Insert here your thesis' task.

Czech Technical University in Prague
Faculty of Information Technology
Department of Theoretical Computer Science

Master's thesis

Parking assistant using web cameras

Bc. Martin Keršner

Supervisor: Ing. Martin Klíma, Ph.D.

3rd May 2015

Acknowledgements

I would like to thank to Ing. Martin Klíma, Ph.D. for guiding the works on the thesis and František Hána for helpful advices. I would also like to thank to my family for support throughout my studies and my girlfriend for encouraging me when it was most needed.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on 3rd May 2015 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Keršner, Martin. *Parking assistant using web cameras.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2015.

Abstrakt

Tato práce popisuje metody rekonstrukce 3D obrazu. Metody rekonstrukce jsou porovnávány z hlediska využití pro parkovacího asistenta vybaveného webovými kamerami. Byl navržen algoritmus pro detekci objektů ve scéně a pro odhad jejich vzdáleností. Práce se dále zabývá detekcí obrubníku a detekcí člověka podle jeho dolních končetin. Jednotlivé algoritmy jsou vyhodnoceny jak z hlediska úspěšnosti detekce a přesnosti, tak i časové náročnosti.

Klíčová slova stereo obraz, SfM, detekce obrubníku, detekce lidské nohy

Abstract

Thesis explains methods for 3D reconstruction of scene. Reconstruction methods are compared in terms of application for parking assistant using web cameras. Algorithms for detection of objects in scene and estimation of their distances are proposed. Further, thesis describes methods for detection of curbs and human being based on legs. Successful detection rate and precision of particular methods is also presented in this thesis.

Keywords stereo imaging, SfM, curb detection, human leg detection

Contents

List of Figures

List of Tables

Introduction

Automobile industry experienced growth in connection with information technologies in the past few years. Renowned companies announced development of self-driving cars that happened to be objective for next decade. Self-driving cars contain various sensors enabling detection of road lines, road signs, traffic lights or obstacles which might affect safety of ride. All mentioned features have mostly been contributed to ordinary cars and gradually facilitate a car driving.

A parking assistant is also among the mentioned features that are daily exploited by millions of drivers. Currently, most of new cars have a built-in radar sensor that is able to measure distance between the sensor and object. A detection of close object helps to back a car, whereas no one is able to see right behind a car while parking. Some cars are equipped with a back up camera. A necessity and usefulness of a parking assistant has even been confirmed by US law that requires each car to possess a back up camera by 2018[1]. On the other hand, the radar sensor also has its own drawback – single purpose. There are no other scenarios for which it might be utilized.

While most of cars possess radar or laser sensor for an adaptive cruise control and collision avoidance system, Subaru proposed and implemented system based on stereo cameras. Stereo cameras preserve spatial information about scene and allow to run image processing algorithms, e.g. distinguishing between vehicles and pedestrians. It makes stereo cameras multipurpose equipment. Current enhanced digital sensors are able to see in the dark, therefore the prior advantage of radar and laser sensors decrease. Moreover, an overall price of stereo based system is generally lower[2]. According to specification of the newest Subaru stereo system called 2015 Subaru EyeSight[3], differential speed to which the system is able to respond is up to 30 mph.

[1] http://www.cnet.com/news/u-s-requiring-back-up-cameras-in-cars-by-2018
[2] http://www.subaru.com/engineering/eyesight.html
[3] http://www.planetsubaru.com/2015-subaru-eyesight-capability-revealed.htm

In this thesis we aim to create a simple parking assistant using web cameras that enable to detect a stationary object in a colliding distance, detect an edge of a potentially dangerous object, such as curb, and classify an object as human being or another object. All mentioned features are designed to be processed in real-time.

This thesis is divided into two parts: theoretical and practical. The first chapter of theoretical part describes camera models, their properties and calibration methods. The next two chapters explain state of the art methods that can be used for 3D scene reconstruction. The chapter 2 introduces stereo imaging and the chapter 3 examines structure from motion. Practical part proposes design of parking assistant in chapter 4. Further, it provides a description of implementation details (chapter 5) and includes evaluation of employed methods (chapter 6). Chapter 7 discusses results of thesis and finally the thesis is finished by conclusion.

Chapter 1

Camera Models and Calibration

An understanding of camera models is a crucial part from which other important aspects and algorithms are derived. It allows to incorporate parameters of real cameras and thus enhance acquired images. A process of calibration is employed to obtain fundamental parameters of particular camera, because these parameters vary from piece to piece.

This chapter provides a basic understanding of a *pinhole camera model* and later explains parameters of commonly used cameras and their imperfections. The last part of chapter explains a calibration of monocular camera.

1.1 Camera Models

A basic camera model, which we firstly introduce, is called the pinhole camera model. It consists of an *image* and *pinhole plane* with a small aperture in a center. This model reacts to rays coming towards to the pinhole plane. Only the rays, which pass through the aperture, are projected onto the image plane. All other rays are blocked. Due to the aperture projection is upside down than an actual object or a scene appears. The pinhole camera model defines a distance between the image plane and pinhole plane as *focal length* (f). Distance between the pinhole plane and observed point of object in scene indicates depth. Using these information we can describe the pinhole camera model by the equation

$$-x = f\frac{X}{Z}, \tag{1.1}$$

where x is a reflection on the image plane of object's point X, both expressed as length vectors. Parameter Z is a distance between the pinhole plane and the object X. Finally, f indicates the focal length.

The pinhole camera mode is not usually used for computer vision tasks. However, a derived model [1] (fig. 1.1) is utilized to simplify matters. One of the differences is that a captured object does not appear upside down

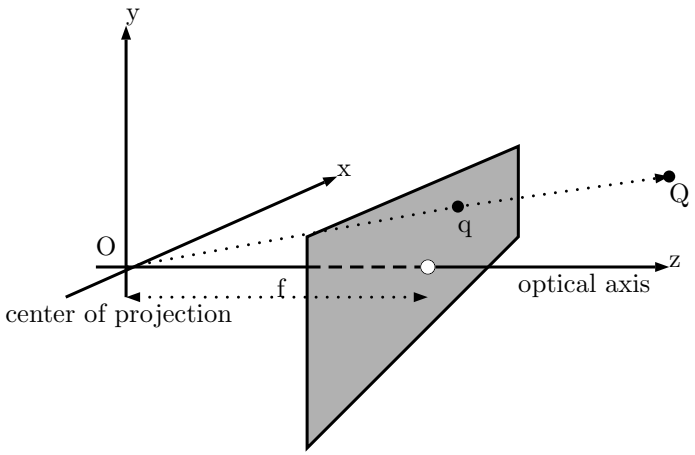


Figure 1.1: Derived camera model from the pinhole camera model.

anymore. The reason for that is because the image plane and the pinhole plane are swapped. The aperture is replaced by a *center of projection* located behind the image plane. Every ray now comes through the image plane to the center of projection. A point where the ray intersects the image plane is called a *principal point* and all of them create an image of a distant object. Since the image is rightside up, the equation of model has slightly changed to the form

$$\frac{x}{f} = \frac{X}{Z}. \tag{1.2}$$

1.2 Camera Parameters

Camera parameters can be divided into two sections: *intrinsics* and *extrinsics parameters*. The intrinsics parameters include a camera intrinsics matrix and distortion coefficients. These parameters describe a particular camera. Hence, once we calculate intrinsics parameters we can use them for that camera whenever after. On the other hand, the extrinsics parameters have to be computed for each new view of camera. The extrinsics parameters comprise of rotation and translation of camera with respect to preceding position.

1.2.1 Camera Intrinsics Matrix

A center of imager in real cameras is commonly not on the optical axis. In order to define (eq. 1.3) the model more precisely we present two new parameters c_x and c_y.

$$x = f_x \left(\frac{X}{Z}\right) + c_x, \qquad y = f_y \left(\frac{Y}{Z}\right) + c_y \tag{1.3}$$

These parameters express a shift of the optical center of imager. Another refinement of the model is because of shape of pixels in imager. Their shape occurs more often as rectangular than square one. Therefore, a computation of focal length ($f_x = Fs_x$, respectively $f_y = Fs_y$) incorporates a physical focal length F and a size of the individual imager elements (s_x, respectively s_y).

One of the conventions of image plane represention is to transform points to *homogenous coordinates*. The transformation causes an enlargement of dimensions by one. Homogenous coordinates enable to express the same point in various ways, because points with proportional coordinates belong to the same point. The conversion from homogenous coordinates back to former ones can be performed by dividing them through by the last coordinate value.

Since homogenous coordinates are employed, the point $q = [x\ y\ z]^T$ in the image plane consists of three coordinates. The projection of the points in the scene into the image plane can be calculated by projective transformation defined as $q = MQ$, where M (eq. 1.4) denotes a *camera intrinsics matrix* [2] and Q is the point in the scene.

$$M = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

The only camera property, which have not been mentioned yet, is skew of the pixel array in the imager (denoted as s). For the sake of simplicity it can be safely set to zero [3].

1.2.2 Distortion Coefficients

The pinhole camera model is not frequently used for recording of images. This model cannot absorb enough light per unit of time and exposure lasts unpleasantly long time. It results in using cameras with lenses instead. Unfortunately they also have their own drawbacks, called *lens distortions*. Phenomenon of the lens distortion appears mostly at the cameras including cheap lenses.

There are two main types of lens distortions: *radial* and *tangential*. The distortion coefficients encode lens imperfections within 5 coefficients and can be used for image correction called an *undistortion*.

Radial distortions are caused by shape of lens, because manufactured lenses are imperfect. This distortion introduces a "barrel" effect which defects the image mainly at edges. Pixels closer to the center of imager are less affected. The distortion in the center of imager is 0 and becomes larger with distance from the center. The radial distortion of x coordinate can be eliminated using equation 1.5, where k_i denotes terms of a Taylor series of distortion function. The parameter r expresses the distance from center of imager for a particular pixel. The correction of y coordinate is computed in a similar manner.

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{1.5}$$

The tangential distortion is a result of mechanical inaccuracy when the lens and imager are not parallel. A distorted image can be recovered by equation 1.6, where p_1 and p_2 describe tangential distortion coefficients.

$$\begin{aligned} x_{corrected} &= x + [2p_1y + p_2(r^2 + 2x^2)] \\ y_{corrected} &= y + [p_1(r^2 + 2y^2) + 2p_2x] \end{aligned} \tag{1.6}$$

1.3 Calibration

The calibration is a process of finding both camera parameters: intrinsics and extrinsics. The intrinsics parameters can be used in order to correct a distorted image. Rotation and translation information, contained in the extrinsics parameters, gives us more or less precise movement estimation of camera within scene. The calibration is an essential part of algorithms requiring usage of calibrated camera and their performance greatly depends [2] on calibration accuracy. In this section we shortly describe two types of calibration.

A first method [3] keeps calibration as simple as possible. It assumes square straight pixels, alignment of imager at the center of the image and no lens distortions. We should also emphasize that without any modifications proposed calibration works only for the same image resolution and orientation as was employed for calibration. To setup a calibration environment we need flat object with known dimensions X, Y and determine a distance Z from object to camera. If we want to meet our expectations about calibration we have to adjust a camera and calibration object to be parallel. Then we take a picture of object and measure a size of object x and y in pixels. Partial computations are substituted to early introduced camera intrinsics matrix.

$$M = \begin{bmatrix} \frac{x}{X}Z & 0 & \frac{col}{2} \\ 0 & \frac{y}{Y}Z & \frac{row}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{1.7}$$

More robust calibration approach [4] comes with an idea of detection planar pattern in different positions. Simple chessboard [1] has become one of the options. It is recommended not to use chessboard with the same grid in both dimension due to ambiguity of position. The chessboard pattern has the advantage of simple detection of corners that are symmetrically aligned. Fundamental preprocessing steps, like equalizing histogram or thresholding, typically precede the calibration process.

The calibration process employs a mapping of pattern on imager of camera. This mapping is called a *planar homography* and is formulated as

$$\tilde{q} = sH\tilde{Q}, \tag{1.8}$$

where H represents a homography matrix, $\tilde{q}$ is a point on imager and $\tilde{Q}$ is its corresponding mapping. The variable s denotes scale factor. The homography matrix can be decomposed into two parts: camera intrinsics matrix M (eq. 1.4) and extrinsics matrix W containing a physical tranformation. The extrinsics matrix is composed of a rotation matrix and translation vector. As we see from equation 1.8 there is no need to know camera intrinsics matrix in order to compute homography. Therefore, we can focus on another six unknown parameters: three angles from the rotation matrix and three offsets from the translation vector. From each image with detected pattern four points (x, y) can be derived. Hence, we can compose eight equations which are ample to calculate six unknown parameters of each view. However, as result of calibration there should be camera intrisics matrix as well. That adds to our set of equations another nine parameters introduced in section 1.2 that need to be served.

Our sought parameters are deduced gradually acording to listed order: camera intrinsics parameters, rotations and offsets and finally lens distortions. A derivation [5] is built upon knowledge that rotation vectors are orthonormal. It leads to constraint $h_1^T M^{-T} M^{-1} h_1 = h_2^T M^{-T} M^{-1} h_2$ which meets requirements. The part of the equation $M^{-T}M^{-1}$ produces 3-by-3 symmetrical matrix B (eq. 1.9) which is later reduced to six-dimensional vector b.

$$B = \begin{bmatrix} \frac{1}{f_x^2} & 0 & \frac{-c_x}{f_x^2} \\ 0 & \frac{1}{f_y^2} & \frac{-c_y}{f_y^2} \\ \frac{-c_x}{f_x^2} & \frac{-c_y}{f_y^2} & \frac{c_x^2}{f_x^2} + \frac{c_y^2}{f_y^2} + 1 \end{bmatrix} \tag{1.9}$$

A final step consists of solving several equations with elements from homography matrix H and vector b on the other side. Particular results are then calculated [5] with simple formulas. Sometimes it can happen that found rotation matrix R does not satisfy $R^T R = RR^T = I$[5] condition and correction is required. A common solution employs factorization of rotation matrix using SVD. A unitary matrix of decomposition is replaced with an identity matrix and multiplied again to create a rotation matrix.

[4] Both $\tilde{q}$ and $\tilde{Q}$ are expressed using homogenous coordinates as follows $\tilde{q} = [x\ y\ 1]^T$ and $\tilde{Q} = [X\ Y\ Z\ 1]^T$.

[5] I denotes identity matrix.

