Monocular Depth Perception

ADVANCED SEMINAR

submitted by
Tobias Watzel

NEUROSCIENTIFIC SYSTEM THEORY
Technische Universität München

Supervisor: Lukas Everding
Final Submission: 07.07.2015

Abstract

In the advanced seminar work two algorithms for monocular depth perception are presented. The first one is optimized for images of static outdoor scenes and employs a multi-resolution framework with a combination of dark pixels and saturation to increase depth accuracy. The second algorithm is used for dynamic scenes like videos with local and global camera motion. The combination of a background depth generation (BDG) and a motion analysis with outlier rejection leads to a creation of decent depth map. The work ends with a short evaluation of the proposed algorithms.

TECHNISCHE UNIVERSITÄT MÜNCHEN
NEUROWISSENSCHAFTLICHE SYSTEMTHEORIE
PROF. DR. JÖRG CONRADT

2015-04-01

A D V A N C E D S E M I N A R

Monocular Depth Perception

Problem description:

For humans it seems like an easy task to orient themselves in the environment and create an internal representation of their surroundings. But in order to solve this task a lot of information processing has to be done. One important part is to find out how far objects are away. Traditionally, it is assumed that humans learn the distances by comparing the two views which our eyes provide and estimate the distances through disparities of object images on the retinas. This is certainly true and this finding has also often been used in computer vision to extract depth information from the environment using two cameras. But binocular stereo vision is not the whole story. Due to the structure of the world and a priori knowledge humans (and animals) are also able to extract depth information by using only cues of a single eye in dynamical scenes. To this day, performance levels of depth information extractions in nature are far superior to those of computer vision and it is desirable to find ways to increase the performance of stereo vision algorithms. In the past it has often been rewarding to study how nature solves problems and transfer the insights to technical applications. So, for this project we want you to focus on the area of monocular cues for depth perception and how to make them usable for technical systems. Your tasks include:

- Get familiar with the depth extraction problem and binocular approaches to solve it
- Find out which monocular cues are used primarily in nature to extract depth information
- Research under which circumstances and in which ways those cues get used to assist or complement classical binocular stereo vision
- Investigate which prerequisites need to be fulfilled in order to use monocular depth extraction in artificial vision systems (for frame-based cameras and dynamic vision cameras)
- Explore if there are technical system that already use techniques you found out about and present an example or give reasons why they are not used

Supervisor: Lukas Everding

(Jörg Conradt)
Professor

Contents

Chapter 1

Introduction

The following work for the advanced seminar is divided into three major parts: *Introduction*, *Main Part* and *Summary*. The *Introduction* deals with different cues which are used for depth perception in nature and robotics. Afterwards in the *Main Part* two algorithms are presented which are able to create depth maps out of 2D images. This part ends with an evaluation to find out if these algorithms are suitable for real applications. The *Summary* completes the advanced seminar work and gives a short summary.

1.1 Problem Statement

The advanced seminar focuses on the area of monocular cues for depth perception and how to make these usable for technical systems. The aim is to investigate two algorithms which are used for monocular depth perception. One is used for static scenes, the other one for dynamic scenes. Furthermore, these algorithms are examined with respect to accuracy, robustness and calculation time to find out whether the algorithm is suitable for real applications.

1.2 Monocular and binocular depth cues

Roughly one can divide visual information into two parts: monocular and binocular cues. Figure 1.1 presents an overview of different cues for depth perception.

Monocular cues can be divided into static and dynamic cues. Static monocular cues contain perspective (linear, textures), interposition (occlusion, transparency), lighting (shading,shadow), aerial (optical haze, mist) and focussing (image blur, accommodation) cues. On the other hand, dynamic cues are optic flow, motion parallax and accretion (deletion). Furthermore, binocular cues are divided into vergence (static, changing) and disparity (occlusion disparity, position disparity) cues.

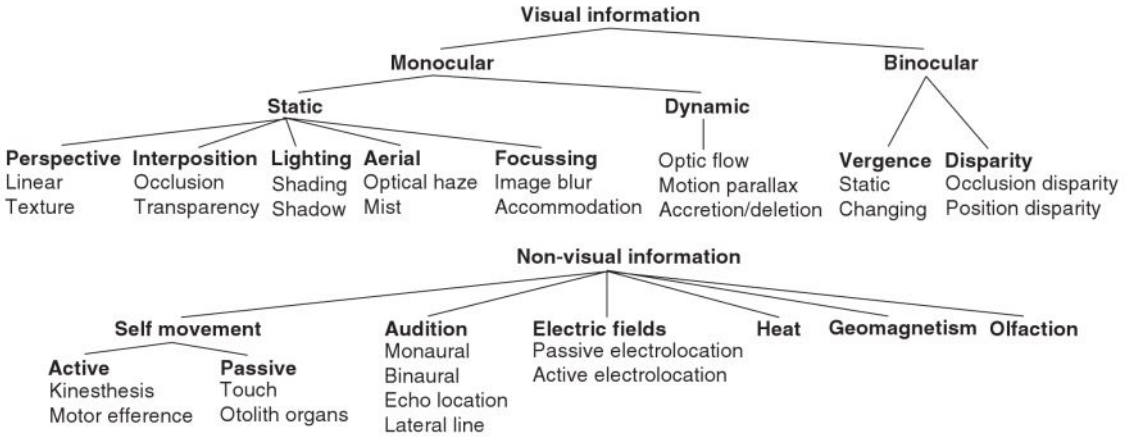

Figure 1.1: Overview of different cues for depth perception [How12, p.2]

In the advanced seminar the chosen algorithms use monocular cues for creating a depth map. The algorithm for static scenes employs aerial cues and the algorithm for dynamic scenes applies cues of optical flow.

