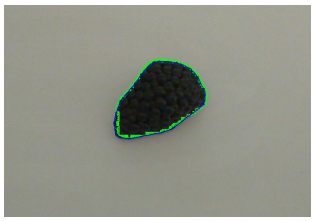
was done in the case of contour based methods by dilating the ground truth contour by 10 percent while the initialization for supervised GrabCut was done by declaring all the pixels inside the eroded ground truth as foreground and all pixels outside the dilated ground truth as background.
For all the methods we empirically determined the parameters that produce the best results, fixed the parameters and used them for segmenting all images.
For measuring the performance of the segmentation methods, the overlap measure was used, considering the segmented and ground truth regions as follows:

$$Overlap = \frac{|R_{result} \cap R_{groundTruth}|}{|R_{result} \cup R_{groundTruth}|} \quad . \tag{6.1}$$

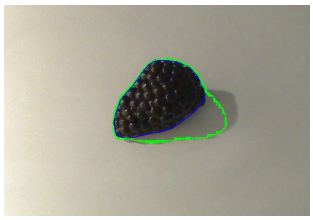
The results are depicted in Fig. 6.5 and are further analyzed in the remaining of this section.

Type 1 images: For images with low intensity and less contrast between the foreground and background pixels, the overlap accuracy of unsupervised Grabcut was found to be lower than with all other methods (Fig. 6.5(a)). This is because of the fact that GrabCut is based on the probabilistic model for pixel color distribution and this makes it ineffective to segment low-contrast images (an example is shown in Fig. 6.6(h)). The performance of unsupervised Chan Vese was found to be the best (almost similar to that of supervised GrabCut method) as the Chan Vese model is capable of segmenting images with minimal gradient. By providing perfect initialization the results significantly improve (by 50% for GrabCut and 30% for DRLSE) compared to its unsupervised counterpart.

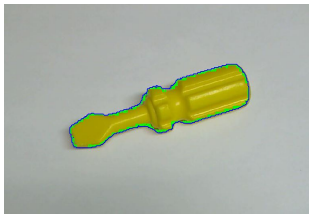
Type 2 images: For images taken under sufficient lighting conditions with clear edges and good contrast all the methods performed well (Figs. 6.6(a) and (c)). Unsupervised DRLSE and Chan Vese model performed best with the mean overlap of 81% and 82% respectively, while in the case of the perfect supervised initialization DRLSE achieved



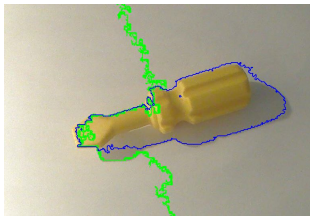
(a) Snakes (Type 2 image)



(b) Snakes (Type 4 image)

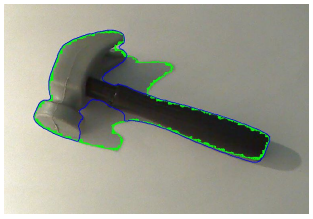


(c) Chan-Vese (Type 2 image)

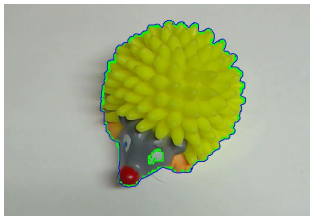


(d) Chan-Vese (Type 4 image)

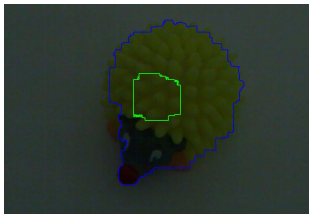
(e) DRLSE (Type 3 image)



(f) DRLSE (Type 4 image)

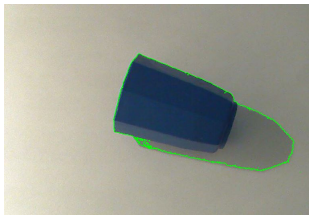


(g) GrabCut (Type 3 image)



(h) GrabCut (Type 1 image)

(i) Adap. Thr. (Type 4 image)



(j) Otsu's Seg. (Type 4 image)

Figure 6.6: Performance of the segmentation methods on different images. Green contour is the result of unsupervised segmentation and blue contour is the result of supervised segmentation.

even 94% of the mean overlap (Fig. 6.5(b)).