(a) (b)

Figure 1.2: Figure displays comparison of image (a) before and (b) after undistortion process. Image courtesy of [1].

Calculations in preceding parts of camera calibration assumed no distortions. However, these parameters have to be retrieved in order to undistort image (fig. 1.2). That can be performed [6] by following equation

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} 2p_1 x_d y_d + p_2(r^2 + 2x_d^2) \\ p_1(r^2 + 2y_d^2) + 2p_2 x_d y_d \end{bmatrix}, \quad (1.10)$$

where x_d, y_d is distorted position on image plane and x_p, y_p determines a corrected position of distorted point.

A question remains how many images it is necessary to perform sufficiently accurate calibration. According to [1] one is able to compute calibration within two images. Nonetheless, they advice to utilize at least ten images of rather larger chessboard in different positions.

Chapter 2

Stereo Imaging

A first stereo vision system which we could have encountered was through our eyes. Due to our stereo visual system we can perceive the third dimension called depth. Basically, each eye captures an image of scene from different point and then they are put together to induce a depth perception. Throughout the chapter we presume that images have already been undistorted.

This chapter explains stereo imaging methods, namely a *rectification*, *correspondence* and *triangulation*, leading to acquisition of depth from stereo pair of images. However, firstly we introduce *epipolar geometry* as fundamental part of stereo imaging.

2.1 Epipolar Geometry

The aim of stereo imaging is to find corresponding points in both images, left and right one[6]. If we approach searching for these points without any constraints, we end up with inefficient solution, because we have to always search through all image. Fortunately, there is a way using epipolar geometry which saves computation time.

The epipolar geometry introduces a model of two cameras depicted in figure 2.1. The model consists of two centers of projection, O_l and O_r, related projective planes, Π_l and Π_r, and lastly point P from scene. Projection of point P onto image planes is expressed by p_l for the left image plane and p_r for the right one. The most fundamental objects are *epipoles*, *epipolar lines* and *epipolar planes* The epipoles, e_l and e_r, lie on its image planes accordingly to their side and are characterized as images of the center of projection of the camera from the other side. The epipolar lines ($e_l p_l$, respectively $e_r p_r$) intersect epipoles and projection points and form together with point P the epipolar plane.

[6]In this thesis we assume horizontal alignment of stereo cameras, although algorithms work fine for vertical adjustment as well.

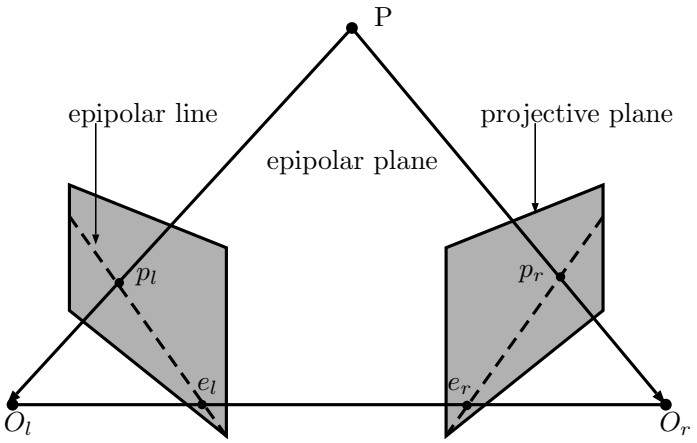


Figure 2.1: Representation of epipolar geometry

It is impossible to estimate depth of point P with only one image, because that point can be located anywhere on the line $O_r p_r$, respectively $O_l p_l$. However, that line is projected on the other image plane and matches its epipolar plane. Due to that there is relation between point p_l (respectively p_r) in one image plane and epipolar line $e_r p_r$ (respectively $e_l p_l$) in the other one. This cognation is referred as *epipolar constraint*.

The epipolar geometry can be represented by *fundamental matrix* F [7] which defines a relation (eq. 2.1) between pixel coordinates of stereo images.

$$p_l^T F p_r = 0, \tag{2.1}$$

The fundamental matrix includes information about rotation and translation of right camera relatively to left one[7]. It also contains intrinsics parameters of cameras.

Another way of formulating epipolar geometry is by *essential matrix* E [7]. The essential matrix does not possess any information about cameras, hence we know only geometry of two cameras. Due to that minor difference[8], the essential matrix can be easily derived from fundamental matrix as

$$E = M'^T F M, \tag{2.2}$$

where M is camera intrinsics matrix (eq. 1.4).

The last important fact about epipolar geometry is an *epipolar ordering constraint*. This constraint relates to two horizontally aligned points. The order of these points is preserved. Therefore, if they are both visible in one

[7]We assume that left camera has no rotation and translation because it is aligned with world coordinates.

[8]In case we rectify images (sec. 2.3) and divide points by focal lengths, fundamental and essential matrix become equal [1].

image and occur in specific order, that order is preserved even for another image where they both also appear.

2.2 Stereo Calibration

With the knowledge of epipolar geometry we can proceed to stereo calibration; a method estimating intrinsics parameters of each camera and rotation and translation of right camera relative to left one.

Matching points positioned on planar area are expected as the input of stereo calibration. In the first phase each camera is calibrated individually according to explanation in the section 1.3. Originally both cameras have their own coordinates, denoted with subscript l, respectively r. Therefore, an arbitrary point P have coordinates P_l for left image and P_r for right one. Using a rotation matrix R_l and translation vector T_l, respectively R_r and T_r, which we have obtained at the first phase, we can declare their relations as $P_l = R_l P + T_l$ and $P_r = R_r P + T_r$. Recombination of these equations leads to

$$\begin{aligned} R &= R_r (R_l)^T \\ T &= T_r - R T_l, \end{aligned} \tag{2.3}$$

where R is a rotation matrix of right camera relative to left one and T is its translation. In the second phase equation 2.3 is solved, and thus rotation matrix R and translation vector T are retrieved. Individual subresults vary because images used to stereo calibration are usually affected by noise. One of the possible proposed solutions [1] is to employ Levenberg-Marquardt iterative algorithm which minimizes an error.

2.3 Rectification

Section 2.1 revealed the epipolar constraint that facilitates searching for corresponding points. However, it does not guarantee common direction of all epipolar lines, and thus computation of corresponding points is still not satisfactory. The rectification method transforms left and right image in a way that they are both mutually row-aligned. After images are rectified, corresponding points lie on the same row of image.

2.4 Correspondence

The correspondence is method which relates points in one image to points in the other one. Each couple of corresponding points differ in position on image planes and their relative distance between them is known as *disparity*. The disparity can be assigned only to points which are located on both image planes. Otherwise they are called *occluded*. The outcome of correspondence is

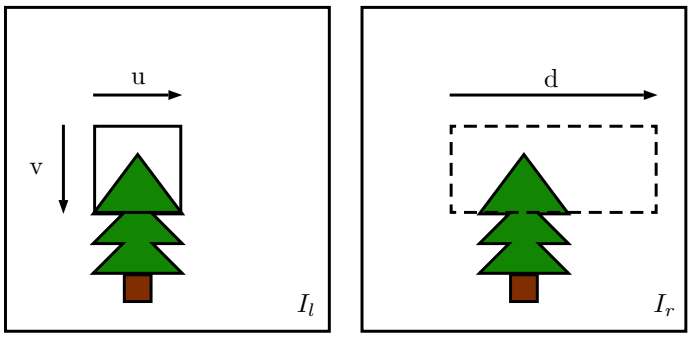


Figure 2.2: Block matching representation

disparity map that stores information about distance between all corresponding points. The next section 2.5 depicts how to employ disparity map in order to reconstruct 3D scene.

Methods [8] applied to obtain disparity map are divided to local and global. The local methods target on neigborhoods of examined points. Hence, it poses disadvantage determined by the size of neighborhood. There can also arise an issue with ambiguity of compared regions e.g. uniform texture or occluded regions. The global methods are not limited by neighborhood, therefore they can perform better on ambiguity regions. On the other hand, computation time of global methods is higher [8].

2.4.1 Local Methods

The local methods are further divided into three categories: *block matching*, *gradient methods*, and *feature matching*[9]. The block matching (pseudocode of left-to-right matching in algorithm 1) introduces a *template* which stands for a block of image with a point of interest in the middle. The template is then compared with blocks contained in *search region* of the other image. Due to epipolar constraint and rectification, the search region is row-aligned with the template. Figure 2.2 displays a process of block matching where the I_l determines left image and I_r right one. Coefficients u and v define the size of neighborhood and d expresses maximum predefined disparity.

There are many different metrics [8] which can be used for comparison of blocks. *Sum of absolute differences* (SAD)

$$\sum_{u,v} |I_l(u,v) - I_r(u+d,v)| \tag{2.4}$$

[9]Gradient methods and feature matching are addressed in chapter 3.

and *sum of squared differences* (SSD)

$$\sum_{u,v}(I_l(u,v) - I_r(u+d,v))^2 \tag{2.5}$$

belong to basic statistical methods.

Another approach is *rank* [9] which first performs local nonparametrical transformations on input images and then apply any of statistical methods (e.g. SAD) mentioned earlier. The local nonparametrical transformation is performed for each point that is replaced by a number of neighbor points with less intensity than a center point possess. By using this transformation we lose part of descriptive information of image, and thus decrease a probability of correctly found correspondence. The method called *census* [9] is considered as eligible replacement. Census also applies local nonparametric transformation, but preserves geometrical information about vicinity. The output of particular points are binary votes which are composed of comparisons between the center point and each of points in the neighborhood. When the center point is higher, then census returns 1, otherwise 0. This binary encoding is penalized by enlargement of dimensionality by factor of neighborhood size. Finally, generated binary strings are matched using Hamming distance.

Algorithm 1 Block matching

```
1:  M ← auxiliary matrix with the size of image and large initial values
2:  D ← disparity matrix with the size of image
3:
4:  for k ← 0 to maxDisparity do
5:      I_r ← ShiftToRight(I_r, k)
6:      C ← Compare(I_l, I_r)
7:
8:      for all i ← 0 to rows do
9:          for all j ← 0 to cols do
10:             if C[i, j] < M[i, j] then
11:                 M[i, j] ← C[i, j]
12:                 D[i, j] ← k
```

2.4.2 Global Methods

As we have mentioned earlier, the global methods are not constrained by local neighborhood. In contrast, this great asset of calculation superior disparity maps requires longer computation time. We shortly introduce *dynamic programming* [10] and *graph cut* [8].

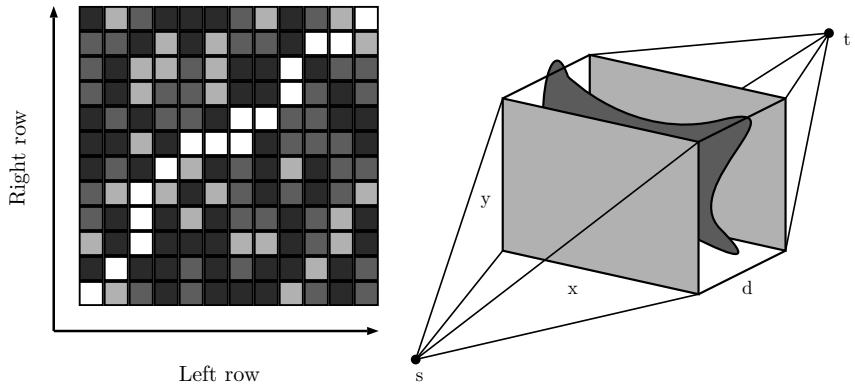
Dynamic programming decomposes a problem to subproblems which are then solved in continuous stages and utilized to obtain a final disparity map. Global cost function based on epipolar ordering constraint is determined as

minium-cost path of each pair of corresponding rows of images. All possible combinations of pixel correspondences create a square matrix with dimensions equal to width of image. For each pair the cost of potential match is computed and then selected that with the lowest value. Thereafter, the costs of individual paths are summed in order to get the cost of optimal path.

The latter global method is called graph cut and seeks for a maximum flow in a graph. The graph cut is defined by directed graph $G = (V, E)$, set of vertices V and set of edges E. The main vertices are *source*, denoted as s, and t expresses *sink* . The rest of them is defined as three-dimensional vectors

$$V^* = \{(x, y, d), x \in [0, x_{max}], y \in [0, y_{max}], d \in [0, d_{max}]\}. \tag{2.6}$$

The flow starts from s and leads to all vertices V^* with coordinates $(x, y, 0)$. Each of them is then connected to vertices V^* with coordinates (x, y, d_{max}). Finally, they are linked to t. Each edge has assigned *flow capacity* that is computed as cost of connected adjacent nodes. The cost is directly proportional to amount of flow which can be sent from s to t through particular edge. To obtain vertex correspondences we perform cut of edges between connected vertices that have the lowest capacities.



(a) (b)

Figure 2.3: Figure a displays matrix, computed by dynamic programming, with combination of all pixels in two corresponding rows. Figure b depicts model of graph cut.

2.4.3 Evaluation

A performance of particular correspondence methods varies and depends on their parameters. In order to achieve quantitative evaluation of these methods it is required[10] to possess a *ground truth*. The ground truth is comprehended

[10]There are also methods [11] which evaluate correspondence methods without ground truth, however we do not consider them in this thesis.

as the best result that could be retrieved by algorithm. It is basically the disparity map with correct disparities. Among the other evaluation methods [10] there are two called *root-mean-squared error* (RMSE)

$$R = \sqrt{\frac{1}{N}\sum_{x,y}|d_C(x,y) - d_T(x,y)|^2} \tag{2.7}$$

and *percentage of bad matching pixels*

$$B = \frac{1}{N}\sum_{x,y}(|d_C(x,y) - d_T(x,y)| > \delta_d), \tag{2.8}$$

where N denotes total number of pixels, d_C is computed disparity map and d_T is ground truth disparity map. The latter method includes extra disparity error tolerance δ_d. This constant sets a minimal correspondence difference that is counted as error.

2.5 Triangulation

The last step of stereo imaging, named triangulation, is utilized to reconstruct 3D scene. The disparity and depth are inversely proportional [1], and therefore one can employ disparity map for depth derivation. Equation 2.9 represents that conversion where f[11] denotes focal length of cameras and T is distance of centers of projection between cameras. Objects closer to camera have smaller disparity and vice versa. Due to inverse relation, depth resolution is high only for objects closer to camera and decreases with distance.

$$Z = \frac{fT}{x_l - x_r} = \frac{fT}{d} \tag{2.9}$$

[11] We expect the same focal length for both cameras of stereo rig.

