Motion estimation

Moritz Loos 112385

Winter term 2014/15

1 Abstract

The goal of the project was to estimate the motion of a UAV using a stereo camera system with help of the optical flow in the stereo images.

2 Introduction

The term optical flow describes the tracking of points from one image to another. Generally there are two possible approaches: At first we try to track every single pixel from frame to frame (called optical-field matching in the following). In order to reduce the number of points to track we can use an interval to determine points to track (e.g. tracking every tenth pixel within every frame). The second approach would be tracking defined features instead of every pixel in every frame(feature-based matching).

(a) optical-field-matching   (b) feature-based-matching

Using optical-field matching the number of points is much higher than in feature-based matching. Also the distribution of points is much better. On the other side the number of outliers as well as the computational effort are growing.

1

In order to make the framework portable and stable (for the future use as a method for realtime flight-path detection) the basis is a feature-based matching algorithm.

Unfortunately a simple mapping of optical flow vectors to the motion of a camera is just possible when some special constraints are fulfilled [1].

1. The scene has to be static

2. The scene has to be parallel to the image plane of the camera

3. The scenes height has to be predefined.

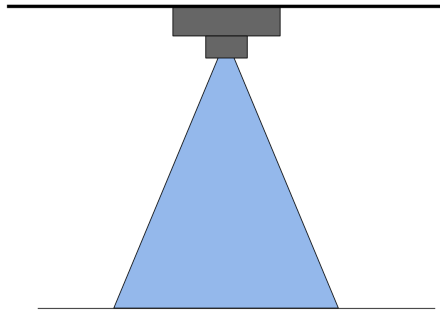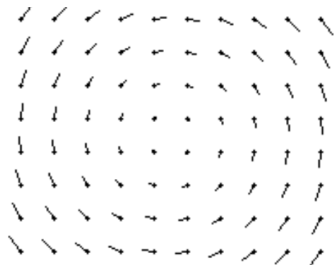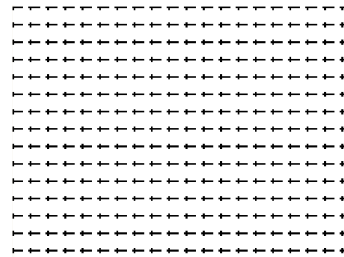That means the camera has to look down permanently (see figure 2).

Figure 2: camera constraint

If all these constraints are fulfilled it is possible to estimate the motion of the camera just using the length and the direction of the optical flow vectors.

(a) rotation

(b) translation

Figure 3: optical flow [1]

2

As it is not possible for us to hold this constraints we had to find other approaches.

3   Settings

The system we are using is a calibrated stereo rig [2] and all values are defined like described in figure 4. Where P represents the transformation matrix of the current camera(P_L is defined as the projection matrix from the left camera in previous frame i-1 to the left camera in current frame i). C describes the previous respectively current camera image.
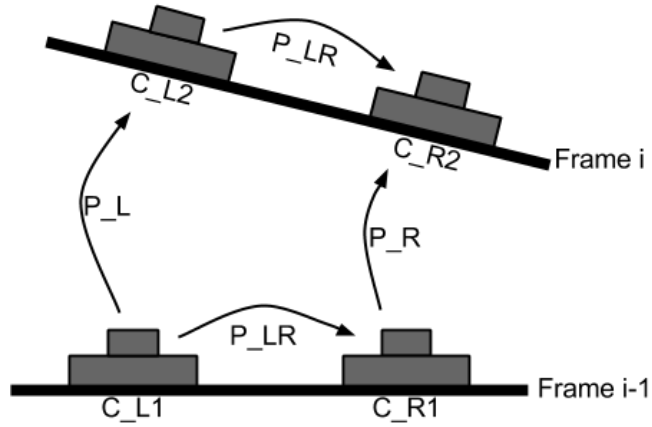

Figure 4: camera setup

4   Visual odometry

In order to estimate the motion we need to find corresponding points in all 4 images. Our approach for performing visual odometry is represented in figure 5 and is explained in the following.
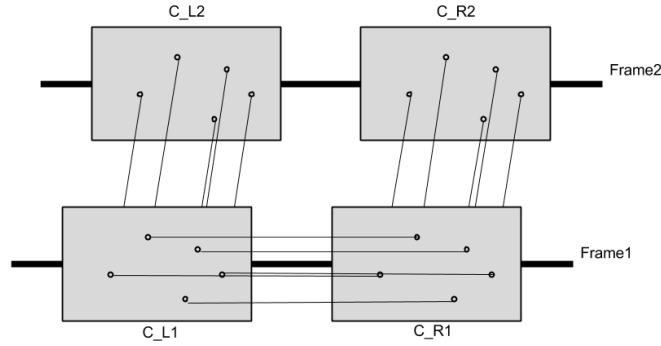

Figure 5: image setup

We start to detect features in C_{L1} with the Shi-Tomasi Corner detector that is implemented in OpenCV. The Shi-Tomasi approach is a faster and more stable modification of Harris Corner detector [3].