Type 3 images: For the images with moderate shadows, gradient based unsupervised

contour methods (Snakes and DRLSE) performed surprisingly well with the results comparable to their supervised counterpart (Figs. 6.6(e) and (g)). This is because of the presence of good contrast (and hence high gradient) resulting from shadows along the object boundaries. Even though unsupervised DRLSE and GrabCut were able to achieve the mean overlap as high as 92% and 87% respectively (Fig. 6.5(c)), the performance of Chan Vese model decreased from Type 2 to Type 3 images. This is the consequence of the nonhomogeneous intensity distribution of the image. As the Chan Vese model is based on global image information, it fails to segment images with intensity inhomogeneities.

Type 4 images: Due to presence of strong shadows and nonuniform illumination, the performance of all the methods dropped by 25%, on average, from Type 3 to Type 4 images (Fig. 6.6 (b), (d), (f), (i) and (j)). The effect of intensity inhomogenity was significant on unsupervised Chan Vese model, with its accuracy reduced by 60% (Fig. 6.5(d)).

By analysing the results we can therefore conclude that: **(a)** for images with low intensity and minimal gradient, Chan Vese method performs better than other unsupervised methods, **(b)** for images with no shadows and with uniform illumination, DRLSE and Chan Vese model outperform other methods, **(c)** for images with high gradient along edges and moderate shadows around the object, unsupervised edge-based active contour (Snakes and DRLSE) perform as good as their supervised counterpart, and **(d)** strong shadows and nonhomogeneous intensities significantly influence the effectiveness of the segmentation methods, especially the Chan Vese method.

In previous subsections we considered a general dataset of different objects. In Fig. 6.7 there is an example of the final segmentation contour, as obtained by the DRLSE algorithm on a foot image from Fig. 6.2b. Figure 6.9 shows additional segmentation results on real feet.

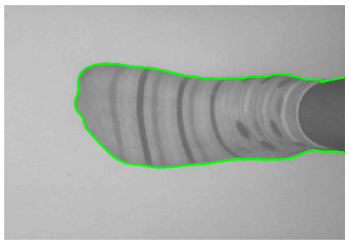


Figure 6.7: Segmentation contour as obtained by DRLSE on the image from Fig. 6.2a

The contour initialization turned out to be a very important step, so in our future work we will aim at improving the automatic initialization. We will also incorporate information about the color in the segmentation process. Furthermore, we will aim at finding an automatic way of setting parameters of individual methods to further improve the overlap accuracy of the segmentation algorithms.

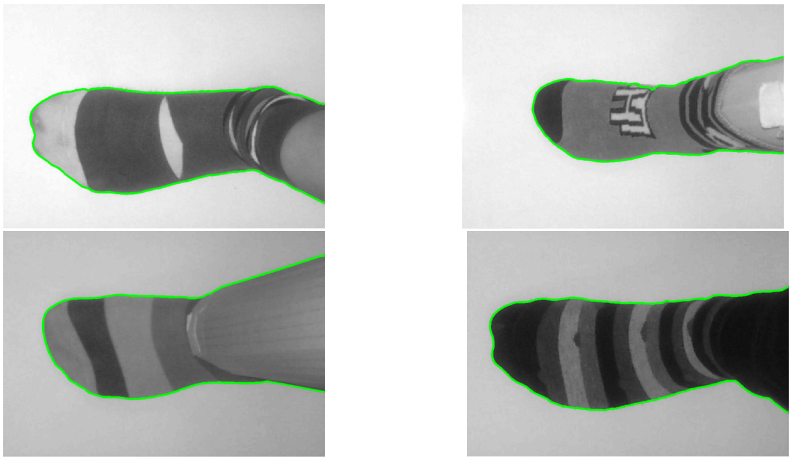
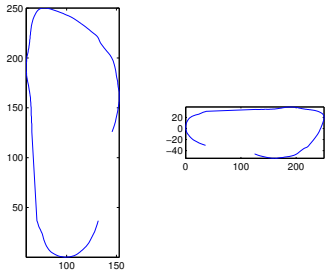


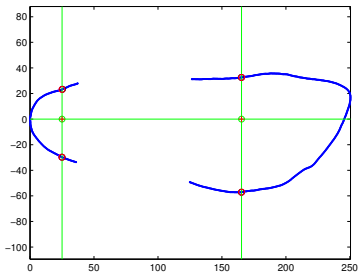
Figure 6.8: Segmentation results on real feet.