Chapter 3

Structure from Motion

Structure from motion (SfM) [12] is another method for reconstruction of 3D scene. In contrast to stereo imaging there is no need to have stereo rig with dependent calibrated cameras. SfM requires only a couple[12] of independently calibrated cameras, thus there is available camera intrinsics matrix created for each of them. As an alternative to pair of cameras there can be employed only single monocular camera that captures a scene. Continously taken images replace a necessity of the second camera, however this option is constrained by minimal movements in scene, otherwise reconstruction could become inaccurate.

While the stereo calibration uses chessboard for estimation of translation and rotation between cameras, SfM needs to find own specific matching points that can be utilized for computation of fundamental matrix. An accuracy and correctness affect quality of estimated spatial relation between cameras or between captured images.

This chapter focuses on different approaches that find corresponding points in pair of images. There exists a couple of methods used for these purposes, but we introduce more thorougly only *feature descriptors* and *optical flow*. Further we explain possible ways of evaluation and shortly describe process of 3D scene reconstruction.

3.1 Local Feature Detection and Extraction

The aim of local feature detection and extraction is to find reliable significant features in given images and describe their neighborhood in manner that they are easily matched with other features. The feature extractors are supposed to provide results that are repeatable and precise [13]. Therefore, the found features of the same object illustrated on the two different images should be

[12] It is also possible to utilize more than two cameras, however we do not consider this option in this thesis.

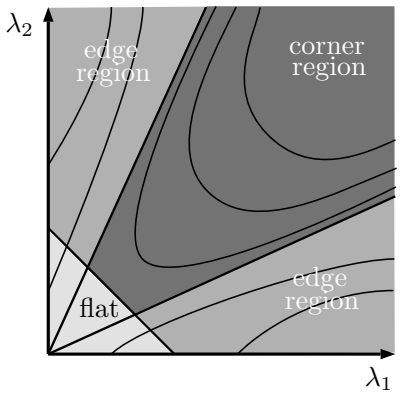


Figure 3.1: Figure depicts relation between eigen values λ_1 and λ_2 from Harris corner detection. If one of these eigen values is much larger than the other, then examined region is edge. Approximately similar large eigen values determine corner region. In all other cases flat region is found.

the same. Moreover, the features obtained from different parts of image should be distinctive to each other.

3.1.1 Feature Detection

The feature detection is preceding step before the descriptor extraction. In this step we search for such features that can be found under varying image conditions, different viewpoints changes and even in noisy images. The features that are found should not be dependent on translation or rotation of image. In conclusion, there is limited number of points that satisfy these conditions. Among other feature detection algorithms there is a *Harris detector* [13] which is introduced in following paragraph.

The Harris detector is a method used to find corners as very significant features. The detector computes a second-moment matrix

$$C = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \tag{3.1}$$

where the left part of equation represents computing a weighted sum (weight is denoted as w) and the right part is matrix composed of multiplication of image derivatives. The matrix C is computed for each block of image. Using equation 3.2 and 3.3 the eigenvalues of matrix C can be computed and based on them one can determine the type of block (fig. 3.1). The blocks that can be recognized are edge, corner and flat region. If the both eigenvalues are large, then the examined block is corner.

$$det(M) = \lambda_1 \lambda_2 \tag{3.2}$$

$$trace(M) = \lambda_1 + \lambda_2 \tag{3.3}$$

3.1.2 Descriptor Extraction

The descriptor extraction is a way how to obtain and encode information around a point of interest in order to be matched only with identical point in the other image. The extracted descriptors aim to be rotation and scale invariant and concurently illumination and noise resistant. Such properties of descriptors significantly increase a number of correct point correspondences. One of the famous descriptor extraction algorithms is Scale Invariant Feature Transform (SIFT) [14].

SIFT is a method that includes even keypoint detection, however only descriptor extraction is explained in this thesis. Keypoint does not have to be necessarily aligned within pixel coordinate. If it occurs, neighborhood pixels need to be interpolated. This first step consists of finding dominant orientation of its neighborhood and crop accordingly around it. The result is 16×16 px region which is then divided to 16 square blocks. Thereafter, for each block histogram of gradient orientation is computed. Histogram is composed of 8 bins. In order to reward pixels closer to keypoint and at the same time punish farther keypoints gaussian, weighting function is employed. Calculated histograms are lined up next to each other and create feature vector of length 128 bins. The final step is normalization of descriptor.

The algorithm explained above suffers from rotation and illumination variancy. To meet the previously set goals rotation of keypoint should be subtracted from each gradient orientation. The illumination variancy is handled by thresholding and normalization of final vector.

3.1.3 Descriptor Matching

The most straightforward approach of descriptor matching is to gradually compare each descriptor of one image with all descriptors at the other image. For each couple of feature vectors distance metric such as euclidian or manhattan distance is computed.

SIFT utilizes *ratio test* to avoid false matches and algorithm called a *best-bin-first* that reduces computation time when searching for nearest neighbor of keypoint.

3.2 Optical Flow

Optical flow is a method that estimates motion of particular pixels between two images. It is mainly used for motion detection and tracking objects in scene. However, it also might be utilized for reconstruction of 3D scene,

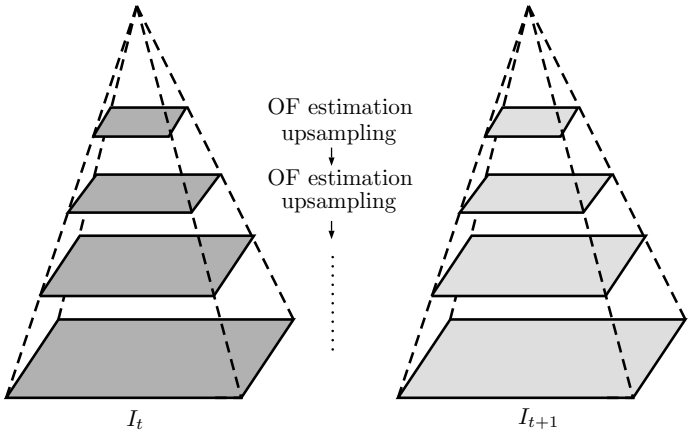


Figure 3.2: Firstly, optical flow is calulated for the most downsampled image, denoted in the top layer of pyramid. The next layer exploits result from previous layer, refines it by estimating optical flow on image with higher resolution. In the same manner optical flow is improved until the layer with the original image is reached.

because optical flow keeps information about motion that is necessary[13] for such task. Each pixel has assigned velocity determining a direction where the pixel moved from previous to current image. If one performs motion estimation on all pixels of image, it is called a *dense optical flow*. It is not easy to calculate dense optical flow due to ambiguity pixels in image. An example of ambiguity could be a block with solid color. In order to tackle with this issue, dense optical flow algorithm must employ methods that interpolate ambiguity pixels. Finally, this leads to requirement for longer computation time.

Another type of optical flow is a *sparse optical flow*. Points of image, whose motion is supposed to be estimated, are input for sparse optical flow. These points should be selected in favor simple and unique detection and recognition. Distinguishable corners might be considered as realiable points. In the next subsection we present the most popular sparse optical flow method called *Lucas-Kanade optical flow*.

3.2.1 Lucas-Kanade Optical Flow

Lucas-Kanade optical flow [15] describes examined points by information about their local neighborhood. Algorithm assumes that velocity does not exceed neighborhood around point of interest, otherwise it cannot estimate optical flow for such points. The solution that is able to estimate optical flow even for large motions utilizes pyramidal approach (fig. 3.2). At first, the algorithm is

[13] Objects closer to camera shift more than farther objects.

applied to downsized image, which lacks the details, however blocks of neighborhood cover larger parts of image. Thereafter, the algorithm calculates at gradually larger images that possess precise details. At each layer of pyramid previous result is refined until the final layer is reached.

Lucas-Kanade algorithm itself without pyramidal approach is based on three assumptions: *brightness constancy*, *temporal persistence* and *spatial coherence*. The first assumption means that pixel does not change intensity over time. Temporal persistence was already mentioned shortly. It requires small motions between images. The third assumption, spatial coherence, introduces relation between points in scene and their projections on image plane. Adjacent points in a scene remain adjacent even after projection on image plane.

Mathematical equation expressing Lucas-Kanade optical flow is

$$I_x u + I_y v + I_t = 0, \tag{3.4}$$

where I_x (respectively I_y) is a spatial derivative across image in x-dimension (respectively y-dimension) and I_t is a derivative between images over time. A component u characterizes velocity in x-dimension, respectively a component v is for y-dimension. The equation contains two unknowns for each pixel, thus it is not possible [1] to solve them individually in order to retrieve both motion components. One is able to obtain a line described by equation 3.4. Therefore, motion componenents could be solved only for motion that is perpendicular to that line. This problem is known as *aperture problem* (fig. 3.3) and arises from observation through a small aperture. If objects are larger than the aperture, their edges occur most of the time, however corners do not. The edges itself are not sufficient to estimate correct motion of object. Spatial coherence enables to surpass the aperture problem because we assume that the motion in neighborhood is consistent. For each examined pixel we can employ all pixels from its neighborhood and lay set of equations

$$\underbrace{\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix}}_{A} \begin{bmatrix} u \\ v \end{bmatrix} = \underbrace{\begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_n) \end{bmatrix}}_{b}, \tag{3.5}$$

where p denotes a pixel and n is a total number of pixels in neighborhood. Set of equation is solved using least-squares minimization, therefore final solution can be formulated as

$$\begin{bmatrix} u \\ v \end{bmatrix} = (A^T A)^{-1} A^T b. \tag{3.6}$$

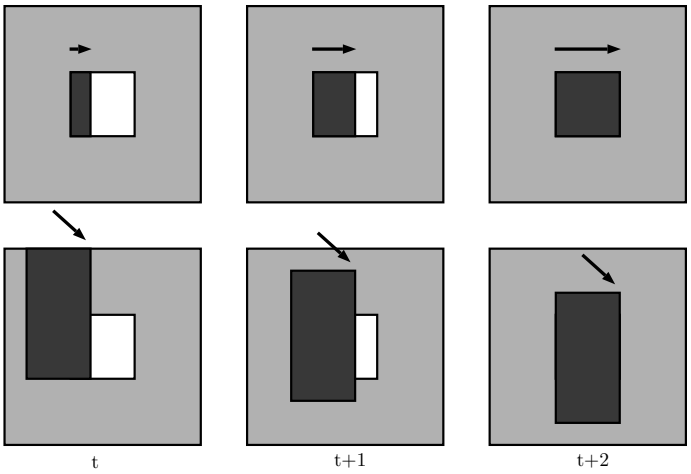


Figure 3.3: Figure explains an aperture problem. The first row denotes view through small aperture behind which object is moving. This movement appears as shift in x-dimension, however the real motion, as it can be seen at the second row, is actually along the diagonal direction from top left to bottom right.

3.3 Evaluation

A high precision in correspondence of descriptors and estimation of optical flow is mandatory in order to achieve accurate 3D scene. Since the methods used for structure from motion possess many options of adjustment, evaluation of individual settings becomes necessary. We assume that we have ground truth of optical flow.[14]

Fundamental evaluation methods [11] are *L2 endpoint error*

$$E_{ep2}(\mathbf{u}, \mathbf{u}^*) = \sqrt{(u - u^*)^2 + (v - v^*)^2} \tag{3.7}$$

and *L2 endpoint error*

$$E_{ep1}(\mathbf{u}, \mathbf{u}^*) = |u - u^*| + |v - v^*|, \tag{3.8}$$

where $\mathbf{u} = (u, v)$ denotes a calculated optical flow and $\mathbf{u}^* = (u^*, v^*)$ is a ground truth. Another evaluation metric called *angular error* measures error in spatio-temporal directions.

$$E_{ang}(\mathbf{u}, \mathbf{u}^*) = acos\left(\frac{\mathbf{u}^T\mathbf{u}^*}{|\mathbf{u}||\mathbf{u}^*|}\right). \tag{3.9}$$

[14] Ground truth for optical flow can be employed even for evaluation of feature descriptors.

3.4 Scene Reconstruction

A final phase of structure from motion is scene reconstruction which covers obtaining rotation and traslation between cameras and triangulation. The rotation and translation is retrieved from essential matrix and the whole process is explained in section 1.3, 2.1 and 2.2.

In a nutshell corresponding matches are employed to compute the fundamental matrix (eq. 2.1) that is converted to essential matrix (eq. 2.2) with knowledge of camera intrinsics matrix (eq. 1.4). In the next step rotation and translation of the second camera is derived from essential matrix using SVD. The first camera is considered static, therefore there is no rotation or translation. The rotation R and translation matrix T compose a *projection matrix* $P = [R|T]$. For each point of scene projection matrix satisfies equation $x = PX$ (respectively $x' = PX'$ for the other image) where x (respectively x') is a point from image and X (respectively X') a is 3D point of scene. Corresponding points in normalized coordinates are put to set of equations and solved for particular points in scene.

Chapter 4

Design

The parking assistant is supposed to detect a stationary object in a colliding distance, detect an edge of a potentially dangerous object, such as curb, and classify an object as human being or another object. This chapter describes design of three main parts which fulfill mentioned demands of parking assistant and thoroughly explains the chosen method for obtaining depth perception.

4.1 General

An equipment of parking assistant consists of stereo rig and computation unit. Stereo rig contains two cameras[15] that are horizontally aligned. Stereo cameras are fastened to back side of a car (fig. 4.1), therefore they can watch area behind a car. Acquired images by stereo cameras are transfered to the computation unit where all these information are processed and adequate response is returned.

The parking assistant operates in three modes: calibration, standby and running mode. The calibration mode runs only in order to obtain or refine cameras' intrinsics and extrinsics matrices and detect distortions of cameras' lenses. More about calibration can be found in subsection 4.4. The standby mode is actively running on background and waiting for interruption. There is no output of this mode and it is used only for saving of computation time and switching to and from running mode. Switching to running mode can be triggered by one of two possible conditions. The first one requires interaction of driver that turns manually parking assistant on, otherwise parking assistant remains in standby mode. The second condition is met by engaging a reverse gear.