1.3 Depth perception in nature and robotics

In 3D space humans and animals are able to orient and move easily. This seems to be a simple task for most of the creatures. However, blind people demonstrate what can happen if one of the most important sensors for orientation is not available. They aren't able to orient in their environment and need help to prevent collisions with obstacles. This demonstrate us how important visual information is for motion and orientation.
Humans and animals use mobile eyes as sensors. Insects have immobile eyes like the mantid *Empusa fasciata*. The insects e.g. apply a consist forward and backward movement of the head [Kra03] to compensate the immobility of their eyes.
During a motion through space creatures have to estimate depth to prevent a collision with obstacles. That is a difficult task because the image of a 3D object is projected onto the retina which is a 2D surface. That is the reason depth information is lost. Hence, creatures need to utilize different kind of cues which contain information of the distance between an observer and an object. Creatures are able to use these cues easily. Moreover, they use their prior knowledge of how obstacles act in distance (e.g. size, occultation,...).
In robotics estimating depth is necessary to calculate a trajectory for a robot in motion. It is desired to find a trajectory in an environment without colliding with obstacles. Certainly one could use a distance sensor to percept depth but in case of a dysfunction the robot isn't able to orient in a unknown environment. Moreover, active sensing requires a lot of energy to run different kind of sensors. However, using

computer vision for depth estimation is often computationally expensive. Nowadays researchers investigate which cues are useful and how to use these cues more effectively.

1.3.1 Depth perception in nature

In nature monocular cues as well as binocular cues are used. Most of the predators take advantage of binocular depth cues because they need accurate depth information to successfully hunt their prey. Therefore, the eyes are arranged in front of their heads and the images of both eyes overlap in the field of view. Through a neural comparison of the difference between these images a depth estimation is applied[Cro05, p.110]. However, prey animals often employ monocular cues because they aren't able to use binocular cues [CC06]. Their field of view has only an small overlap of their retina images but in return they gain a near 360° view. This feature helps them to spot predators as soon as possible and to flee. Therefore they use perspective, interposition, light, aerial, focussing and dynamic cues. Though, their depth perception isn't relying on one specific cue rather than on a combination of different monocular cues. In [CC06] pigeons were tested on different static cue conditions and they discovered that their pigeons performed best in discrimination for the three-cues condition. Another experiment examined how tiny insects like flies are able to land on an object by only applying the dynamic cue optic flow[vBMD14]. In summary, nature employs a wide range of different kind of cues. Even tiny insects with a small central nervous system (CNS) are able to use these cues easily which help them to survive.

1.3.2 Depth perception in robotics

In robotics the most common way to estimate depth is by employing binocular cues. Usually, two cameras are calibrated and the extrinsic and intrinsic camera parameters are determined. Then the cameras take an image of a scene and features are extracted out of each image. The features are used in a matching phase to find similar parts of the scene in each image. There are different kind of approaches like in [CxYh09] which employs Speed Up Robust Feature (SURF) for binocular depth measurement or [SL11] which uses a new matching algorithm with colour segmentation, local matching in pixel domain, the disparity plane estimation and disparity plane assignment to estimate depth.
Nowadays applying monocular cues for depth perception gets popular. Most of the affordable consumer digital cameras are only capable of monocular 2D images. In addition, 3D displays are getting more popular these days and thus create a 3D scene out of a 2D image becomes an important issue[KL11]. Furthermore, monocular cues are able to support depth estimation of binocular systems [SSN07a] because they are useful to improve the accuracy of stereo depth map. Accurate depth maps are e.g. necessary for robots to drive through an unknown area because they have to find

trajectories to prevent collisions with obstacles. In case of noisy or inaccurate depth maps the robots will more likely collide with objects. Especially in autonomous driving this could result in fatal damage. To summarize, in robotics there are several approaches for extracting features out of images to estimate depth. However, these approaches are often computationally expensive and are outperformed easily by nature.
In the next chapter we will now have a closer look at two algorithms which are able to create depth maps out of monocular images.

Chapter 2

Main Part

In the *Main Part* two algorithms for depth extraction are presented. The first algorithm is used for static scene and the second for dynamic scenes. The Main Part ends with a short evaluation with focus on feasibility.

2.1 Implementation of monocular cues for depth extraction

The *Introduction* illustrated that there are a lot of different monocular cues which are used for depth estimation in a 2D image. Two algorithms were selected which employ these cues. The first algorithm was created by Tien-Ying Kuo et al. in 2011[KL11]. His approach focuses on static outdoor scenes. Their approach is very efficient and robust by applying a multi-resolution framework. The second algorithm handles dynamic scenes. It is able to construct depth maps out of scenes with global motions like camera motions and also local motions in current frames. Hu Tian et al. proposed these approach in [TZH+13] 2013. They also created a framework which is state-based depending on camera motion.

2.1.1 Depth Estimation from a Monocular View of the Outdoors by Tien-Ying Kuo et al. for static scenes [KL11]

The algorithm of Tien-Ying Kuo et al. contains three major components: *initial depth prior*, *dark channel prior* and *multi-resolution analysis*.

In this approach a multi-resolution framework is applied which is shown in figure 2.1. In the beginning the input-image is down-sampled three times with a factor of 3, 9 and 27. This leads to three images $I_{1/3}$, $I_{1/9}$ and $I_{1/27}$. The down-sampled images help to eliminate noisy depth variants.

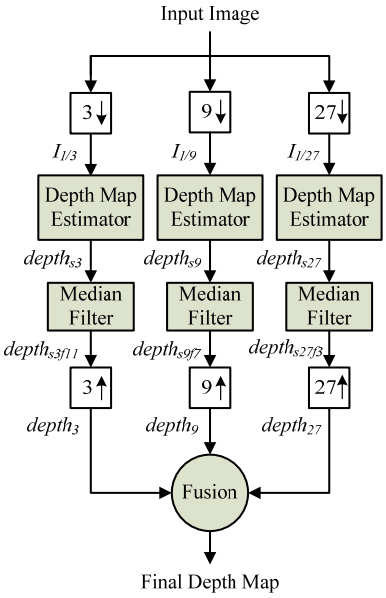

Figure 2.1: Framework for depth map generation [KL11]

Depth map estimator

In the next step the depth map estimation is generated which is the core of the algorithm. First the down-sampled images are filtered with a bilateral filter to remove texture details. Then a rudimentary depth map is created by using object segmentation. For that propose a canny edge detector with a connected component labelling is applied. After that the image is binarized by using the approach in [Ots75] to find the optimal global threshold TH_e. Next, one has to divide the image into section to assign a depth value to each section. For this the result of canny edge detection is transformed into HSV (hue, saturation, value) colour space which is invariant in intensity. The code for a RGBtoHSV transformation can be found in [Ago05, p.303]. In HSV space pixels getting connected to regions based on their colour similarity. If one of the features (hue, saturation or value) is bigger than a pre-set threshold value TH_S, TH_S or TH_V an object boundary is terminated. Otherwise, the pixels are connected to regions if they are close in colour similarity and below their thresholds. This way a rudimentary depth map is created which allows a guess of depths on each object.