Features (points of interest) are characterized by their gradient magnitudes and gradiant orientations in a local neighborhood. With analyzing the eigenvalues of the structure tensor using a window pattern around a pixel is it possible to decide if the point is a corner, an edge or neither of them.
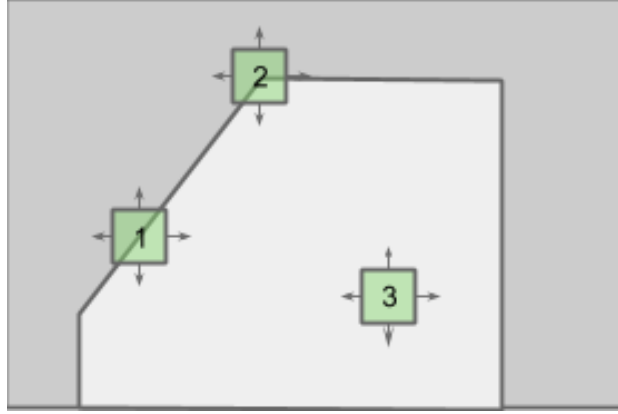
Figure 6: Shi-Tomasi approach

In figure 6 are shown different window pattern with different gradient. The first pattern is an edge because there is no change of gradient along the edge direction. Second pattern is a corner because there are changes in both directions. Last pattern has no change in any direction and is for this reason just classified as a flat surface (no feature) [[4, 5]. The Shi-Tomasi Corner detector is a really common approach, but an alternate could be the FAST feature detector or SURF (Speeded-Up Robust Features).

After finding the features in C_{L1} the algorithm tries to find the corresponding points in C_{R1} and C_{L2}. Subsequently the points found in C_{R1} are matched with the ones of C_{R2}.
To refind points in different images we use the Lucas-Kanade algorithm that is also part of the OpenCV library.

Beside SURF the Lucas-Kanade Algorithm is one of the most common methods to find features in an image. One requirement for this method is, that the intensity of a point in a specified window will not change from frame to frame. That is the main reason why the algorithm works best on systems with a high frame rate, so that the movement between the single frames is very low [6].

In this project we used the pyramidical approach of this algorithm. Therefore the reference image is downscaled to a certain number of pyramidal layers (see figure 7) with different resolutions. Afterwards the Lucas-Kanade Algorithm is called on each layer in ascending order (only in the range found previously). Using this method the algorithm is way faster, more robust and provides a better accuracy [7].
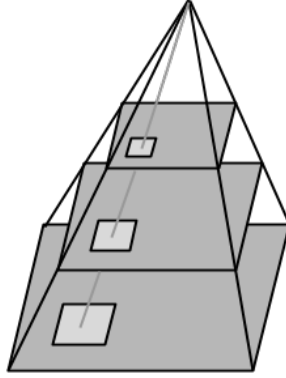
5

Figure 7: pyramedical approach of Lucas-Kanade

In order to prevent wrong associations between points, outliers have to be deleted. Therefore OpenCV provides a method to find the fundamental matrix using RANSAC [8]. Although the method is doing what it is supposed to do, it finds too many good points (inliers) and declares them as outliers, and vice versa. Because it is quiet difficult to retrace the wrong definition of points we have to define inliers manually. One good approach therefore is the constraint of having a rectified stereo camera system. The flow vectors between C_{L1} and C_{R1} as well as C_{L2} to C_{R2} have to be horizontal on the images (epipolar geometry). If that is not the case we delete this point in all four frames. The following figure shows the inliers, represented as green arrows as well as the outliers represented as red arrows.
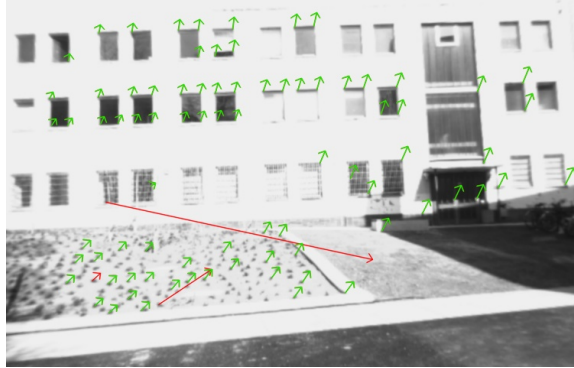

Figure 8: inliers (green) and outliers (red)