While the parking assistant is in running mode, the stereo rig captures images and computation unit processes them. The parking assistant allerts to

[15]The cameras should be able to acquire similar images in terms of focus and pixel resolution. Usage of cameras of the same brand and model are considered as more reliable approach.

objects in a colliding distance, detects curbs and determines if human appears in area behind a car. Termination of running mode is reverse operation to interruption of standby mode, thus turning manually parking assistant off by driver or disengaging a reverse gear.

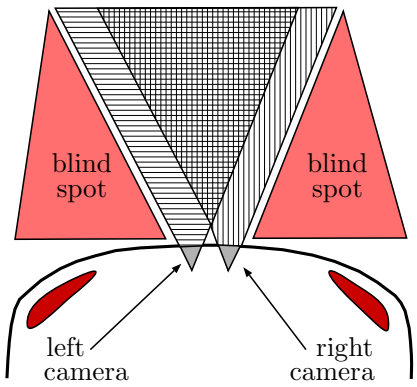


Figure 4.1: Top view on back side of a car shows blind spots of stereo rig.

4.2 Constraints

Proposed parking assistant is designed to work under particular set of constraints. This section explains them and implies possible solutions.

The parking assistant is based on processing of images acquired from real world. The overall confidence of parking assistant is strongly correlated to quality and variance of images taken in different lighting conditions (direct sunlight, cloudy, dimness, etc.). Possible solutions[16] for managing all these different cases reach outside the scope of this thesis and thus we assume[17] for all our performed evaluations stable lighting conditions.

As the name suggests, the parking assistant is intended only for parking purposes. It is not designed to increase safety while backing a car. This constraint arises together with constraint on upper limit of car speed[18]. Since the parking assistant does not assume high speed of a car, the cameras also do not have to capture a large number of frames per second. Therefore, these constraints reduce the complexity of calculation in computation unit. On the other hand, only by increasing of computational power parking assistant can perform at higher speeds.

[16]The solution for bad lighting conditions could be high dynamic range imaging, that is able to capture good quality images even at night.

[17]Even though we are not able to cover all cases, we still try to minimize their influence.

[18]The upper limit of car speed is derived from camera's FPS and duration of performing algorithms. Section 6.7 derives maximum achievable FPS.

All information that can be employed comes through stereo cameras. Ordinary cameras capture a scene with narrow angle of view[19]. It leads to constraint of the size of blind spot (fig. 4.1) behind a car and maximum observation distance. These two constraints are proportional.

4.3 Comparison of 3D Reconstruction Methods

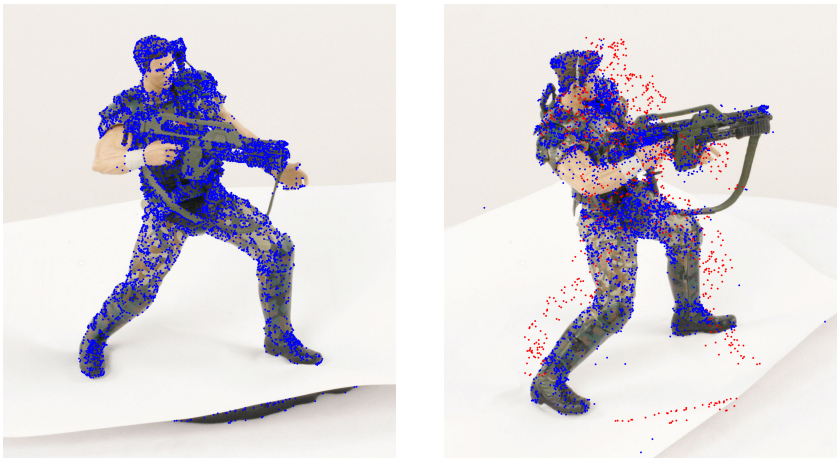
Theoretical part of thesis introduced two types of 3D reconstruction. This section is addressed to selection based on experimental comparison of pros and cons between these two approaches.

Struture from motion possess several advantages which are applicable to more sundry situations, however the parking assistant is rather bounded scenario. Together with these advantages SfM brings complexity, moreover, it does not depend on whether one uses single camera or couple of them. The movements of single camera between capturing images leads to complicated computations that need to be performed at each change. At the same time, if one decides to exploit more cameras whose position is fixed, stereo imaging approach seems to be more realiable approach. The reason is that even with pair of stereo cameras position has to be fixed in a fashion that allows to reconstruct 3D scene, but guarantees that view contains minimum of occluded regions. Therefore, in order to capture images that contain mutually little of occluded regions, some sort of decision while positioning of cameras has to be done.

In the next several paragraphs we perform SfM on pair of images (fig. 4.2) and discuss results. The examined pair of images could be considered as images captured by two cameras or as continuously taken images with single camera. To obtain 3D structure of object we employ the sparse and dense optical flow (explained in section 3.2). Figure 4.2 shows feature correspondences (blue dots) and features that are not utilized (red dots) due to high error. The sparse optical flow is calculated using precomputed features. A result of 3D reconstruction after undistortion of points, computation of essential matrix and triangulation is shown in fig. 4.3. Particular projections are rotated counterclockwise around vertical axis. One can notice that some projections are disproportional, and therefore it is hard to recognize what kind of object is displayed. This also applies to object recognition using suitable algorithms. As we tested further, acquired models seem to have low depth precision for all tested cases of methods `goodFeaturesToTrack` and `calcOpticalFlowPyrLK`[20]. Another disadvantage is that sparse model of object does not contain information about surface between the closest points. Therefore, one can only presume that there are no high variances of depth.

[19] http://en.wikipedia.org/wiki/Angle_of_view

[20] Both methods `goodFeaturesToTrack` and `calcOpticalFlowPyrLK` are part of the OpenCV library.



(a) (b)

Figure 4.2: Figure displays two images of soldier taken from different angles. The angle difference is 45°. The blue dots on the left image represent found distinctive features using method from OpenCV called `goodFeaturesToTrack`[21]. There is about 5 thousand features. The blue dots on the right image are found correspondences (using OpenCV method `calcOpticalFlowPyrLK`[22]) to the features on the left image. The red dots are either features which could have not found correspondences or they are among 20 % of the features with the highest estimated error. Image courtesy of [16].

The method of dense optical flow does not rely on selected features, but endeavors to estimate optical flow for all pixels. This approach should overcome disadvantage of sparse optical flow which allows to use only selected features for 3D reconstruction. Figure 4.4 represents result of reconstruction using dense optical flow (OpenCV method `calcOpticalFlowFarneback`[23]). In order to downsize number of points employed for computation of fundamental matrix, only each sixth row of image was retained. A resulting 3D model is composed of rather continuous points if we compare it with model created using sparse optical flow. However, only right arm and head of soldier possess nearly correct depth estimation. Using other combinations of parameters of dense optical flow did not lead to significantly better results and accuracy of all 3D reconstructions were more or less similar.

It should be also mentioned that resulting inaccuracies may not be caused by estimation of optical flow, but rather in some of the following steps of 3D reconstruction. The undistortion coefficients are provided to pair of images (fig. 4.2), therefore there should not be the problem hidden. Searching for fun-

[21] `maxCorners: 5000; qualityLevel: 0.001; minDistance: 3`

[22] `winSize: 40; maxLevel: 3`

[23] `pyr_scale: 0.5; levels: 3; winsize: 40; iteration: 20; poly_n: 5; poly_sigma: 1.2; flags: OPTFLOW_USE_INITIAL_FLOW`

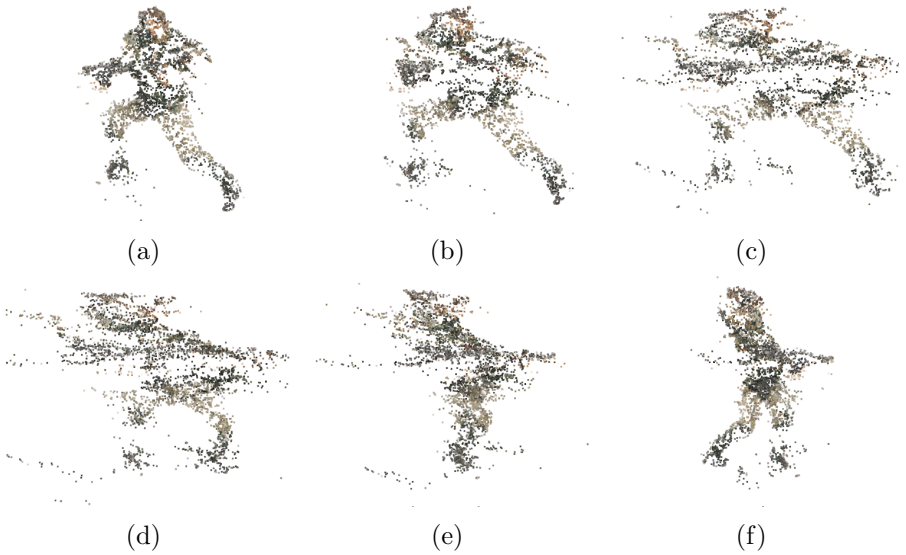


Figure 4.3: Figure depicts rotation of object reconstructed from pair of images (fig. 4.2) using sparse optical flow. Images through (a) to (f) represent counterclockwise rotation of object around vertical axis.

damental matrix was performed using OpenCV method `findFundamentalMat`, thus we do not consider it as weak part of reconstruction algorithm. According to [7] there are four possible camera matrices, but in any of our experiments we did not find camera matrix that would provide decent reconstruction.

Stereo imaging requires pair of undistorted, rectified images from which one can compute disparity map (chapter 2 describes stereo imaging more thoroughly and chapter 6 contains examples of disparity maps and pairs of images used to obtain disparity map). Disparity map includes information about depth and is employed for final triangulation. One of the main advantages of stereo imaging is that relation between left and right camera is calculated just once within calibration, unlike SfM that employs single camera. Figure 4.5 displays rotation of 3D model of teddy bear created using stereo imaging (disparity map was obtained using OpenCV method `StereoBM`). One can notice that all parts of teddy bear are proportional and depth can be well perceived. The model is composed of nearly continous points, and therefore surface is sufficiently precisely defined. However, it could happen that some regions of images, mostly filled with solid colors, are not matched to corresponding regions at the other image, thus disparity values would become inaccurate. White spaces inside model of teddy are result of either occluded or ambiguity regions.

Properties of particulas methods and even experimental evaluation make stereo imaging conclusively suitable method for parking assistant.

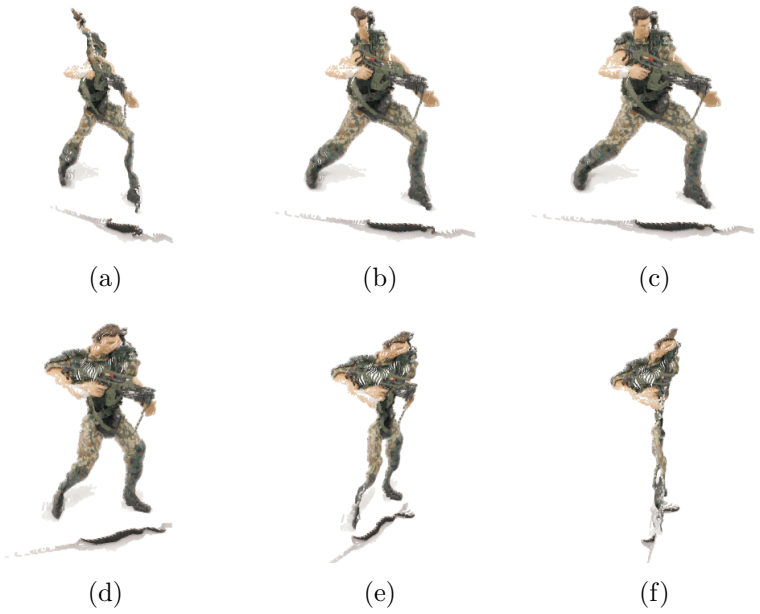


Figure 4.4: Figure depicts rotation of object reconstructed from pair of images (fig. 4.2) using dense optical flow. Images through (a) to (f) represent counterclockwise rotation of object around vertical axis.

4.4 Calibration

The calibration is necessary preceding step before the parking assistant can be properly used. It needs to be performed only once and then only if the relative position of stereo cameras changes. In order to obtain accurate calibration dozens of various chessboard positions have to be employed. To get an equally accurate results each stereo rig should be calibrated independently[24].

A decision whether stereo cameras should be calibrated before fastening to back of a car or after is associated to manufacturing process and consecutive handling of stereo rig. If one can guarantee that mutual position of cameras is same for all stereo rigs and fixed until fastening is finished, calibration can be performed in advance for all cameras. A problem which can arise is that each camera likely possess different properties of lens. These properties were explained in section 1.2 and are essential for stereo imaging. Therefore, the other option for calibration when each car with fastened stereo rig is calibrated independently would meet demands. Each car has to be placed to calibration room where calibration chessboards can be projected on plane surface behind a car. The parking assistant captures such scene and after acquisition of spe-

[24]The calibration of individual stereo rigs decreases influence of lens differences and relative position of cameras

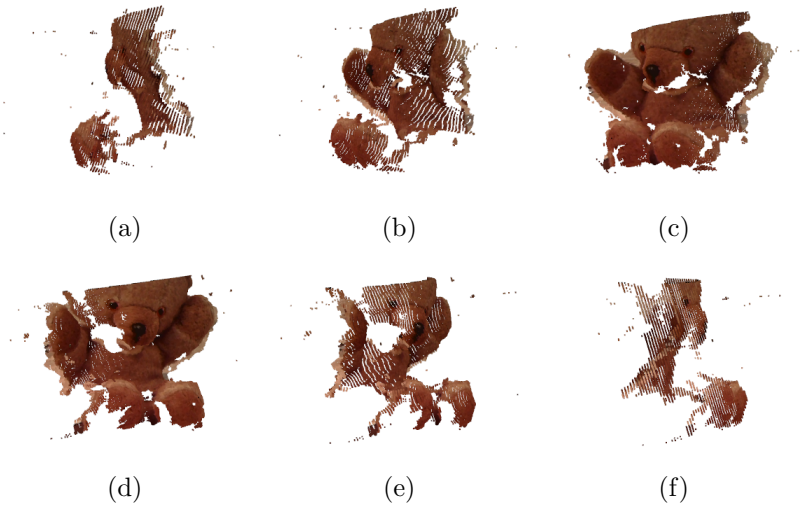


Figure 4.5: Figure depicts rotation of object reconstructed from pair of images using stereo imaging. Images through (a) to (f) represent counterclockwise rotation of object around vertical axis.

cified number[25] of calibration chessboards sets internal settings and becomes ready to use.