6.5 Contour alignment

The obtained contour's coordinates are given in real world coordinates. For further processing we need to align the contour into a so called "canonical form" 6.9a. This is done by aligning two points of the contour with the x-axis and setting the foot's "heel" to the coordinate system origin as shown in Figure 6.9b. We call this the canonical form of the contour. The back alignment point is obtained by taking 10% of the contour's length and half of the contour's heel section. The front alignment point is obtained by taking 66% of the contour's length and splitting the front end $40{:}60$, as shown in 6.9b. These two points are than aligned with the x-axis and the foot's "heel" is set to the coordinate system origin. The contour in canonical form is then used as input to the UCS developed algorithm for 3D foot model construction.



(a) Example of an unaligned contour left, and a contour in canonical form right.



(b) Example of contour alignment to the so called canonical form.

Figure 6.9: Canonical form of foot contour.

Chapter 7

Space carving

In this section we consider the problem of computing the 3D shape of an foot object using multiple photographs taken from different viewpoints [14]. The method was presented in two variations of 3D shape reconstruction. It can be used to reconstruct entire scenes or objects. Our paper describes object reconstruction where the base for 3D reconstruction are object contours obtained by segmentation from photographs. The main goal is to obtain 3D mesh without textures.

An example of 2D space carving is shown in figure 7.1. From the figure we can outline the method flow and implementation challenges. First we must obtain photographs from different view points. From each photograph we must extract the contour of the desired object to be carved. Next to the photograph we must estimate camera position and orientation. Those two parameters must be accurate as possible to successfully carve the final 3D model. The initial mesh must be generated according to the domain but in any case the volume of mesh must fit inside the all the view points. Each view point reduces the initial mesh by carving vertices that are not included in the contour. Final result is the reduced initial mesh with vertices. Fundamentally the method does not generate any faces.

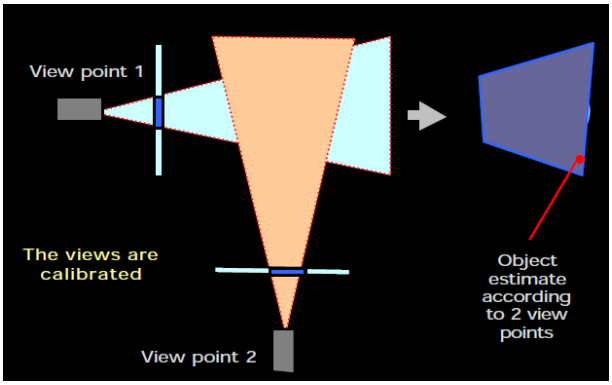


Figure 7.1: 2D space carving result carved from 2 view points [26].

7.1 Domain and limitations

The domain, on which the Space Carving method is implemented and adopted, is contactless foot measuring and 3D reconstruction. Almost each human foot can be placed on a standard A4 sheet size of 210 x 297mm which is used as measure reference. Through the study of the method we didn't encounter any critical limitations. The biggest limitation is the time complexity of the algorithm which is O(n3) where n is the length or number of vertices. In our concrete case the time complexity is O(w*l*h) where w = 210 vertices, l = 297 vertices and h = 200 vertices. The initial mesh is also normalized with amount of 1.000.000 vertices. Consequence of elevated time complexity is that not all mobile devices are recommendable for executing such task. Many mobile devices are limited with little computational power (memory, CPU, GPU) in confront of desktop computers and servers. Due to foot form with absence of concavities we avoid the biggest method limitation. The method is not able to model concavities on objects.

7.2 Solution

The solution was developed using two different platforms - Android and Matlab. In that way the user can obtain all data with mobile device such as photographs and other parameters in a form of a data archive. For Matlab platform there is a fully automated script that receives as input data archive and 3D model as output. The entire Space carving solution can be summed up in 8 steps:

1. Mobile application is described in previous sections. If we summarize the step, user take pictures for available perspective projections and next to picture wireframe parameters are stored. Regardless of mobile device camera also calibration matrix is calculated and stored in the archive.