After that, one has to map the vertical axes of the images to the normalized coordinates ranging from 0 to 1 uniformly. Since most outdoor scenes behave in the way that objects close to the bottom of the scene are located nearer to the observer than objects at the top, the value $depth_{y,l}$ should be close to 0 at the bottom and close to 1 at the top of the image. Thus, it is possible to assign a depth value $depth_{y,l}$ linearly from the bottom (shortest distance) through the top (farthest distance) to each region. l indicates the layer of the multi-resolution. As a last step the depths

$depth_{y,l}$ are mapped to $depth_{init,l}$ by applying a human perception function to get closer to a stereo vision perception of ground truth.
So far initial depth maps $depth_{init,l}$ are created. However, these are quite inaccurate and for this reason dark channel prior and saturation are used to improve their depth accuracy at different resolutions.
The dark channel prior exploits the fact that atmospheric haze is observed in outdoor images. The haze results in a reflection of more atmospheric light for a distant object which looks more faded and tends to be white. Therefore, so called *dark pixels* are extracted. These *dark pixels* are very low in intensity in at least one RGB channel. Their values increase for objects located far away because its intensity is blended with air light. In opposite *dark pixels* which are darker tend to be closer to the viewer[HST11]. All these *dark pixels* are collected in $J_{dark,l}$:

$$J_{dark,l}(x,y) = \min_{c\in r,g,b}(\min_{(i,j)\in\Omega(x,y)} I_l^c(i,j)) \tag{2.1}$$

Thus, to get $J_{dark,l}$ one has to take a patch $\Omega(x,y)$ centred at position x and y of the image I_l^c. The patch consists of three colour layers(red, green, blue). First one has to find the minimum in each layer. Then the smallest value of the three values of the layers is taken to determine $J_{dark,l}$ at the position of x and y. $J_{dark,l}$ is the dark channel of I_l for layer l and represents the depth. In the last step $J_{dark,l}$ is normalized (0 to 1).
In order to increase the robustness of the dark channel prior, colour saturation is used. Color saturation is defined as the difference between the maximal and the minimal RGB channel like in equation 2.2.

$$S_l(x,y) = \max_{c\in r,g,b} I_l^c(x,y) - \min_{c\in r,g,b} I_l^c(x,y) \tag{2.2}$$

It helps to decide if a *dark pixel* has been suppressed. This is necessary for regions with low saturation because almost all RGB channels have a quite high intensity. Hereby, values occur which aren't representative to depth. Due to this reason, equation 2.3 is applied which combines the initial depth map, saturation and the dark channel to get the depth map.

$$\begin{aligned} depth_{s,l} = {} & w_1 depth_{init,l} J_{dark,l} \\ & + w_2 J_{dark,l} S_l^{1-sigmoid(y)} \\ & + w_3 depth_{init,l} (1-S_l)^{1-sigmoid(y)} \end{aligned} \tag{2.3}$$

The first part of $depth_{s,l}$ combines the initial depth map $depth_{init,l}$ with the dark channel map $J_{dark,l}$. In the second part of 2.3, $S_l^{1-sigmoid(y)}$ suppresses areas where dark channel prior can fail. These suppressed areas are compensated by the last part $(1-S_l)^{1-sigmoid(y)}$. y describes the vertical coordinates of the image. The weighting parameters w_1 to w_3 result in of a learning system which employs AdaBoost[FS97].

Median filter

After the depth map estimator is applied a median filter is used to remove noisy depth variations in local areas of the images. The filters are of the size 11x11, 7x7 and 3x3 for $depth_{s3}$, $depth_{s9}$ and $depth_{s27}$. This results in depth maps $depth_{s3f11}$, $depth_{s9f7}$ and $depth_{s27f3}$ which then are up-sampled to the original image size. The outputs of the up-sampling process are named $depth_3$, $depth_9$ and $depth_{27}$

Final depth map

Finally the final depth map DMAP(x, y) is created by a fusion of $depth_3$, $depth_9$ and $depth_{27}$. Therefore equation 2.4

$$\text{DMAP}(x,y) = \begin{cases} depth_{27}(x,y) \text{ if } \forall \text{diff}_j(x,y) < \text{TH}, j = 1,2,3 \\ depth_9 \text{ if } \forall \text{diff}_j(x,y) > \text{TH}, j = 1,2,3 \\ \text{avg}(\sum\limits_{i=3,9,27} depth_i(x,y)), \text{otherwise} \end{cases} \tag{2.4}$$

with equations 2.5 - 2.7

$$\text{diff}_1(x,y) = |depth_3(x,y) - depth_9(x,y)| \tag{2.5}$$

$$\text{diff}_2(x,y) = |depth_3(x,y) - depth_{27}(x,y)| \tag{2.6}$$

$$\text{diff}_3(x,y) = |depth_9(x,y) - depth_{27}(x,y)| \tag{2.7}$$

is used. In [KL11] for TH a value of 5 is recommended which provides good results.

(a) Original image **(b) Depth estimation by single resolution** **(c) Depth estimation by multi- resolution**

Figure 2.2: Depth estimation with single and multi-resolution [KL11]

In figure 2.2 one can see what the final depth map looks like. As one can see the small middle image is quite noisy. For this reason the multi-resolution is employed. Darker colours of the depth map correspond to nearer objects, whiter to further objects.

Results

Before we end the chapter with the section *Problems and limitations* some results are presented to show how the final depth maps of different outdoor scenes look like by applying this algorithm.