4.5 Collision Avoidance System

This section provides detailed specification of the entire collision avoidance system (fig. 4.6). The system is splitted into several steps that must follow consecutively. The first few steps, namely image acquisition, undistortion and rectification, perform independently for each camera. However, they require some kind of synchronization in order to acquire left and right image at nearly the same moment. In this section we assume that cameras have already been calibrated, thus cameras' intrinsics and extrinsics properties are available.

Cameras of stereo rig have predetermined side where they belong to, left and right one. However, computer where they are connected to does not know their relative position. In Linux cameras obtain specific index counted from number zero[26]. In order to express relative position between cameras one should first connect left camera and then right one. It is also worth to mention that each camera should be connected to separate USB bus. The reason is that it avoids an issue when transmission of images from camera could be protracted. Selection of output camera format like YUYV or RGB depends

[25]The minimum number of calibration chessboards necessary to obtain precise calibration is derived in chapter 6.1.

[26]If computer has built-in webcam and boots with no other connected cameras, zero index is assigned to built-in webcam.

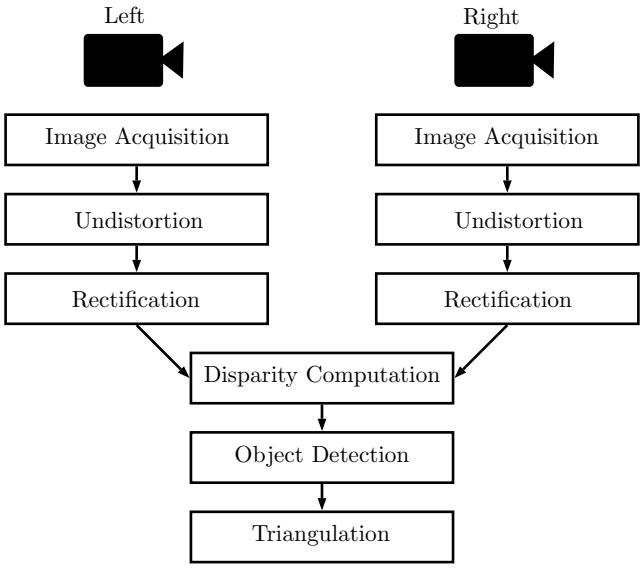


Figure 4.6: Figure depicts main components of collision avoidance system and data flow between them.

on particular cameras and support in system where cameras are connected to. A resolution of video stream was determined as 640×480 px and for some time-consuming tasks could be downsized. FPS is be determined (section 6.7) based on evaluation of processing time of single image. Before processing any image one should ensure that cameras possess the same properties, otherwise following steps do not lead to successful performance.

Since images from stereo cameras are captured we can proceed to next two steps: undistortion and rectification. Both steps are simple remapping of images from cameras and do not change in time. They require to know intrinsics and extrinsics properties of cameras, therefore a change of position between cameras causes incorrect remapping.

The last three steps are very related, and therefore following explanation includes all of them together (fig. 4.7). An ouput of these steps is position of the closest object and its distance to cameras. Firstly, a disparity map is calculated using undistorted and rectified images. A quality of disparity map is influenced by type of utilized method and its properties. Selection of method and its properties is described in section 6.2. To make detection of object easier it is necessary to keep disparity map of scene without any objects. This disparity, reffered as background, has to be calculated with the same method and using the same properties as regular disparity calculation do. Thereafter, background is element-wise subtracted from calculated disparity map of scene and only pixels that exceed determined threshold are maintained in resulting mask. The mask is applied to input disparity map to obtain only objects which appear in scene. From the resulting disparity map a histogram

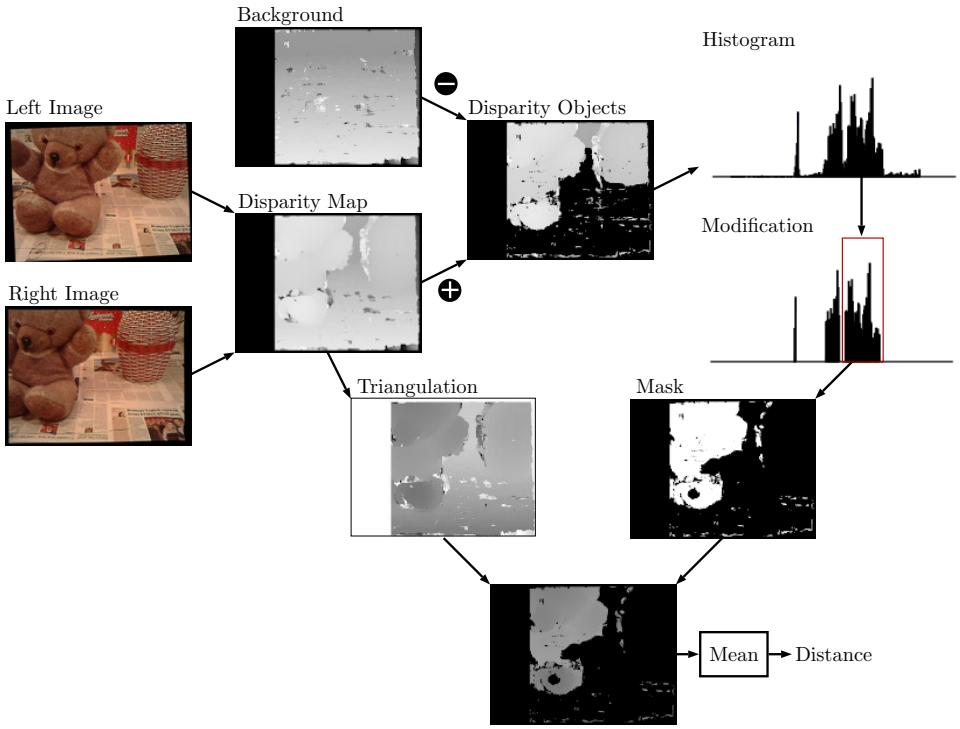


Figure 4.7: Figure shows process of computation position and distance of the closest object toward camera.

is computed. Peaks in histogram represent particular objects that remained in scene, however histogram still contains noisy data. In order to tackle with this issue minimum size of object is required. Each bin of the histogram is thresholded by minimum size of object. Detection of the closest object using such histogram is based on assumption that the closest object in disparity map have higher intensity values and that histogram is sorted from the lowest to highest intensity. To find the closest object values of particular bins are read in descending order until nonzero value is found. Number of such bin indicates upper intensity value of the closest object. Further, lower intensity value need to be find, therefore checking of bins continues. Terminating condition is finding a bin with zero value. Lower intensity value of the closest object is derived from bin which still has nonzero value, but his left neighbor is zero. Lower and upper intensities are employed to mask disparity map with subtracted background and this results in finding of position of the closest object. The mask is also applied to output map of triangulation computed from disparity map. Distance to closest object is determined as mean of all nonzero values of masked triangulation map. In a similar manner the other objects and their distances can be found, however collision avoidance system targets only on the closest one.

4.6 Curb Detection

Proposed collision avoidance system is sufficient to detect any object in a colliding distance. However, if freckles occur in disparity map, detection and classification of particular object can become more difficult. In order to detect curbs more precisely, RGB image is employed. On the other hand, it also possess its own drawbacks, like noise and dissimilarities of scene captured at different lighting conditions.

There is a variety of curbs but they all have common features. These features are represented as lines that form shape of curb. The main line is at the edge of curb. Another line, underneath the main line, separates curb from road. The third line, above the main line, lays on the borderline between curb and pavement. Rarely the third line is missing.

Curbs may vary in shape, from the straight to circle ones. During the observation of curbs, while creating an evaluation dataset, we found out that curbs at parking lots are rather straight. Since curb detection is designed to enhance options of parking assistant, we decied to apply curb detection only at straight curbs. Our proposed curb detection algorithm (fig. 4.8) is composed of sequence of image processing algorithms and heuristics derived from an empirical observation.

In the first step RGB image from left camera is converted to grayscale. In case of curb detection color information has no added value, therefore curbs should not be treated based on it. The RGB image is obtained from left camera, because left-to-right matching was utilized to compute disparity map in the collision avoidance system. This assumption enables to put final lines of curb over a disparity map and acquire an augmented reality. In the next step, histogram equalization [11] reduces issues related to various lighting conditions. Histogram equalization cannot suppress for example an influence of strong shadows, bad lighting conditions, but it works well overall. Amount of noise in captured images can do a harm in further steps of detection, and therefore noise filter is applied. A median filter was chosen among the others, because it preserves edges better [17], unlike Gaussian filter. The next steps are to detect edges and then find straight lines. For the purpose of edge detection canny edge detector [18], state-of-the-art method, is suggested. The lines are supposed to be searched by method called a Hough transform [19]. This method returns plausible lines of sought curb and in the next step they are reduced to maximum number of three. The following paragraph describes an idea of proposed heuristics.

The heuristics firstly tries to merge lines that lay close to each other. The merge of lines is performed by creating a new line that is an average of lines which are perceived as one cluster. After the new line is created all lines from cluster are removed. The remaining lines are iterated through from the bottom to upper side of image. Since one cannot estimate from which angle car approaches to curb the first line is considered as line of curb. The next

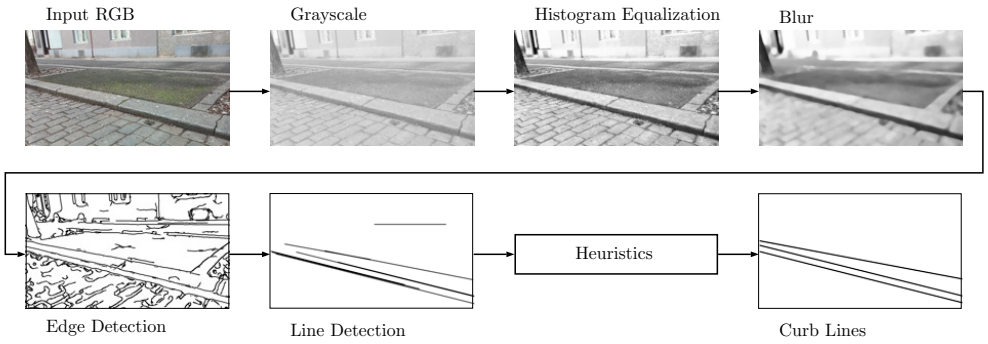


Figure 4.8: Figure depicts steps of curb detection algorithm.

two lines are controlled regarding to maximum allowed angle difference. The size of angle has to be large enough to handle perspective in images. After those three lines are explored algorithm for curb detection in single image is over. The possible outputs of algorithm are none, one, two or three lines.

4.7 Human Detection

Detection of human being belongs to challenging computer vision tasks. Proposed methods usually focuses on detection of human that fully appears in scene, from head to toe. In our case we have to deal with human detection based on legs, because legs are the only body part that stereo cameras could see. Within research of interaction with robots several algorithms for human leg detection were proposed [20][21]. These approaches utilize laser range scanners and assume high precision of measured data, and therefore our computed disparity map would not be sufficient.

We propose an algorithm for detection of human legs that handles disparity map even with freckles. Legs have their own specific shape, especially between a fibula and foot. Using this shape we extract features and employ them in machine learning algorithm. However, one should bare in mind that this approach is not applicable to all views of leg. Current feature extraction method does not exploit values of disparity map. In case human is rotated toward cameras, and thus curved line of leg is not so obvious, values of disparity map could be useful.

Process of feature extraction is depicted in fig. 4.9. Firstly, objects from scene need to be detected. Detection of objects in scene was introduced in section 4.5. For sake of simplicity we assume in this section we have already obtained single objects. In the second step we convert detected object to binary mask. To enhance binary mask white freckles are removed and black holes within object are filled. Detected objects differ in size according to distance from camera, and thus they need to be normalized to fit the same size of image. In the next step Euclidean distance transform is applied. After

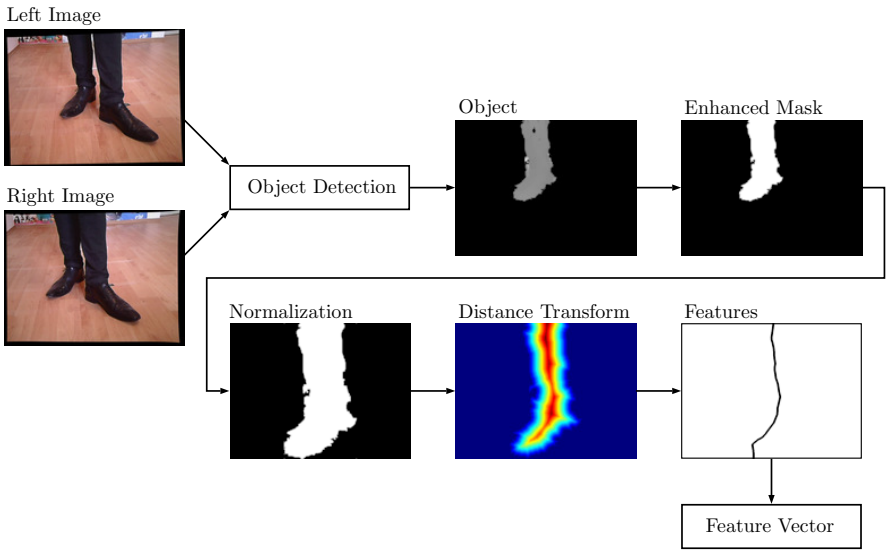


Figure 4.9: Figure shows process of feature extraction from detected object for purpose of human leg detection.

applying the distance transform, each pixel of image has value that equals to the distance between that pixel and its nearest zero pixel. This transformation highlights inner shape of object. During the next step maximum values for each row of image are sequentially found and their positions are considered as features. If there are more values with the same maximum value in a row, the middle position is used. Features from individual rows form feature vector. Length of feature vector equals to the height of normalized image. Due to this relation, feature space can be simply reduced by downsizing input binary image.

After feature extraction is performed, training phase of machine learning algorithm follows. Human leg detection is binary classification problem, therefore positive and negative training samples are needed. Positive samples stand for human legs. Negative samples consist of arbitrary objects. The output of training phase is model which can decide whether given feature vector corresponds to human or not. For the purpose of classification, *random forests* (RF) [22] algorithm was selected.