2. We generate the initial rectangular mesh for foot domain with volume of 210 x 297 x 200 vertices as shown in figure 7.2a.

3. The script reads archive data and stores local parameters in an array object. The object contains the image and wireframe rectangle. Missing parameters (internal camera parameter matrix, external parameter matrix for rotation, translation matrix, contour) are calculated in following steps.

4. On each photograph we must detect reference object, in our case A4 sheet. The whole process is described in section 3. The outcome of the method is homography calculated for each photograph. If the algorithm fails to obtain the result, we discard the photograph and it is no longer used in further steps.

5. On each photograph we calculate camera position and orientation from homography as described in section 4. An example of a photograph oriented towards the initial mesh is shown in figure 7.2b.

6. On each photograph we must find the desired object. The segmentation described in section 6 has some limits in space carving case. Regarding to perspective projection we can't ensure that the object is situated on homogeneous basis as shown in figure 7.3a. The outcome of segmentation is a binarized image shown in figure 7.3b.

7. After all parameters set we can reduce the initial mesh with carving. The mesh must be projected into segmented image with following formulas for each vertex:

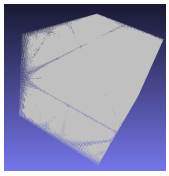
$$
\begin{aligned}
P &= K * \begin{bmatrix} R & T \end{bmatrix} \\
z &= P[3,1] * world_X + P[3,2] * world_Y + P[3,3] * world_Z + P[3,4] \\
image_y &= (P[2,1] * world_X + P[2,2] * world_Y + P[2,3] * world_Z + P[2,4])/z \\
image_x &= (P[1,1] * world_X + P[1,2] * world_Y + P[1,3] * world_Z + P[1,4])/z
\end{aligned}
$$

   First formula shows how projection matrix P is calculated using position matrix K, rotation matrix R and translation matrix T. Image pixel position $(image_x, image_y)$ is calculated with projection matrix P and 3D mesh points $(world_x, world_y, world_z)$.

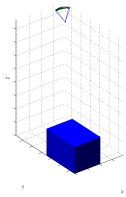
8. We first clear vertexes that are out of the image and then that are not inside the silhouette. The procedure is repeated for each image.

9. In the final step we save the result mesh. The length an width can be calculated from number of vertexes toward each X and Y axis. An example of partial result of space carving is shown in figure 7.3c.

To get the complete 3D mesh result we must repeat carving (step d. to h.) with many different images from different angulations. In future work we must calibrate and upgrade parameters to get complete results of space carving method. Each step must be very robust to minimize carve errors.

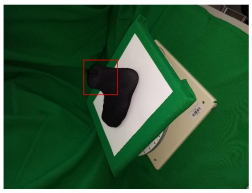


(a) Initial rectangular mesh ready to be carved.

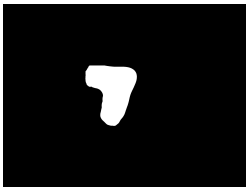


(b) Photography positioned and oriented in 3D space toward the initial mesh.

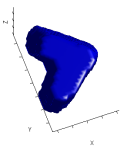
Figure 7.2: Space carving result example.



(a) Original image with problematic area marked with red rectangle.



(b) Segmented image with interesting area shown in white colour.



(c) Partial result of space carving.

Figure 7.3: Space carving result example.

Chapter 8

Structure from motion

8.1 Overview

As a second method for 3D reconstruction from a sequence of images we implement structure from motion. We assume that the object is rigid and motionless during acquisition of the images. This method also assumes that the images are ordered and that the intirinsic parameters of the camera are known. To retrieve the structure information from the images we require image points correspondences between multiple images (2 or more) originating from the same scene point. Overview of the process is shown in Figure 8.1. The method used here is in more detail described in [23].