5   Motion Estimation

Estimating motion between images is a hard task with many different approaches. Most of them use the projection matrix computed out of the essential matrix. The matrices we need are computed separately (left and right) between the current and the previous frame using detected features and correspondence points. First we need to compute the fundamental matrices for both sides (using the OpenCV method described before with RANSAC). Next we have to calculate the essential matrices using the camera matrices we get from the rectification using following formula [9].

$$E = K' \cdot F \cdot K \tag{1}$$

To obtain the perspective matrix we have to decompose the essential mat using SVD [10]. Unfortunately the essential matrix can be decomposed to 4 different projection matrices and we have to find the right one. Therefore the points are triangulated using all four possible projection matrices with the goal to find the one where all points are in front of both cameras [11].
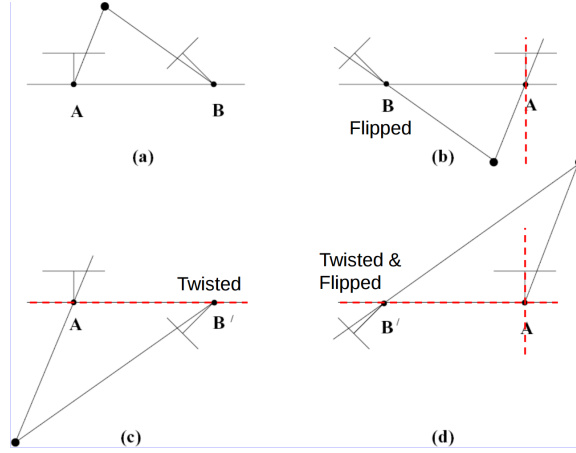

Figure 9: Decomposing of essential matrix [11]

One problem left is that the found projection matrix is up to scale, which basically means that we just get the normalized direction vector ($||T|| = 1$) and the rotation of each camera. In order to get the correct scale factors we have to triangulate a reference point cloud from the left to the right camera called X. This point cloud is in the right scale because it is computed using P_{LR} which already contains metric values. Next we triangulate points from the previous left image to the current left image resulting X_L, the same

procedure is used to calculate X_R. To estimate the scale factor for L and R we just compare the point clouds with the reference X as the following formulas show [12].

$$\mu = \frac{1}{n} * \sum_{i=1}^{n} \frac{\|X\|}{\|X_L\|} \qquad \lambda = \frac{1}{n} * \sum_{i=1}^{n} \frac{\|X\|}{\|X_R\|} \qquad (2)$$

Improving the estimation can be done by skipping frames where the motion estimation seems to fail and try to estimate the motion directly to the next valid frameset. To decide if the frameset is valid or not we can check against certain constraints. One of them is checking whether the determinant of E is zero, the determinant of a valid rotation matrix should be either positive or negative one [13]. To determine a good fundamental matrix we need as much valid inliers as possible (a specific number depends on various factors like image quality, size etc.). Also we defined a maximum motion per frame as a maximum of one meter, because moving more than one meter with a frame rate of either 30fps (binned) or 60fps is not realistic and can be declared as wrong. If one of this constraints cannot be fulfilled we skip the frameset and check the next one always remembering the last valid frameset. The method is robust as long we do not skip more than 4 frames in a row because of the increased distance between the taken pictures and the resulting distance between corresponding points. A reason for that is that the Lukas Kanade [6] algorithm does not work properly with high distances between corresponding images. If there is a circular motion a correspondence analysis is nearly impossible.
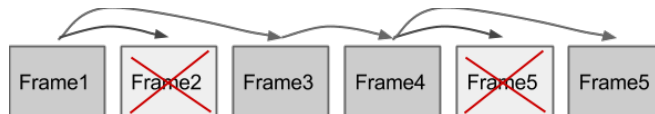

Figure 10: approach of skipping frames

As described above there exist a lot of methods in the field of visual odometry. Beside the presented one we implemented two other approaches. The Perspective-n-point method is another way to estimate motion using a stereo-vision system. Therefore we have to find 3D-point to 2D-image-point correspondences. These are easy to gather using triangulation. Fortunately this method is included as RANSAC in OpenCV. The main idea is to use the dependency between 2D and corresponding 3D points. 2D points are here described as the multiplication of the projection matrix and the 3D

points as explained in the following formula [14].

$$x_i = P * X_i \qquad x_i := \text{2D points} \tag{3}$$
$$X_i := \text{3D points}$$