Broad description of training and evaluation of proposed feature extraction method used to human detection is described in section 6.6.

Chapter 5

Implementation

One of the main requirements of parking assistant is to perform fast. In real application there should be minimum delay between evaluation of captured images. To meet this demand C++ programming language with standard C++11 using STL was selected. Since proposed parking assisstant is largely connected to image processing and computer vision, open source computer vision library called OpenCV was utilized. OpenCV 2.4.10 completely fulfill our needs on parking assistant. Project was compiled under GNU/Linux 3.13 using cmake 2.8.12.2 with gcc 4.8.2.

The main controlling mechanism of parking assistant is finite-state machine (fig. 5.1). After launching, application stays in standby mode from which can proceed to either calibration or running mode. Return from these modes is allowed only back to standby mode. Standby mode simulates passive waiting for interruption. During calibration any of modules, such as collision avoidance system or curb detection, is not performed. Running mode is responsible for an active seeking of objects in a colliding distance and detection of curbs.

The parking assistant is designed as object-oriented project. Objects[27]

[27] In the rest of this chapter we use word object in abstract meaning, not as programming term.

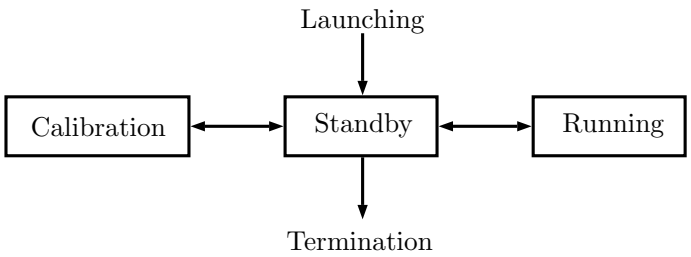


Figure 5.1: Figure defines states of parking assistant and explains transitions between them.

of parking assistant are divided similarly as proposed modules described in chapter 4. Hence, if any new module would be desirable, adding it to existing implementation would not cause major issues. Entry point for all modules is image acquisition object. Images acquired by this object are then distributed to particular modules and processed further. Collision avoidance system is divided to objects (depicted in fig. 4.7) that take care of specific part of algorithm. Undistortion and rectification, disparity computation, object detection and triangulation are fundamental objects utilized in collision avoidance system. Communication between all mentioned objects is carried out by OpenCV data structure called `Mat`. `Mat` is defined as n-dimensional dense array that is employed for storing images.

Each of implemented modules in C++ has their own return structure that would be accepted by reaction mechanism of car. Collision avoidance system returns structure that contains position of object, expressed as `Mat` binary mask, and distance toward to object stored as `float` in units of centimeters. Curb detection system returns structure with boolean variable determining if any curb was found and STL structure `vector` containing individual lines of curb. Line of curb is stored as `vector` of OpenCV data structures `Point`.

In current implementation parking assistant can process video stream captured from cameras. However, one needs to possess stereo rig and perform correct calibration in advance. To present results of particular modules parking assistant can run in two modes: stream and demo. Selection of mode is done before compilation using cmake variable called `DEMO`. Number zero of `DEMO` variable correspondents to stream mode, number one is for demo mode. Stream mode can be controlled using keyboard shortcuts. Key `c` switches from standby mode to calibration mode. To start running mode `r` key has to be pressed. Termination of running mode is performed by `s` key. For purpose of termination of application, `q` key is determined. Demo mode runs sequentially through created datasets and performs particular task on each of them. Results of single input are visualized. By pressing any key evaluation of dataset moves forward.

Human detection system is based on quality of feature extraction and use of an appropriate machine learning algorithm. GNU Octave 3.8.1 and Python 2.7.6 using scikit-learn 0.15.2 were employed for this purpose. Octave produces labeled features that are then imported to Python, splitted, normalized, trained and finally classified.

Chapter 6

Evaluation

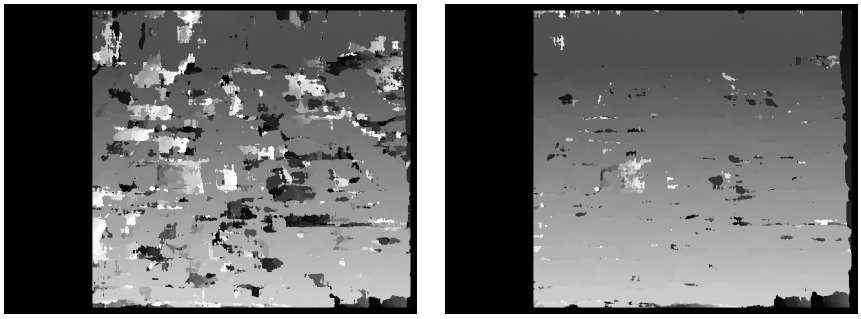
Evaluation of designed and implemented modules of parking assistant is final necessary step, performed in testing environment. Each module was evaluated independently and for each of them testing dataset was created. Collision avoidance system was tested regarding to position of detected object and its distance from camera. Curb and human detection modules were evaluated in terms of success rate detection. Because of importance of low execution time, all modules were time-analysed.

Stereo rig (fig. 6.1a), employed for evaluation purposes, consists of two web cameras Genius FaceCam 2000. Height of stereo rig from the ground was about 40 cm. Focus of web cameras was manually set in order to perceive similar images from both cameras. Distance between cameras was approximately set to 8 cm. The maximum achievable FPS was around 8. Specification of computer, where evaluation was performed, is GNU/Linux 3.13, Intel® Core™ i3, 4 GB RAM.

This chapter describes all details of evaluation process and sheds some light on a usage in real environment.

(a) (b)

Figure 6.1: Figure shows stereo rig composed of two web cameras (a) and arrangement of place (b) where some evaluation datasets were captured.



(a) (b)

Figure 6.2: Figure compares quality of computed disparity maps. Disparity map (a) contains large number of freckles caused by poor calibration. Disparity map (b) on right side was calculated using more precise calibration.

.

6.1 Calibration

This section provides some observations which were encountered during calibration. All attempts to calibrate stereo rig were performed using 9-by-6 chessboard[28].

To verify calibration accuracy one can read the output after calibration of parking assistant. This output, returned from OpenCV method `stereoCalibrate`, is final value of the re-projection error. The less value is returned, the better calibration was performed. However, re-projection error should not be considered as the only evaluation criterion. If the number of stereo images intended for calibration is small, rectification happens to be insufficiently precise even though re-projection error is low. From our experience we prefer to verify calibration by rectification and drawing horizontal lines in left and right image. These lines should connect the same points that appear in both images.

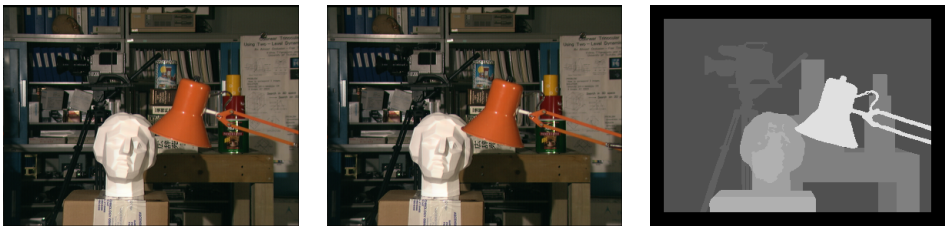
A lot of calibration issues arises from blurry images. Corners cannot be detected with subpixel accuracy and calibration does not perform well. In case of parking assistant we assume that calibration would be done in stable environment, thus this issue would be avoided.

The result of experiments is that 20 stereo pairs are sufficient for calculation of decent calibration. Using web cameras Genius FaceCam 2000 (fig. 6.1a) with standard settings for parking assistant the lowest achieved re-projection error was 0.3.

[28]This number of fields was determined, because it fits well on the size of paper while orientation of chessboard can be understood definitely[1].

6.2 Stereo Correspondence

In order to evaluate accuracy of computed disparity maps we exploited Middlebury stereo datasets [10][23][24][25] and Tsukuba stereo pair with lamp and bust (fig. 6.3). Datasets consist of 36 different scenes. Each scene includes couple of undistorted horizontally rectified images and two corresponding disparity maps. For our evaluation we employed only single disparity map and corresponding stereo pair of images. Among the tested methods for computation of disparity maps belong SAD, SSD, rank and census, all of them described in subsection 2.4.1. We have also included `StereoBM` method from OpenCV. RMSE and percentage of bad matching pixels were applied as the evaluation methods.



(a) (b) (c)

Figure 6.3: Figure displays views from (a) left and (b) right camera (courtesy of the University of Tsukuba). The most right image denotes ground truth disparity map (courtesy of Rick Szeliski and Ramin Zabih).

.

A process of evaluation was performed in several steps. At first, for each couple of stereo pairs disparity map was computed. Since we test block matching methods, which strongly depend on the size of neighborhood, we examined various sizes. The tested sizes were odd numbers between three and nineteen, including these boundaries.

Performance of basic stereo corresponding methods (SAD, SSD and rank) resulted (fig. 6.4a) in strong dependency on the size of neighborhood. Their minimum error differed with size of neighborhood. Census method has steady course and is not dependent on size of neighbohood[29], however the error ranks high. The overall winner of stereo correspondence evaluation is `StereBM` method whose error rate is the lowest. It seems to be very independent on size of neighborhood (fig. 6.4b). On the other hand correct number of disparities is essential. Both displayed graphs (fig. 6.4a and 6.4b) are computed with error tolerance 15.

[29]Census is not dependent on size of neighborhood, because precomputed values of temporary disparity map are compared only using element-wise Hamming distance.

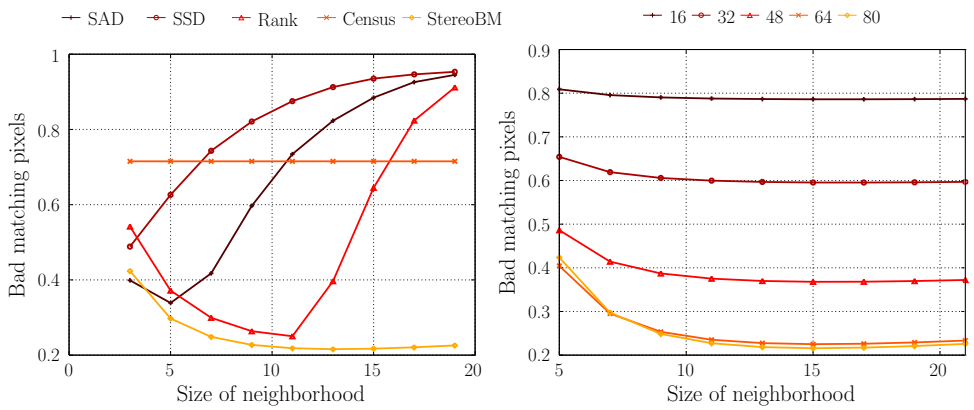


Figure 6.4: Figure (a) presents comparison of stereo corresponding methods according to percentage of bad matching pixels depending on the size of neighborhood. Figure (b) expresses performance of `StereoBM` algorithm with different combination of number of disparities and size of neighborhood.

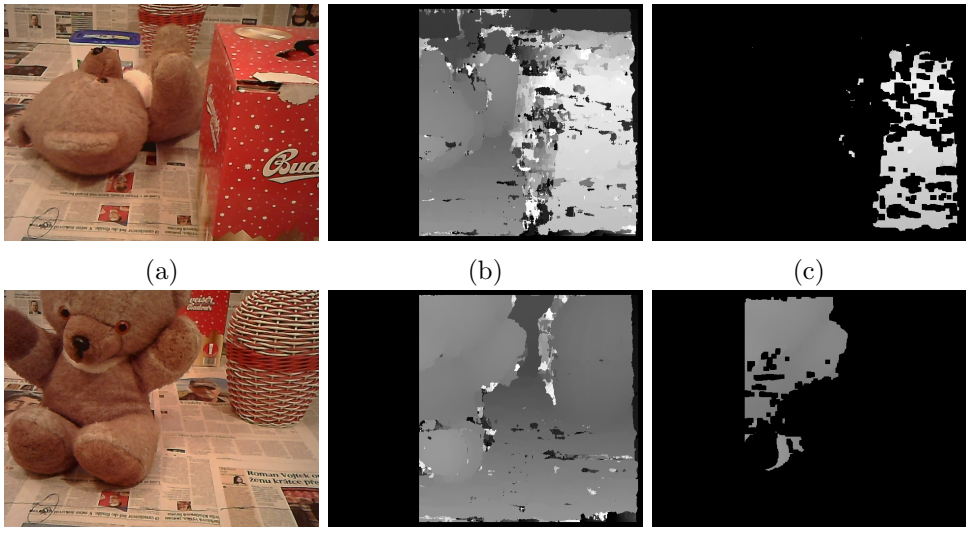
6.3 Object Detection

The following evaluation targets on reliable detection of the closest object toward to camera. In order to confirm quality of proposed detection method evaluation dataset was created (fig. 6.5). This dataset consists of 10 scenes. The scenes are composed either from single object or multiple ones. Single objects are placed at 12 different positions. The scenes with multiple objects consist only of one composition. For each scene there are rectified left and right images (resolution 640 × 480 px) and computed disparity maps with various number of disparities[30]. Images were captured in nearly ideal lighting and texture conditions. In the section 4.5, where collision avoidance system was explained, there is also described the necessity of background possession when the object is detected. Images of empty background without any object (fig. 6.2b) are part of dataset too. Such images could be replaced by artificially created background images. It would require only the knowledge of stereo rig rotation and distance from the ground. Finally, intrinsics, extrinsics and distortion information about cameras that captured this dataset are also attached.

To make evaluation equal for all scenes and simulate real environment[31] object detection was performed with the same number of disparities, specifically 176. This number was chosen as the smallest one which can be utilized for correct disparity map computation. It is worth to mention disadvantage of

[30] There can be found disparity maps with 112, 128, 144, 160 and 176 disparities.

[31] In real environment the same number of disparities, which were set at the beginning, would be utilized all time since selection of various numbers of disparities for different situations was not proposed.