Figure 2.3: Resulting depth maps of Google map street view images [KL11]

In part (a) of figure 2.3 four Google map street view images are taken to create depth maps.
(b) shows Saxena's method [SCN08, SCN05, SSN07b, SSN07a]. His method works quite good, however, the second original image contains a cloudy sky which results in an inaccurate depth map. The cloudy part looks as far away as the street at the bottom of the image.
The last part (c) represents the results of the proposed algorithm [KL11]. The depth maps appear to be much smoother than in (b).

Problems and limitations

As seen in 2.3 the depth maps of the proposed algorithm [KL11] look quite smooth. However, most of the generated depths aren't accurate to ground truth. This can be seen in 2.2 in the right image. The hedge which fills half of the image should have the same depth value but that is not the case.
Moreover, the algorithm isn't suitable for all kind of images. First of all in [KL11] there isn't a precise definition of an outdoor scene. Most of the input images for the algorithm contain sky in the background. Thus, it is unclear if a sky in the background is necessary (e.g. for finding dark pixels) because outdoor images taken in a jungle with dense vegetation don't contain any parts of sky at all. In addition a cloudy sky leads to inaccurate depth maps because the algorithm doesn't recognise

these as a part of the sky. That can be seen in the third column in figure 2.3 where the upper half of the depth map consists of regions with smaller depth value. Though, the whole sky should have the same depth value which is near 1.
Furthermore, for images which contain the nearest objects at the top of the image the proposed algorithm [KL11] will fail because Tien-Ying Kuo et al. assumed that the nearest objects in outdoor scenes are at the bottom. Thus, a rotation of 180° using the images of figure 2.3 will probably result in wrong depth maps.
Finally the algorithm is only applicable for coloured images because the dark pixels are found by equation 2.1 which uses the three RGB channels. In fact a grey image as input isn't intended but in that case the depth map will not be accurate because the refinement of the final depth map strongly relies on dark pixels.

2.1.2 Depth Estimation for monocular videos with camera motion by Hu Tian et al. for dynamic scenes[TZH+13]

The algorithm of Hu Tian et al. is split into four parts: *background depth generation (BDG),motion analysis*, *moving objects extractions* and *depth fusion*.
As one can see in figure 2.4 the presented algorithm is based on a framework like the previously one to create a final depth map.

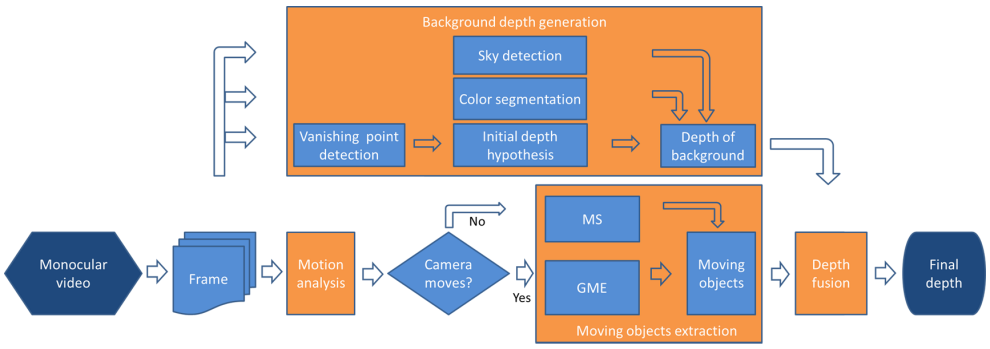

Figure 2.4: Framework for depth map out of monocular video [TZH+13]

Background depth generation (BDG)

The BDG focuses on the geometric perspective and sky detection. Geometric perspective consists of vanish lines and vanish points. Vanish lines are lines which are in reality parallel. However, in images they seem to converge in one point, the so-called vanish point. By following the vanish lines, distance is increasing gradually. This cue is used in this approach. Therefore, an algorithm for edge detection [MG01] and a Hough transformation [Bal81] are applied to find vanish lines in a frame. This procedure helps to find interceptions between these lines. The interceptions are clustered by k-means with K=3. Next the clustering centre (vanish point) is selected which contains the highest number of interceptions. If a vanish point exists, a gradient plane is created out of five depth hypotheses cases [BCLC+04, CLC10] which depend on the position of the vanish point and vanish lines. In figure 2.5 one can see how these gradient planes look like.
In contrast, if no vanish points exists a bottom-up depth hypothesis[CLC10] is employed. So far the image isn't divided into segments. Hence, a mean shift [CM02] for segmentation is applied and a hypothesis depth value to each segment is assigned.
Furthermore, a sky detection is used to improve the obtained depth map. The sky detector contains a support vector machine (SVM) [SV99] with RGB values and vertical coordinate of a pixel as input variables. In the case that the SVM detects sky in a region the highest depth value is assigned to that region.

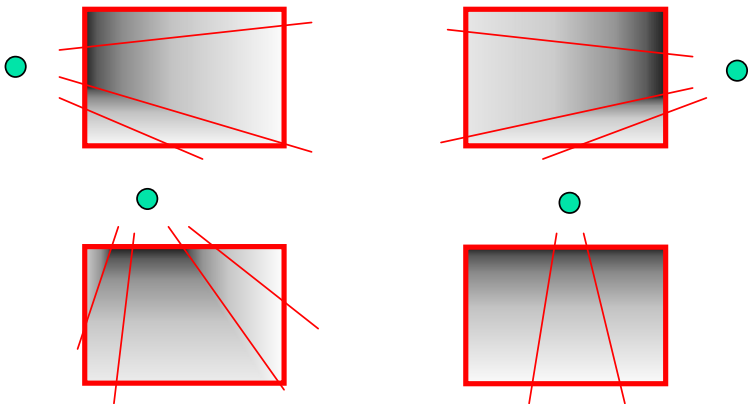

Figure 2.5: Gradient planes created depending on the position of vanish points and vanish lines [BCLC+04]