The first step is to extract the features from the images. Because we assume the images are ordered we only search for correspondences between features of consecutive images. Wrong correspondences are usually present therefore RANSAC [8] and epipolar geometry is used to reduce spurious matches. Next step is the reconstruction of feature points. Initial reconstruction is made using the first two views. This is done using epipolar geometry [11]. Any additional view can than be added by doing pose estimation to compute the pose of the new camera with relation to the first camera (the first camera is placed at the origin). Features corresponding to points in the previous views are refined and new features are added to the reconstruction.

Result of this process are the relative poses between the cameras and the reconstruction of feature points from all of the images (up to arbitrary scale factor). In the following subsections we explain in detail the implementation of the method.

8.2 Extracting features and matching

In this step we extract features that can be differentiated from other features and matched correctly. To achieve this we use SIFT keypoints and descriptors (from library VLFeat) [16]. Features between images are than matched using the same library. We usually get some false matches, but they will be eliminated using algorithm described in 8.4.
This is the slowest part of the computation and could probably be sped up using a faster feature detector (e.g. SURF [1]).

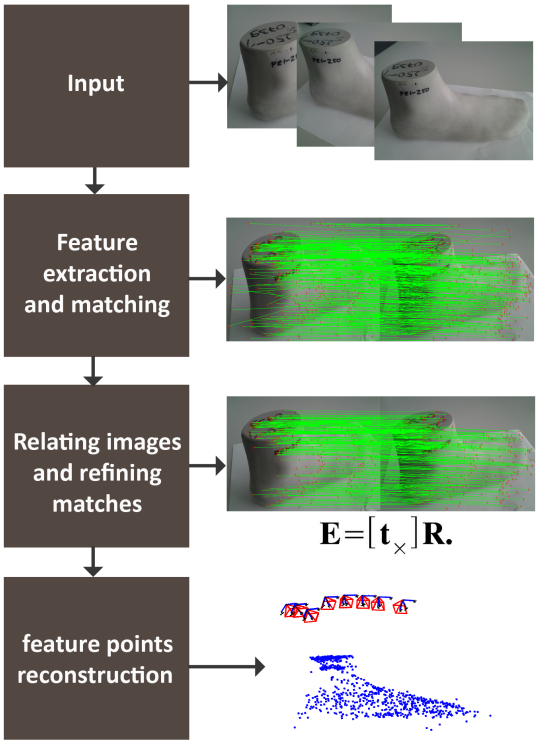


Figure 8.1: Overview of the reconstruction process.

8.3 Two-view geometry

Corresponding image points, u_0 and u_1 in an image pair are related to each other via the fundamental matrix F (points are $3x1$ vectors written in homogeneous image coordinates). Vector Fu_1 describes a line in the first image on which the corresponding point u_0 must lie on. Therefore for all pairs of points equation 8.1 must hold.

$$u_0^T F u_1 = 0 \tag{8.1}$$

To compute fundamental matrix we use the eight-point algorithm [11]. Note that F is determined up to an arbitrary scale factor, so 8 equations are needed to obtain a unique solution. Equation 8.1 can be rewritten in the form of equation 8.2.

$$\begin{bmatrix} x_0x_1 & x_0y_1 & x_0 & y_0x_1 & y_0y_1 & y_0 & x_1 & y_1 & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ \vdots \\ F_{33} \end{bmatrix} \tag{8.2}$$

Here the symbols $F_{11}, F_{12}, \cdots, F_{33}$ are elements of the fundamental matrix and $u_0 = [x_0, y_0, 1]^T$, $u_1 = [x_1, y_1, 1]^T$. By putting 8 of these equations in matrix A and elements of fundamental matrix in vector x we get the following equation:

$$Ax = 0 \tag{8.3}$$

To solve this system we can use the singular value decomposition. With SVD matrix A is decomposed onto matrices U, S and V, such that $A = USV^T$. The solution is the last column of V. Because the intrinsic parameter matrix is known we can calculate essential matrix E using equation 8.4. When the essential matrix is known the relative motion between cameras can be computed directly.