(a) (b) (c)

(d) (e) (f)

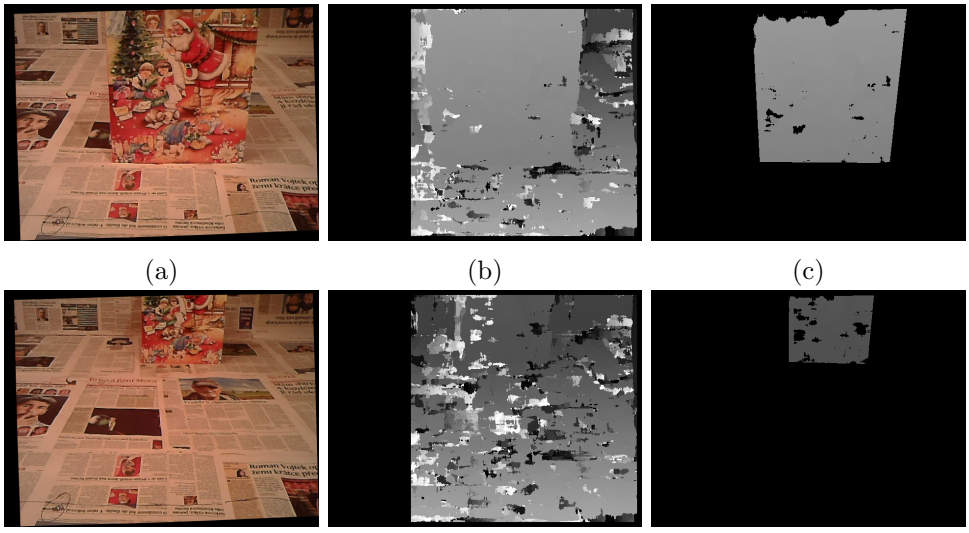
Figure 6.5: Figure depicts part of object detection dataset in the first two columns. The first column represents captured image from left camera. The second column displays computed disparity map using `StereoBM` method with 176 disparities. The last column shows the closest detected object.

large number of disparities. The larger number of disparities[32] is, the smaller part of disparity map is applicable. For example, if disparity map is computed from stereo images with resolution 640×480 px and number of disparities is 144 (in fig. 6.2b visualized as black segment on the left side), disparity map lacks of slightly over 20 % of view. The side of blind spot depends on if disparity map is computed as in left-right or right-left manner. Within this knowledge there arises an option to compute disparity maps in both directions and then stitch these disparities together. However, for evaluation purposes only simple disparity maps were employed. Parameter size of neighborhood was set according to result of experiment displayed in figure 6.4b to 21.

The only necessary input parameter to algorithm, except background for subtraction, is threshold for minimum size of object that we consider as dangerous. The size is defined as number of pixels and all intensities are treated independently. Using empirical approach we found out that threshold value 3100 is sufficient for detection of all closest objects from evaluation dataset. Although this number might seem large, it covers only 1 % of examined disparity map.

The scene number 1 (fig. 6.5d), which strongly affected discussed threshold, is composed of multiple objects. Two of them are in similar but still significantly different distance. Their disparity values fused together and apparent

[32] Large number of disparities is essential for scenes where objects appear closer to camera.



(a) (b) (c)

(d) (e) (f)

Figure 6.6: Figure displays part of dataset (a), (d) utilized for estimation of depth accuracy. The second column depicts calculated disparity maps of stereo pairs. The estimation of distance from camera to object is calculated using masked disparities, shown in the last column. Significant freckles contained in disparity maps are caused by suboptimal calibration.

space between objects disappeared in disparity map. In case that objects would be considered as one depth estimation would happen to be inaccurate.

Another issue which was encountered relates to rather smaller curved objects or parts of objects (noticeable in fig. 6.5f). Disparity values of curved objects are distributed among several intensity values in disparity map. Therefore, they are discarded using threshold together with freckles.

6.4 Depth Estimation Accuracy

The most important result of stereo imaging regarding to parking assistant is an accuracy of depth estimation. In order to evaluate depth estimation a set of ten stereo images with object at different distance from camera was created. Images were taken with resolution 640×480 px. Shape of object is planar and real distance of object from camera was measured as the shortest line between them. Because of this kind of measurement we should be aware of measurement error which can affect evaluation. Measured distances are located between 37 and 144 *cm*. Each stereo pair contains additionaly mask that highlights only the examined object. Object detection module was not employed for finding the closest object. Masks were created manually. As it was already mentioned in section 2.5, disparity and depth are inversely

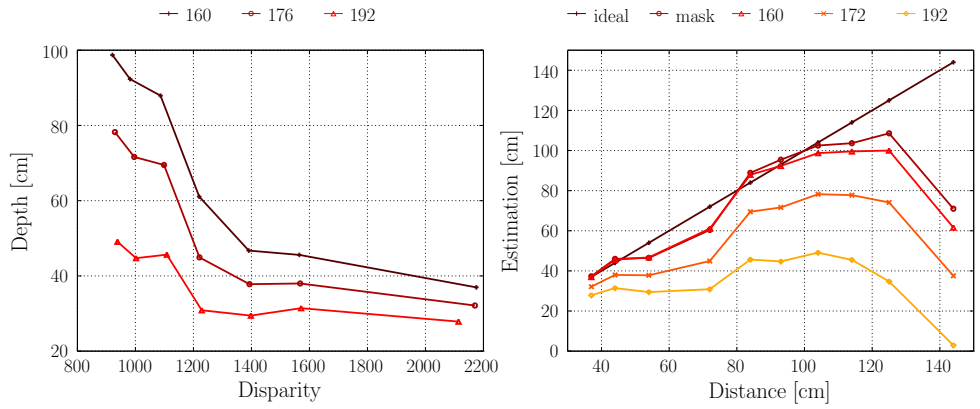


Figure 6.7: Figure (a) confirms non-linear inverse proportion of depth and disparity. For large disparities there is a large range of shallow depths, however low disparities are spread among a small range of greater depths. Graph depicts relations computed by `StereoBM` with various number of disparities. Figure (b) expresses accuracy of depth estimation according to different number of disparities. The best estimation draws near to ideal line.

proportional, moreover, that proportion is non-linear [1] (fig. 6.7a). This property causes that for objects closer to camera one can expect more accurate depth estimation than for farther ones. The estimated value of distance is computed using mean for all triangulated values inside of masked object.

Accuracy of estimated depth was evaluated within three number of disparities: 160, 176 and 192. Selection of disparities were shortlisted in advance. Remained disparities performed best in preceding tests. Parameter size of neighborhood was set according to result of experiment displayed in figure 6.4b to 21. A method utilized for disparity calculation was `StereoBM` from OpenCV.

According to results (fig. 6.7b) it has turned out that selection of right disparity is crucial for obtaining rather accurate depth. Disparity map calculated using 160 disparities performed best. Accuracy of depth estimation decreases with larger number of disparities. The estimation error did not exceed 8 *cm* (RMSE 5.6 *cm*) within tested range up to 104 *cm*. Tested distances which are larger than 104 *cm* possess larger estimation errors. This observation correlates with early explained depth distance non-linearity.

Since not all calculated disparities are perfect, and thus tested object contains freckles, one more specialized test was performed. For each stereo pair lower and upper boundaries of disparity map within examined object were found. These boundaries were used to determine which disparity values are considered. Accuracy (depicted as *mask* in fig. 6.7b) has slightly improved

Figure 6.8: Figure displays random images from dataset of curbs.

mainly for larger distances[33], however course of function is overall same. Even though it is not possible to break non-linearity constraint between depth and distance, accuracy of estimated depth can be enhanced by increasing calibration accuracy.

6.5 Curb Detection

Algorithm proposed in section 4.6 deals with curb detection in RGB image. It is intended as additional function that enhance capabilities of parking assistant. Because we did not find any dataset of curb images, which would meet our demands, we collected our own dataset (fig. 6.8). The dataset consists of 191 images captured with resolution 4160×2340 px. For the purpose of evaluation, images are six times downsized to 640×360 px which is approximately the same size as we consider in another units of parking assistant. Dataset is very challenging. Images were captured at different locations, under various lighting conditions, road was wet and dry, curbs and road appear with cracks and one can find even sign on the road. Height of curbs contained in dataset varies from low to high one.

Evaluation was performed manually for each image from dataset. The number of found lines can be between zero and three and the same range applies to number of poorly detected lines. Number of particular combinations obtained from evaluation is shown in table 6.1. The largest group defines three correctly found lines. The second one consists of two curb lines with no detected error. Finally, the third largest group presents three curb lines with

[33] Accuracy of estimation is higher due to removed freckles. Size of object decreases with larger distance, but the size of freckles is more or less still the same. Therefore, smaller objects are more prone to bad results because of freckles.

one error detection. There are 19 images, nearly 10 % of all images, for which not a single line of curb was detected.

The main purpose of curb detection was to identify potentially dangerous edge, therefore, one more statistics were measured. The most important line is the middle one. This line describes curbs best and by examining its neighborhood other important features can be easily found. The middle line of curb was found in 156 cases out of 191. That makes successful detection rate slightly over 81 %.

Some images that caused issues with curb detection had specific patterns of scene. The most frequent ones are listed in the rest of paragraph. Rectangular manholes with straight hole lines were mostly detected as curbs. Manholes usually appear just before curb and heuristics unfortunatelly considers only the first few lines from the bottom of image. As it was already mentioned, curb detection is not stable to variancy of sunlight. Dataset contains several images where curbs cast a shadow. Heuristics algorithm misclassifies those shadows as lines of curb. Tree needles, gravel or any untidiness at curbs deteriorated detection too. In several cases the highest line of curb was confused with cracks or the end of pavement. Each of these patterns in scene are difficult to deal with using RGB image.

		Found Lines			
		3	**2**	**1**	**0**
Error Detection	**0**	46	44	22	19
	1	37	8	0	
	2	11	1		
	3	3			

Table 6.1: Results from evaluation of curb detection.

6.6 Human Detection

Human detection system introduced in section 4.7 detects human based on leg detection. In order to perform evaluation of proposed feature extraction method, positive and negative datasets must be available. To preserve the same view and overall quality of images, we created positive dataset using our constructed stereo rig (fig. 6.1a). As negative dataset was exploited DUT-OMRON image dataset [26] that contains masks of different objects.

Positive samples of human legs were captured by stereo cameras with resolution 640×480 px. Four different subdatasets were created and contain between 38 to 42 stereo images. Each couple of images in particular subdataset represents legs from another angle or distance. Within preprocessing of posit-

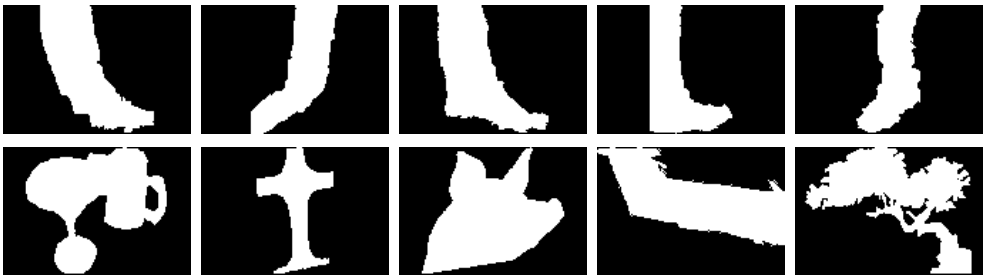


Figure 6.9: Figure shows small part of binary images from human leg detection dataset. The first row displays human legs. The second row contains images of negative samples.

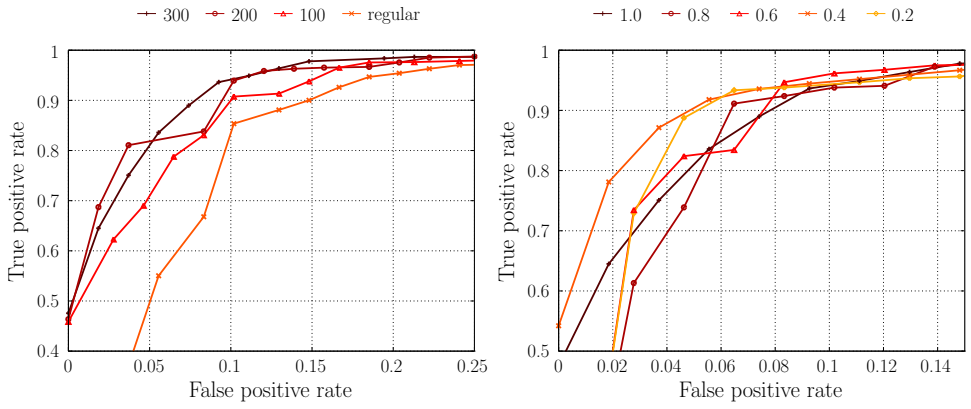
ive dataset disparity map was computed using OpenCV method `StereroBM`[34] for each couple. Individual legs were extracted from disparity map using algorithm described in section 4.5. Because not every extracted leg met the conditions on which proposed method is based, only those with significant curved shape were retained. Total number of extracted legs is 165, counted together left and right ones. When we vertically rotated all of them, total number of legs doubled. Further steps of preprocessing follow instructions explained in section 4.7. Legs are normalized[35] to height 450 px and centered on black background of width 656 px.

DUT-OMRON image dataset contains 5172 binary masks of various objects. Their size differed from determined 656×450 px, and therefore all objects detected in these masks had to be resized. Due to such normalization some of the objects did not fit ratio well and covered large part with white pixels. Because these objects would not bring any potential value to negative dataset, besides artificially decreasing classification error rate, objects which covered more than 70 % of image were removed from dataset. 4220 binary masks were left.

Introduced positive and negative samples of data are imbalanced. Therefore, evaluation of classification should be performed with respect to their class ratio. Another method that also deals with imbalanced data is SMOTE [27]. SMOTE over-samples minority class by creating synthetic examples. In order to run SMOTE, one has to specify amount of over-sampling. If the amount is e.g. 200 %, two nearest neighbors of each sample in minority class are found and from each of them synthetic example is created. New example is obtained by subtraction of particular feature vector from its nearest neighbor and then multiplied with random number between 0 and 1. This result is added to original feature vector. Oversampled data should be used only for training, not for testing.

[34] `SADWindowSize: 21; numberOfDisparities: 176`

[35] Height of normalized image was derived from leg with the highest vertical length. Width of image was derived from the widest leg after normalization in height.



(a) (b)

Figure 6.10: Figure (a) compares classification rates on datasets enriched by SMOTE and original dataset. The second figure (b) presents relation between classification rate and size of input images from which features were calculated.