Motion analysis

In motion analysis one tries to distinguish between a camera in motion and a static camera. In order to find out which case occurs one has to calculate the sum of absolute difference (SAD) of four corner blocks between two consecutive frames. If no camera motion is detected a motion segmentation (MS) [KLK12] is employed. Otherwise the global motion estimation (GME) in the part below is used to find the parameter for a motion of the camera.
GME aims to find the parameter of the global motion of the camera. For this purpose a three-level frame pyramid is created by down-sampling two times with a factor of two [QGA08]. This reduces the computational complexity. Next, in each level an outlier rejection with a following parameter optimization is applied that will be explained in detail below. For the rejection process pixels at the bottom and at the boundary of the frame are used as the initial outliers. Then an optimization process is employed twice to obtain good parameters for the current level of the pyramid. The estimated parameters are projected to the next level and another outlier rejection process followed by an optimization process is applied. The final parameters at the top of the pyramid are used for the next frame. Figure 2.6 demonstrates how the GME method works.
I_n is the current frame and I_{n-1} is the previously frame. Frames which are denoted with $'$ are down-sampled frames.
The parameters which are optimized in the GME contain the global motion of the camera. Hence, in equation 2.8 a model is created to describe the global motion. The vector $\mathbf{a} = (a_0, a_1, a_2, a_3, a_4, a_5,)^T$ contains the global motion parameters and needs to be solved.

$$\begin{cases} x'_i = a_0 x_i + a_1 y_i + a2 \\ y'_i = a_3 x_i + a_4 y_i + a_5 \end{cases} \tag{2.8}$$

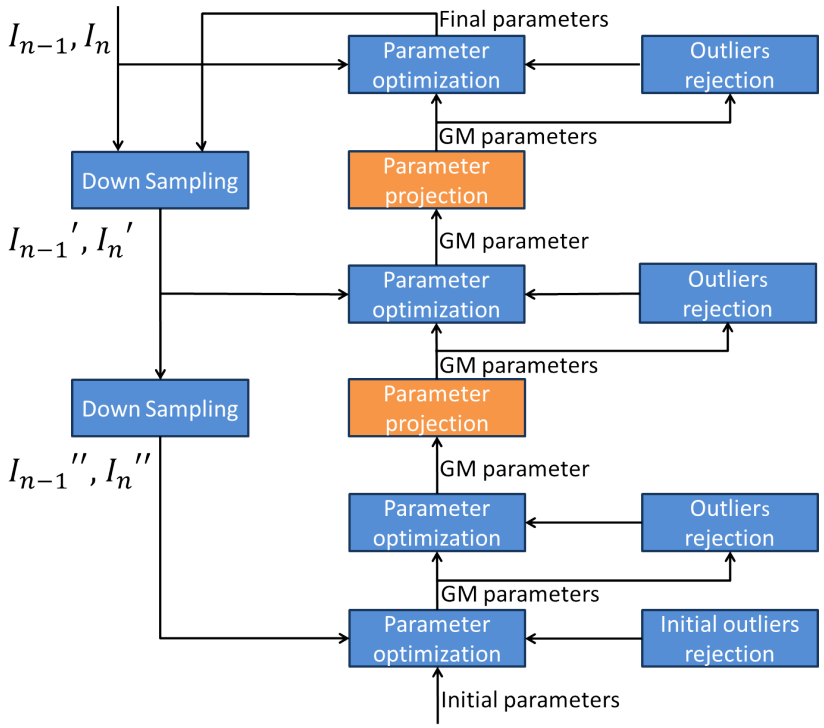

Figure 2.6: GME method using a three-level frame pyramid [TZH+13]

(x_i, y_i) are the coordinates of the i-th pixel in the current frame $\mathbf{I}$ and (x_i', y_i') are the coordinates of the previous frame $\mathbf{I}'$. In the case of projecting the lower level of the pyramid to the next higher one a_2 to a_5 are multiplied by a factor of 2. The remaining parameters stay unchanged. In order to solve $\mathbf{a}$ one has to minimize equation 2.9:

$$E(\mathbf{a}) = \frac{1}{2}\sum_{i=1}^{N}[\underbrace{\mathbf{I}(x_i', y_i') - \mathbf{I}(x_i, y_i)}_{r_i}]^2 \tag{2.9}$$

N is the number of pixels in $E(\mathbf{a})$ and r_i is the residual between the current and previous i-th pixel. Hu Tian et al. recommend to rewrite equation 2.9 to use Levenberg-Marquardt method [Mor78] because this method converges faster to a local minimum of $E(\mathbf{a})$ than the Gaussian-Newton method. Therefore, the vector $\mathbf{r}(\mathbf{a}) = (r_1, r_2, ..., r_N)^T$ is defined which contains N residuals r_i out of equation 2.9:

$$E(\mathbf{a}) = \frac{1}{2}\sum_{i=1}^{N}[\mathbf{I}(x_i', y_i') - \mathbf{I}(x_i, y_i)]^2 = \frac{1}{2}\mathbf{r}(\mathbf{a})^T\mathbf{r}(\mathbf{a}) \tag{2.10}$$

A minimization with the Levenberg-Marquardt method leads to a solution for vector $\mathbf{a}$.

So far, equation 2.9 takes all pixels in the frame $\mathbf{I}$ and $\mathbf{I}'$ into the optimization process. Hu Tian et al. noticed that residuals pixels on the border of the frame often change more strongly than pixels inside the region during each iteration of the optimization process. Thus, to reduce the computational complexity only pixels with large gradients are selected because GME benefits most of these values. However, sometimes these pixels are gathered in small region. Therefore, the image is divided into blocks and only the $\theta\%$ pixels in the block are taken.
In order to refine the optimization process an outlier rejection is applied. First, a motion compensated prediction (e.g. [WZG99]) is employed to find the different frame $\mathbf{D}$ between $\mathbf{I}$ and $\mathbf{I}'$. Then the $\rho\%$ distinct pixels of $\mathbf{D}$ are considered as outliers. In the next step an outlier rejection mask $\mathbf{B}$ is created. $\mathbf{B}$ contains all divided blocks b_m (with size $W * W$) of the image as seen in equation 2.11:

$$\mathbf{B} = \bigcup_{m=1}^{M} b_m \tag{2.11}$$

Moreover, the blocks b_m mustn't overlap and thus satisfy the condition 2.12

$$b_m \bigcap b_n = \emptyset, b_m \neq b_n, \text{ and } m, n \in \{1, 2, ..., M\} \tag{2.12}$$

where $\emptyset$ is empty and M the number of blocks. After that, $\mathbf{B}$ is divided into a set of outlier blocks $\mathbf{B}^O$, a set of inlier blocks $\mathbf{B}^I$ and a set $\mathbf{B}^B$ which contains all boundary blocks. In $\mathbf{S} = \{\mathbf{S}_{b_1}, \mathbf{S}_{b_2}, ..., \mathbf{S}_{b_M}\}$ all selected pixels of the current frame are gathered where $\mathbf{S}_{b_m}$ consisting of all selected pixels of the blocks. Furthermore, $\mathbf{S}_{b_m}$ is divided into $\mathbf{S}_{b_m}^{>T}$ and $\mathbf{S}_{b_m}^{\leq T}$. The threshold T is used to select the $\rho\%$ largest values of $\mathbf{D}$.
So far, no outlier rejection is applied to refine the optimization process. In order to find out whether a pixel is an outlier or an inlier one, the block b_m is classified as a candidate for outlier blocks $\mathbf{B}^{\widehat{O}}$ by using rule 2.13:

$$b_m \in \mathbf{B}^{\widehat{O}}, \text{ if } (\beta|\mathbf{S}_{b_m}^{>T}| > |\mathbf{S}_{b_m}|) \tag{2.13}$$

where β is a positive integer and $|\cdot|$ indicates the number of pixels in the set. Thus, all candidates which are potential outlier blocks are collected in $\mathbf{B}^{\widehat{O}}$. Then, three rules are employed to reject outliers:

1. If block b_m is in $\mathbf{B}^B$ it is an outlier:

$$b_m \in \mathbf{B}^O, \text{ if } (b_m \in \mathbf{B}^B) \tag{2.14}$$

2. The block b_m is an outlier if it has more than T_{ob} candidate blocks in its eight-neighbourhood:

$$b_m \in \mathbf{B}^O, \text{ if } (|\mathbf{B}_{b_m}^{b_l}| \geq T_{ob}) \tag{2.15}$$

where $\mathbf{B}_{b_m}^{b_l} = \{b_l : \text{ neighbour of } b_m\}$ contains all neighbours of block b_m

3. If the residual of a pixel in block b_m is bigger than T it is an outlier:

$$b_m \in \mathbf{B}^O, \text{ if } (i \in \mathbf{S}_{b_m}^{>T}) \tag{2.16}$$

where i is the pixel of block b_m.

These three rules can be merged into one condition 2.17 to select all pixels which participate in the optimization of equation 2.9:

$$\mathbf{S}^E = \{i | (i \in b_m) \wedge (b_m \in \mathbf{B}^I) \wedge (i \in \mathbf{S}_{b_m}^{\leq T})\} \tag{2.17}$$

Moving objects extraction

In case of a moving camera the method above results in GM parameters for the current frame. These parameters indicate a global motion. The residual frame contains local motions which can be found by binarizing the residual frame with a threshold T_{local}. In order to refine the extracted moving object, morphological operations including holes filling and closing operations are employed. Since morphological operations directly relate to shape, it is more efficient than applying convolution operations [DLfOES03].
In contrast if the camera is static a motion segmentation out of [KLK12] is used which is able to find moving objects.

Depth fusion

In the last step the depth of the moving object and the depth of the background are merged. Hence, the depth value of the bottom of the moving object is assigned to the depth value of the ground which is the point where the object touches the ground. A bilateral filter is applied to smooth the result.

Input image sequences

Background depth by geometric information | Extracted moving objects by GME | Final depth after fusion

Figure 2.7: Usage of GME framework [TZH+13]

In figure 2.7 one can observe how this fusion works. The small left image shows the result of background depth, the middle image shows the extracted moving object and the right image shows the fusion of the two depth maps. A closer look at the bottom of the extracted object indicates the same depth value as the background depth value.

Results

The results below are depth maps of different algorithms. For the depth map generation, several videos were taken which contain a local in-frame motion and a global camera motion. The first column of figure 2.8 is the original scene which is used as input and the last column contains the depth map of the proposed algorithm [TZH+13]. Darker pixels are indicating objects which are farther away.

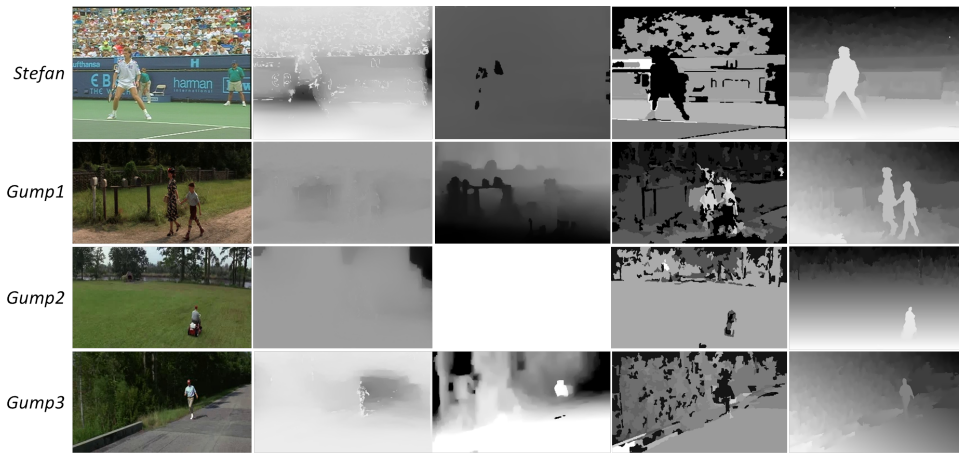

Figure 2.8: Depth map of four scenes by applying different algorithms. The last column contains the result of the proposed algorithm [TZH+13]

The proposed algorithm is able to create a decent depth map compared to the other algorithms in column 2 to 4 (the other algorithms can be found in [TZH+13, p.6]). Especially the moving objects in the frame are extracted clearly. However, the accuracy of the background depth map is not as good as it should be and should be improved.