Second approach is to compare the orientation of the point cloud triangulated in frame i-1 to the point cloud triangulated in frame i (see figure [11]). The whole method is explained in the paper "Combining Stereo Vision and Inertial Navigation System" [15] and is reimplemented by us.
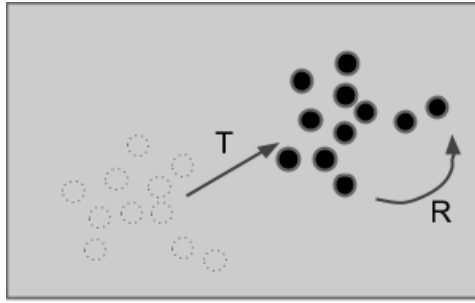

Figure 11: approach of pointclout orientation

6   Results

Using the first approach we gathered some useful results. We used the stereosystem connected to a notebook and walked around the Digital-Bauhaus-Lab in Weimar. Visually the estimated path matches to the one we walked. But as we take a deeper look on the results we can see that the start and end position is not matching though the test began and end in the exact same location. To express the added error in numbers we can take a look at the coordinates estimated. We started at [0, 0, 0] and stopped at [-0,23m, -5.46m, -1.9m], which means we drifted by ± 6m. At this point we were not able to gather any useful results using the other two approaches.

Figure 12: path around the DBL in Weimar

7 Issues

There are many different factors for good results when trying to estimate the motion out of stereo camera images. Obviously the corresponding points and their accuracy are the most important factor. The quality as well as the distribution of points are two very important components, but also the number of correct points. For example, if all points lie on a plane it is not possible to compute the fundamental matrix. For a good triangulation it is essential that the found features are not too far away (e.g at the horizon) or too close together (too small translations can also affect the quality of the triangulation). The reason for that is the small distance between the points, so it is really hard to differ which feature belongs to another within different frames and camera images. Whereby the same problem occurs with too small translation. Another issue which can lead to problems is the size of the baseline between the cameras. If it is too small the triangulation might fail because of very low angles, if it is too big the number corresponding points can be to less.

8 Future work

In order to make the algorithm more robust it is important that the error from frame to frame does not grow more. Bundle adjustment could be a useful approach to solve this problem. A comparison between these three methods would be useful to evaluate the best method for a specific setup. Another idea would be to combine the methods to get a system which is able to cope with very different situations. The current code is basically written to be used in a comfortable way, therefore an implementation with focus on

performance and stability would be important. Live test should be also done in order to evaluate the current performance and robustness.

References

[1] Uni-Muenchen. Bewegungsanalyse: Optische flusse. http://cvpr.uni-muenster.de/teaching/ss09/computerVisionSS09/script/CV08-Bewegungsanalyse.pdf.

[2] Hiller Malik. Fast depth map estimation from stereo camera systems. not_published_yet.

[3] Wikipedia. Corner detection. https://en.wikipedia.org/wiki/Corner_detection.

[4] Aishack. The shi-tomasi corner detector. http://www.aishack.in/tutorials/the-shitomasi-corner-detector/.

[5] Marshall Tappen. Lecture 5: Feature detection. http://www.cs.ucf.edu/~mtappen/cap5415/lecs/lec5.pdf.

[6] Johannes Brndle Christian Eppler, Stefan Mbius. Report - digitale bildverarbeitung 2014 - bildfolgenverarbeitung. http://www.informatikprojekt.de/uploads/dbv_report.pdf.

[7] Patrick Westfeld. Monokulare menschliche bewegungsanalyse in bildsequenzen. http://tu-dresden.de/die_tu_dresden/fakultaeten/fakultaet_forst_geo_und_hydrowissenschaften/fachrichtung_geowissenschaften/ipf/photogrammetrie/dateien/Westfeld2004.pdf.

[8] OpenCV. Camera calibration and 3d reconstruction. http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html.

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

[10] Carl Olsson. Lecture 6: Camera computation and the essential matrix. http://www.maths.lth.se/matematiklth/personal/calle/datorseende13/notes/forelas6.pdf.

[11] Volker Rodehorst. Lecture photogrammetric computer vision. http://www.uni-weimar.de/fileadmin/user/fak/medien/professuren/Computer_Vision/Downloads/intern/WS/Photogrammetric_Computer_Vision/lecture/09_pcv081214.pdf.

[12] Volker Rodehorst. Interessantes paper. `todo..`

[13] Wikipedia. Essential matrix. http://en.wikipedia.org/wiki/Essential_matrix#Properties_of_the_essential_matrix.

[14] Slobodan Ili. Tracking and detection in computer vision - camera models and pose estimation. http://campar.in.tum.de/twiki/pub/Chair/TeachingWs09MATDCV/camera_models_and_pose_estimation.pdf.

[15] LuisRodolfo Garca Carrillo, AlejandroEnrique Dzul Lpez, Rogelio Lozano, and Claude Pgard. Combining stereo vision and inertial navigation system for a quad-rotor uav. *Journal of Intelligent Robotic Systems*, 65(1-4):373–387, 2012.