$$E = K^T F K \tag{8.4}$$

8.4 Robust matching

To reject the wrong matches we use the algorithm called RANSAC (random sample consensus) [9] and epipolar geometry. The algorithm works by first computing the fundamental matrix on the smallest subset of matches needed by the 8-point algorithm. Based on the calculated solution we can segment the matches into inliers and outliers. A match is considered an inlier if the corresponding points are not too far away from the corresponding epipolar line (i.e. closer than 1 pixel). Our initial solution will probably not be the best, therefore we repeat the process multiple times and in the end keep the solution with the largest number of inliers. The solution can then be refined using all of the inliers. Number of samples needed to ensure we get correct solution depends on the percentage of outliers. The algorithm is summarized in the following steps:

1. Extract and match features
2. For n iterations do
   (a) select a sample of 8 points
   (b) compute fundamental matrix F
   (c) determine inliers
3. Refine F based on all inliers

8.5 Initial reconstruction

The first two images are used for the initial reconstruction. Relative pose between the cameras is obtained with algorithm described in 8.3 and 8.4. The matches are then reconstructed with triangulation.

Triangulation determines a point in 3D space given its projections in the images and their camera projection matrices [10]. Because of noise the lines generated by the image points do not intersect, therefore we have to find a point that is the closest to both lines.

We do this by constructing matrix A like shown in equation 8.5

$$A = \begin{bmatrix} \begin{bmatrix} 0 & -1 & y_0 \\ 1 & 0 & -x_0 \\ -y_0 & x_0 & 0 \end{bmatrix} P_0 \\ \begin{bmatrix} 0 & -1 & y_1 \\ 1 & 0 & -x_1 \\ -y_1 & x_1 & 0 \end{bmatrix} P_1 \end{bmatrix} \tag{8.5}$$

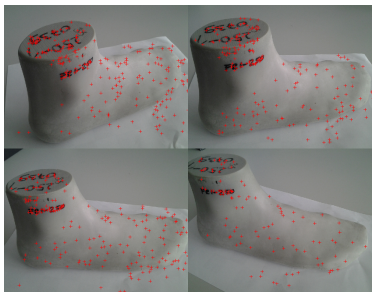
with image points $u_0 = [x_0, y_0, 1]^T$, $u_1 = [x_1, y_1, 1]^T$ and P_0 and P_1 being $3x4$ camera projection matrices. By solvig equation $Ax = 0$ using SVD we obtain a 3D point in homogeneous coordinates.

8.6 Adding view and updating structure

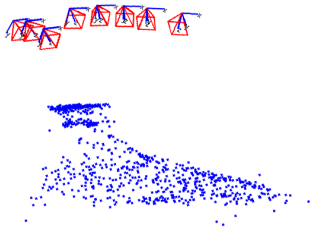
Once the initial structure is reconstructed, additional views can be added. The goal is to find a camera projection matrix for the new image and then add new features to the reconstruction and update existing ones.
Features of the new image can be segmented into features that correspont to already reconstructed points (reconstructed features) and the ones that do not (new features). The first step is to obtain the camera matrix of the new image. For this purpose we use pose estimation to find camera pose using information from reconstructed features and their corresponding image points in the new image. Once the camera matrix is known, new features can be added and existing ones updtated using the same process as described in the 8.5.

8.7 Final reconstruction

Reconstruction and camera poses obtained by the process described in this chapter is shown in Figure 8.2b and four of the nine input images are shown in Figure 8.2a. There is still a lot of space for improvement like global optimization of camera parameters and dense reconstruction which would lead to more accurate results. This will be further analysed in future work towards 3D reconstruction.



(a) Some of the images used for reconstruction.



(b) Reconstruction of feature points from multiple views.

Figure 8.2: Space carving result example.

Chapter 9

Conclusion

In this technical report we addressed the question of whether it is possible to use mobile device platforms for advanced computer vision methods for modelling feet in real-world environments. In real world environments we have very little or no control over various factors, such as illumination changes, that strongly influence scenes and thus affect captured images. We developed a mobile application in order to standardize image capturing over a wide range of mobile devices. A reference object, i.e., an A4 sheet of paper is used to determine the foot's size. After the reference object is detected, the cameras position is estimated and the foot's contour segmented. On the so obtained contour, 3D object reconstruction was applied using space carving and structure from motion.

We showed that mobile devices are suitable for machine vision tasks, depending on the built-in hardware. Objects, in our example feet, can be measured accurately if the reference object detection is accurate enough. In terms of segmentation, the obtained results depend on the illumination of the object in the scene. Satisfactory results can be achieved in controlled conditions with an acceptable processing time.