Problems and limitations

Before we will end the chapter with the algorithm for dynamic scenes there are some problems which can occur by applying the approach of Hu Tian et al.
First, in scenes which don't contain obvious geometric information, an inaccurate BDG is created. Since in the *Depth fusion* state (in figure 2.4) the background depth map and the moving object depth map are combined, a wrong background depth map leads to unstable moving objects in time domain. The reason for this is that

the depth value of the bottom of the moving object is assigned to the wrong depth value of the background.
Furthermore, in *Motion analysis* the SAD of four corner blocks between two consecutive frames is applied to find out if a camera motion occurs. However, this method is unstable because the SAD becomes huge if a new object enters the scene through the corner blocks. Although the camera is static, the algorithm would detect a camera motion and thus would employ GME instead of a motion segmentation (MS) [KLK12]
In the next section the algorithm for static scenes [KL11] and the algorithm for dynamic scenes [TZH+13] are evaluated in accuracy, robustness and calculation time. The aim is to find out if they are feasible for an implementation.

2.2 Evaluation of the presented algorithms

In this short section both presented algorithms are examined in terms of accuracy, calculation time and robustness to find out if the algorithms are suitable for an implementation in real applications.

2.2.1 Algorithm by Tien-Ying Kuo et al. for static scenes

Accuracy and robustness

The accuracy of the algorithm is decent because for depth map generation the dark channel prior as well as saturation is applied which helps to improve the depth map. Therefore, saturation compensates areas where dark pixles fail [KL11]. However, in images which contain a cloudy sky the algorithm creates inaccurate depth maps for the sky because the framework doesn't use a sky detector. Thus, clouds are detected as normal objects instead of a part of the sky. This can be seen in figure 2.3 in the third column where the clouds don't have the same depth value of the sky.
Furthermore, the algorithm is optimized for outdoor scenes which contain the nearest objects at the bottom of the image. Thus, if one rotates the input image of an outdoor scene by 180° the algorithm will probably fail.

Calculation time

The algorithm of Tien-Ying Kuo et al. has a low complexity which leads to fast calculation times. Its performance is much better than in [SCN08]. Hence, it is useful for most of today's hand-held devices[KL11] but isn't optimized for real-time applications. An optimization like parallelization is possible. Therefore, the framework is divided into three independent processes for each image $I_{1/3}$, $I_{1/9}$ and $I_{1/27}$. The output of each process is fused together in the step *Fusion*.

Feasibility of implementation

The algorithm is easy to implement and low in complexity. Hence, it is very suitable for mobile devices but also for low level robots. Since the algorithm isn't optimized for real time processes it can only be used for slow applications which don't require fast responses.

2.2.2 Algorithm by Hu Tian et al. for dynamic scenes

Accuracy and robustness

Since the algorithm aims to estimate the global motion of the camera its accuracy is necessary to determinate the final depth map. The tests for performance in [TZH+13] show good results in mean absolute error (MAE) which reflects the precision of the estimated parameters. However, the implementation relies strongly on

geometric cues for BDG. In the case that these cues are non-existent the robustness of the algorithm will be poor and bad depth maps are created. Thus, objects will be extracted which are unstable in the time domain [TZH+13].
Furthermore, the algorithm can be unstable for new objects that enter the scene because it only takes the SAM of four corner blocks to find out if a camera motion exists. That means a small SAM leads to static camera motion and thus the usage of an algorithm for motion segmentation. In the case that a moving object enters the scene through the corner blocks the SAM becomes huge. The framework would pick GME to find the parameters of the camera motion although the camera is static. This fluctuation isn't considered in the framework and should be implemented in the future.

Calculation time

The approach of Hu Tian et al. is focused on low computational complexity. Hence, only a three-level frame pyramid is used. Furthermore, an outlier rejection is applied which supports GME to estimate the parameters of the global motion [TZH+13]. The algorithm is probably suitable for real-time applications because it is able to extract depth out of videos which are similar to camera inputs. Moreover a parallelization is practicable to decrease calculation time because BDG and motion analysis are two independent processes.

Feasibility of implementation

As seen above the algorithm is very suitable for an implementation in a real application. It is easy to implement and is feasible to implement for multi-core systems.

Chapter 3

Summary

At the end of the advanced seminar work, I want to give a short summary of the work. Two algorithms for monocular depth perception were presented, one for static scenes and one for dynamic scenes.

The first algorithm of Tien-Ying Kuo et al. for static scenes creates decent depth maps in an outdoor environment. Since the algorithm applies a multi-resolution framework and combines dark pixels with saturation the accuracy of the depth map is refined and thus greatly improved compared to the one without any refinement. However, the algorithm is constrained to outdoor images and the nearest objects have to lie at the bottom of the image.

The second algorithm of Hu Tian et al. for dynamic scenes creates accurate depth maps for dynamic scenes with a local in-frame motion and a global camera motion. The algorithm also employs a framework which is divided into a background depth generation (BDG) and a motion analysis. For the case of a global motion a novel approach for global motion estimation (GME) was proposed. However, problems can occur if the BDG contains wrong depth hypothesis which results in unstable moving objects in time domain.

So far both algorithms aren't used in robotics. Normally, algorithms for static scenes aren't useful for moving robotic applications because these aren't optimized for real-time applications. In case one aims to estimate depth in static scenes, the proposed algorithm [KL11] is suitable. Moreover, there are several other approaches e.g. [SMM11] or [SCN08] which are also able to create similar depth maps out of 2D images but their depth maps aren't that accurate and they are more computationally expensive.

In contrast, the proposed algorithm for dynamic scenes [KL11] is feasible for an implementation in a robotic application. In fact, the algorithm is optimized for frame-based videos but there isn't a big difference between a camera input and a video input. Thus, the algorithm should be suitable for real-time robot applications. Furthermore, there are more efficient algorithms which are optimized for high-speed applications. A good example is an algorithm of Jeff Michels et al. [MSN05] which is optimized to drive a high speed ($5\frac{m}{s}$) remote control (RC) car through an unknown

outdoor environment [1]. This approach uses only monocular depth cues with reinforcement learning to estimate depth and is able to avoid collisions with outdoor objects. However, the implementation of the algorithm is complex and thus this approach was not taken into the advanced seminar work.