For the purpose of training and evaluation of classification, datasets were physically divided into two parts: training and validation. Parts are of ratio 80 to 20. Validation part is considered as data that are not seen by exploited machine learning algorithm, and thus it simulates real unseen data. Evaluation of validation part is performed after tuning parameters of algorithm. Obtained precision and recall from validation dataset is decisive compared to evaluation retrieved during training.

Random forests algorithm does not require[36,37] cross-validation, because each tree is built[38] from different bootstrap sample of the training data. Algorithm was trained on 80 % of training data and the rest was used for classification. Evaluation of classification and searching for the best parameters of RF algorithm were performed in three steps.

In the first step, sparse grid search using RF was run on four different datasets: regular dataset and three datasets with enriched training part using SMOTE 100 %, 200 % and 300 %. Grid search provided[39] number of trees (denoted as `n_estimators` in sicikit-learn) in forest from 5 to 160, where each next number doubled, and method for measuring quality of split. Gini impurity and information gain were evaluated as measure for quality of split. The aim of this step was to find out if over-sampling of data actually improves classification performance. Figure 6.10a shows advantage of over-sampled training minority data in comparison to regular dataset. The best result was achieved

[36] www.stat.berkeley.edu/~breiman/RandomForests/

[37] Random forests from scikit-learn has to be run using parameters: `oob_score: true; bootstrap: true`

[38] For purpose of reproduction the same results random forests with identical parameters were built alike.

[39] The rest of random forest parameters used default settings of scikit-learn.

using 300 % over-sampling, 160 trees and split according to information gain. These parameters were therefore employed in the second step of evaluation.

The second step finds out whether size of input image, and thus length of feature vector, is correlated to classification performance. Images of dataset were downsized to 80 %, 60 %, 40 % and 20 %, respectively length of feature vectors was reduced to 360, 270, 180 and 90. Results of this experiment are displayed in figure 6.10b. One can see that reduction of feature space does not negatively affect classification performance. Surprisingly reduction to 20 % and 40 % performs better than on original images, but it cannot be considered as preferable size of input features. More important conclusion is that reducing feature space up to 80 % does not worsen performance.

The last step explores parameters of RF more deeply and evaluates validation data using final determined combination of parameters. Even though RF provides sort of overfitting protection, there is no reason to allow construction of trees with small number of samples required to split an internal node (denoted as `min_samples_split` in scikit-learn) or creation of a new node with small number of samples (denoted as `min_samples_leaf` in scikit-learn). Within this step we tried to find upper limit of these parameters while preserving good classification performance. Similarly we tried to minimize number of trees. Mean accuracy calculated from all tested combinations is 93.7 % which stand for almost all configurations with 4 and less `min_samples_leaf`. Difference between the highest and the lowest mean accuracy using 4 `min_samples_leaf` is only 1.7 %, and therefore we decided to select 16 `min_samples_split` as the largest tested parameter. Smallest number of `n_estimators`, which achieved the same results as configurations with larger numbers, was 40. After training RF on whole training dataset, we achieved 90 % precision and 89 % recall on validation set.

6.7 Time Performance

Common goal of all systems that need to perform in real time is to achieve low computation time. In this section we evaluate time performance of proposed methods employed in parking assistant. At each method we measure time spent in individual modules in order to localize bottleneck that slows down overall time performance. Because computation time is obtained as time difference between two positions in source code, we are not able to avoid including time that processor spent on running different processes. Hence, time duration which we present (table 6.2) here is the lowest[40] measured using appropriate dataset.

[40]The lowest measured time duration of particular module is approximately the closest estimate when only parking assistant would be run.

Module	Time performance [ms]
Collision avoidance system	
Undistortion and rectification†	6.7
Disparity computation	192.4
Object detection	5.4
Triangulation	13.9
Curb detection	
Convert to grayscale	1.2
Histogram equalization	0.7
Blur	99.5
Edge detection	4.9
Line detection‡	33.5
Heuristics‡	0.1
Human detection	
Normalization	4.3
Distance transform	9.4
Feature extraction	7.8
Classification⋆	0.1

Table 6.2: Time performance of parking assistant modules.

Processing of stereo pair of images in collision avoidance system takes about 218 ms. Collision avoidance system should be able to go through 4.5 stereo pairs per second. However, almost 90 % of that time is spent in disparity computation module. In order to accelerate this module, one could downsize[45] input stereo pair. When the size of stereo pair of images was scaled down to 80 %, 60 % and 40 %, time performance of algorithm became 111.1 ms, 53 ms and 22 ms respectively. Downscaling of images speeds up even the other modules.

Overall time performance of curb detection is about 139 ms, therefore it allows to process seven frames per second. Computation of blur lasts the longest. The reason for that is because we utilized median filter, which is known for long time performance. Using gausian filter with similar settings we obtained average processing time 9.3 ms. Time advantage of gaussian filter is obvious, however it does not provide quality blurred images which preseve edges.

†Duration consists of undistortion and rectification of both left and right image.

‡Time performance was calculated as median of measured values, because running time of module strongly depends on particular input.

⋆Time performance was obtained from classification of 904 samples using scikit-learn.

[45]Current implementation of parking assistant does not guarantee correct performance if input images do not have specified diemnsions.

Human detection lasts for one object approximately 21.6 ms. It means that for example ten objects could be classified five time per second.

If one would like to run all modules together on computer similar to one employed for tests here, time performance would be poor. Solution could be either use faster computer, speed up particular modules or utilize smaller input images.

Chapter 7

Discussion

All proposed algorithms besides curb detection are dependent on quality of calibration. In thesis we revealed that without precise calibration algorithms cannot work correctly. Calibration is therefore recommended to perform in stable environment, otherwise one cannot rely on accuracy of parking assistant modules.

Throughout practical part of thesis we employed two ordinary web cameras. These cameras capture scene with narrow angle of view that does not cover all space behind a car. To make proposed parking assistant applicable even for real usage, more cameras or cameras with wide-angle lenses need to be employed. In case of more cameras process of 3D scene reconstruction does not change too much. Particular cameras that lay next to each other would be treated the same way as described in thesis and finally their disparity maps would be stitched together. Pair of cameras with wide-angle lenses would easily cover all space behind a car. However, because of lens shape and its different projection density of scene to image, we cannot guarantee that two cameras are sufficient. Therefore, we suggest further research of utilized cameras with respect to parking assistant.

Cars with radar or laser sensor allert a driver when they approach an obstacle. Proposed collision avoidance system is also able to detect any obstacle using simple technique. Depth estimation is also affiliated module of collision avoidance system. The main drawback of this module is decreasing estimation accuracy by distance. There is not much we can do about it, because it is caused by relation between disparity and depth. However, we can set maximum trustworthy distance which can be acknowledged, or employ size of detected object for depth estimation purposes.

Curb detection that was proposed exploits only RGB image. This approach was selected because of frequent occurrence of freckles in disparity map. Freckles complicate understanding of scene semantics. Besides ambiguity regions in images, freckles occur due to inaccurate calibration. Successful detection rate of curb detection implementation did not reach high. It is

caused by various lighting conditions and wrong interpretation of scene. We believe that employing quality disparity map would prevent these issues.

Human detection employs only side and almost frontal view of human leg silhouettes. Classification algorithm performs rather well on examined dataset. However, if captured leg does not retain curved shape, classification most probably fails. To broaden leg positions that can be correctly classified, spatial information could be derived from disparity map.

Created datasets helped to support and evaluate proposed methods, however the amount of images should be larger if one needs to test real parking assistant. Datasets should be composed of thousands of images captured in different places and their corresponding ground truth.

Conclusion

This thesis describes methods for reconstruction of 3D scene using images captured by web cameras. It also explains camera models and calibration of cameras. Stereo imaging and structure from motion are thoroughly compared and stereo imaging is eventually selected as more reliable method for 3D reconstruction.

Based on acquired knowledge we propose and implement parking assistant using two stereo web cameras. Parking assistant consists of three modules: collision avoidance system, curb detection and human detection.

Main purpose of collision avoidance system is to detect the closest object in scene and estimate its distance. In order to evaluate object detection and depth estimation accuracy, we create datasets for both of these demands. System is able to detect even small objects that cover at least 1 % of image with its area. Depth estimation RMSE is 5.6 cm for distances up to 104 cm range. Time performance of collision avoidance system without optimizations reaches 4.5 frames per second.

Curb detection employs only RGB input image from left camera of stereo rig. Searching for curb is based on detection of straight lines and utilization of subsequent heuristics. Despite challenging dataset, which is created for evaluation purposes within this thesis, obtained successful detection rate is 81 %. One can achieve seven frames per second when using curb detection.

Module of human detection extracts features from detected and normalized objects in disparity map. Random forests classifier is trained on positive samples of legs and negative samples of various objects. Created dataset of positive samples contains four different people in dozens of positions and distances from camera. Classification precision on validation dataset is 90 % and recall 89 %. Time performance of human detection module is 21.6 ms per one object.

Results of experiments confirmed that cameras could be suitable replacement for radar and laser sensors in cars. As benefit, it could also provide vast amount of features, such as curb or human detection.

Bibliography

[1] Bradski, D. G. R.; Kaehler, A. *Learning Opencv, 1st Edition.* O'Reilly Media, Inc., first edition, 2008, ISBN 9780596516130.

[2] Heikkila, J.; Silvén, O. A four-step camera calibration procedure with implicit image correction. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, IEEE, 1997, pp. 1106–1112.

[3] Solem, J. E. *Programming computer vision with Python.* Beijing; Cambridge; Sebastopol [etc.]: O'Reilly, 2012, ISBN 9781449316549.

[4] Zhang, Z. Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. In *ICCV*, 1999, pp. 666–673.

[5] Zhang, Z. A Flexible New Technique for Camera Calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 22, no. 11, Nov. 2000: pp. 1330–1334, ISSN 0162-8828.

[6] Brown, D. C. Close-range camera calibration. *PHOTOGRAMMETRIC ENGINEERING*, volume 37, no. 8, 1971: pp. 855–866.

[7] Hartley, R. I.; Zisserman, A. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[8] Brown, M.; Burschka, D.; Hager, G. Advances in computational stereo. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, volume 25, no. 8, Aug 2003: pp. 993–1008, ISSN 0162-8828.

[9] Zabih, R.; Ll, J. W. Non-parametric Local Transforms for Computing Visual Correspondence. Springer-Verlag, 1994, pp. 151–158.

[10] Scharstein, D.; Szeliski, R. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *Int. J. Comput. Vision*, volume 47, no. 1-3, Apr. 2002: pp. 7–42, ISSN 0920-5691.

[11] Klette, R. *Concise Computer Vision: An Introduction into Theory and Algorithms*. Springer Publishing Company, Incorporated, 2014, ISBN 1447163192.

[12] Baggio, D. L.; Emami, S.; Escriva, D. M.; et al. *Mastering OpenCV with Practical Computer Vision Projects*. Packt Publishing, Limited, 2012, ISBN 9781849517829.

[13] Grauman, K.; Leibe, B. *Visual Recognition, Local Features: Detection and Description*. The University of Texas at Austin, 2009.

[14] Lowe, D. G. Object Recognition from Local Scale-Invariant Features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, Washington, DC, USA: IEEE Computer Society, 1999, ISBN 0-7695-0164-8, pp. 1150–1157.

[15] Lucas, B. D.; Kanade, T. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.

[16] Furukawa, Y.; Ponce, J. Carved Visual Hulls for Image-Based Modeling. In *Computer Vision – ECCV 2006*, *Lecture Notes in Computer Science*, volume 3951, edited by A. Leonardis; H. Bischof; A. Pinz, Springer Berlin Heidelberg, 2006, ISBN 978-3-540-33832-1, pp. 564–577.

[17] Parker, J. R. *Algorithms for Image Processing and Computer Vision*. Wiley Publishing, second edition, 2010, ISBN 0470643854.

[18] Canny, J. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 8, no. 6, June 1986: pp. 679–698, ISSN 0162-8828.

[19] Duda, R. O.; Hart, P. E. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Commun. ACM*, volume 15, no. 1, Jan. 1972: pp. 11–15, ISSN 0001-0782.

[20] Chung, W.; Kim, H.; Yoo, Y.; et al. The Detection and Following of Human Legs Through Inductive Approaches for a Mobile Robot With a Single Laser Range Finder. *Industrial Electronics, IEEE Transactions on*, volume 59, no. 8, Aug 2012: pp. 3156–3166, ISSN 0278-0046.

[21] Bellotto, N.; Hu, H. Multisensor-Based Human Detection and Tracking for Mobile Service Robots. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, volume 39, no. 1, Feb 2009: pp. 167–181, ISSN 1083-4419.

[22] Breiman, L. Random Forests. *Machine Learning*, volume 45, no. 1, 2001: pp. 5–32, ISSN 0885-6125.

[23] Scharstein, D.; Szeliski, R. High-accuracy Stereo Depth Maps Using Structured Light. In *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR'03, Washington, DC, USA: IEEE Computer Society, 2003, ISBN 0-7695-1900-8, pp. 195–202.

[24] Scharstein, D. Learning conditional random fields for stereo. In *CVPR*, 2007.

[25] Hirschmüller, H. Evaluation of Cost Functions for Stereo Matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[26] Yang, C.; Zhang, L.; Lu, H.; et al. Saliency detection via graph-based manifold ranking. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, IEEE, 2013, pp. 3166–3173.

[27] Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; et al. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Int. Res.*, volume 16, no. 1, June 2002: pp. 321–357, ISSN 1076-9757.

Appendix A

List of Acronyms

FPS Frames per second.

RF Random Forests.

RGB Red Green Blue.

RMSE Root-Mean-Squared Error.

SAD Sum of Absolute Differences.

SfM Structure from Motion.

SIFT Scale Invariant Feature Transform.

SMOTE Synthetic Minority Over-sampling.

SSD Sum of Squared Differences.

STL Standard Template Library.

SVD Singular Value Decomposition.

YUYV Y - luminance, U - blue–difference, V - red–difference.

Appendix B

DVD content

- `Datasets` ..... datasets employed within evaluation
  - `Curb_Detection`
  - `Depth_Estimation_Accuracy`
  - `Human_Detection`
  - `Object_Detection`
- `Source` ..... source codes of implementation
- `Thesis` ..... source codes of thesis in LaTeX
- `Readme.txt` ..... instructions for compilation and running
- `Setup.sh` ..... script for setup environment, compilation and running
- `Thesis.pdf` ..... thesis in PDF format