Space carving has shown to be problematic from a computational viewpoint on mobile devices. Structure from motion, on the other hand, has shown to return promising results on textured objects, where a good reconstruction can be achieved.

With all the evaluated studies UCS has gained a better perspective in feasible computer vision applications for mobile devices at the present time. With the gained knowledge the company is able to better facilitate future business decisions and development options.

Acknowledgement

The project is partly funded by the European Union through the European Social Fund. The project is implemented under the Operational Programme for Human Resources Development for the period 2007-2013, Priority axis 1 ”Promoting entrepreneurship and adaptability, and priority guidelines” 1.3. ”Scholarship Scheme”, within the approved operations ”A creative way to practical knowledge”.

REPUBLIKA SLOVENIJA
MINISTRSTVO ZA IZOBRAŽEVANJE, ZNANOST IN ŠPORT

JAVNI SKLAD REPUBLIKE SLOVENIJE ZA RAZVOJ KADROV IN ŠTIPENDIJE

Naložba v vašo prihodnost
OPERACIJO DELNO FINANCIRA EVROPSKA UNIJA
Evropski socialni sklad

Bibliography

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE PAMI*, 23(11):1222–1239, 2001.

[3] G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986.

[5] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[6] D. Chen, K. Baatz, G.and Koeser, S. Tsai, R. Vedantham, T. Pylvanainen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. City-scale landmark identification on mobile devices. In *Com. Vision Patt. Recognition*, 2011.

[7] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.

[8] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[9] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[10] Richard Hartley, Rajiv Gupta, and Tom Chang. Stereo from uncalibrated cameras. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 761–764. IEEE, 1992.

[11] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[12] M. Jung, G. Peyre, and L. D. Cohen. Nonlocal active contours. *SIAM Journal on Imaging Sciences*, 5(3):1022–1054, 2012.

[13] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4):321–331, 1988.

[14] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *Int. J. Comput. Vision*, 38(3):199–218, July 2000.

[15] C. Li, C. Xu, C. Gui, and M. D. Fox. Distance regularized level set evolution and its application to image segmentation. *IEEE Transactions on Image Processing*, 19(12):3243–3254, 2010.

[16] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[17] Roberto Manduchi, Sri Kurniawan, and Homayoun Bagherinia. Blind guidance using mobile computer vision: A usability study. In *ACM SIGACCESS Conference on Computers and Accessibility (ASSETS)*, 2010.

[18] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685, 1989.

[19] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12 – 49, 1988.

[20] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

[21] Qi Pan, Clemens Arth, Gerhard Reitmayr, Edward Rosten, and Tom Drummond. Rapid scene reconstruction on mobile phones from panoramic images. In *Mixed and Augmented Reality (ISMAR)*, 2011.

[22] M. Petter, V. Fragoso, M. Turk, and Charles Baur. Automatic text detection for mobile augmented reality translation. In *Computer Vision Workshops (ICCV Workshops)*, pages 48 – 55, 2011.

[23] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.

[24] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.

[25] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern recognition*, 33:225–236, 2000.

[26] S. Savarese. Space carving, Shadow Carving, Voxel coloring. In *A Tutorial in 3D photography First International Symposium on 3D Data Processing Visualization and Transmission, Padua, Italy*, 2002.

[27] Tomas Svoboda, Jan Kybic, and Vaclav Hlavac. *Image Processing, Analysis & and Machine Vision - A MATLAB Companion*. Thomson Learning, 1st edition, 2007.

[28] Yang Xin and Cheng Kwang-Ting. Ldb: An ultra fast feature for scalable augmented reality on mobile devices. In *Mixed and Augmented Reality (ISMAR)*, 2012.

[29] A. Yogi, M. Majnik, and D. Skočaj. Segmenting an object on a uniform background: comparison of segmentation algorithms. In *Proceedings of the 23rd International Electrotechnical and Computer Science Conference ERK*, pages 88–91, 2014.

[30] K. Zhang, H. Song, and L. Zhang. Active contours driven by local image fitting energy. *Pattern recognition*, 43(4):1199–1206, 2010.