[1] A video of their work can be found at `https://www.youtube.com/watch?v=UZ7_ED9g4FY` (opened on 02.07.2015)

List of Figures

Bibliography

[Ago05] Max K.. Agoston. Computer Graphics and Geometric Modeling: Implementation and Algorithms. Springer, 2005.

[Bal81] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. Pattern recognition, 13(2):111–122, 1981.

[BCLC+04] Sebastiano Battiato, Salvatore Curti, Marco La Cascia, Marcello Tortora, and Emiliano Scordato. Depth map generation by image classification. In Electronic Imaging 2004, pages 95–104. International Society for Optics and Photonics, 2004.

[CC06] Brian R Cavoto and Robert G Cook. The contribution of monocu-lar depth cues to scene perception by pigeons. Psychological Science, 17(7):628–634, 2006.

[CLC10] Chao-Chung Cheng, Chung-Te Li, and Liang-Gee Chen. A 2d-to-3d conversion system using edge information. In Consumer Electronics (ICCE), 2010 Digest of Technical Papers International Conference on, pages 377–378. IEEE, 2010.

[CM02] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24(5):603–619, 2002.

[Cro05] THOMAS W Cronin. The visual ecology of predator-prey interactions. Ecology of predator-prey interactions, pages 105–138, 2005.

[CS02] Jörg Conradt, Pascal Simon, Michael Pescatore, and Paul FMJ Verschure. Saliency Maps Operating on Stereo Images Detect Landmarks and their Distance, Int. Conference on Artificial Neural Networks (ICANN), 2002, p. 795-800, Madrid, Spain.

[CxYh09] Wang Chuan-xu and Sun Ying-he. A new method of depth measurement with binocular vision based on surf. In Computer Science and Engineering, 2009. WCSE'09. Second International Workshop on, volume 1, pages 568–571. IEEE, 2009.

[FS97] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences, 55(1):119–139, 1997.

[How12] Ian P Howard. Perceiving in depth, volume 1: basic mechanisms. Oxford University Press, 2012.

[HST11] Kaiming He, Jian Sun, and Xiaoou Tang. Single image haze removal using dark channel prior. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 33(12):2341–2353, 2011.

[KL11] Tien-Ying Kuo and Yi-Chung Lo. Depth estimation from a monocular view of the outdoors. Consumer Electronics, IEEE Transactions on, 57(2):817–822, 2011.

[KLK12] Kevin Karsch, Ce Liu, and Sing Bing Kang. Depth extraction from video using non-parametric sampling. In Computer Vision–ECCV 2012, pages 775–788. Springer, 2012.

[Kra03] Karl Kral. Behavioural–analytical studies of the role of head movements in depth perception in insects, birds and mammals. Behavioural Processes, 64(1):1–12, 2003.

[MC11] Georg R. Müller, Jorg Conradt. A Miniature Low-Power Sensor System for Real Time 2D Visual Tracking of LED Markers, Proceedings of the IEEE International Conference on Robotics and Biomimetics (IEEE-ROBIO), 2011, pages 2429-35, Phuket, Thailand.

[MG01] Peter Meer and Bogdan Georgescu. Edge detection with embedded confidence. Pattern Analysis and Machine Intelligence, IEEE Trans-actions on, 23(12):1351–1365, 2001.

[Mor78] Jorge J Mor´e. The levenberg-marquardt algorithm: implementation and theory. In Numerical analysis, pages 105–116. Springer, 1978.

[MSN05] Jeff Michels, Ashutosh Saxena, and Andrew Y Ng. High speed obsta-cle avoidance using monocular vision and reinforcement learning. In Proceedings of the 22nd international conference on Machine learning, pages 593–600. ACM, 2005.

[Ots75] Nobuyuki Otsu. A threshold selection method from gray-level histograms. Automatica, 11(285-296):23–27, 1975.

[QGA08] Bin Qi, Mohammed Ghazal, and Aishy Amer. Robust global motion estimation oriented to video object segmentation. Image Processing, IEEE Transactions on, 17(6):958–967, 2008.

[SCN08] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. 3-d depth reconstruction from a single still image. International journal of computer vision, 76(1):53–69, 2008.

[SL11] Dongcheng Shi and Yinghuan Li. Depth extraction method based on binocular stereo matching. In Image and Signal Processing (CISP), 2011 4th International Congress on, volume 3, pages 1420–1423. IEEE, 2011.

[SMM11] Yasir Salih, Aamir S Malik, and Zazilah May. Depth estimation using monocular cues from single image. In National Postgraduate Conference (NPC), 2011, pages 1–4. IEEE, 2011.

[SSN07a] Ashutosh Saxena, Jamie Schulte, and Andrew Y Ng. Depth estimation using monocular and stereo cues. In IJCAI, volume 7, 2007.

[SSN07b] Ashutosh Saxena, Min Sun, and Andrew Y Ng. 3-d reconstruction from sparse views using monocular vision. In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pages 1–8. IEEE, 2007.

[SV99] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. Neural processing letters, 9(3):293–300, 1999.

[TZH+13] Hu Tian, Bojin Zhuang, Yan Hua, Yanyun Zhao, and Anni Cai. Recovering depth of background and foreground from a monocular video with camera motion. In Visual Communications and Image Processing (VCIP), 2013, pages 1–6. IEEE, 2013.

[vBMD14] Floris van Breugel, Kristi Morgansen, and Michael H Dickinson. Monocular distance estimation from optic flow during active landing maneuvers. Bioinspiration & biomimetics, 9(2):025002, 2014.

[WZG99] Thomas Wiegand, Xiaozheng Zhang, and Bernd Girod. Long-term memory motion-compensated prediction. Circuits and Systems for Video Technology, IEEE Transactions on, 9(1):70–84, 1999.

License