Stereo vision with consumer grade high resolution cameras for a micro air vehicle

Joonas Melin,[*] Mikko Lauri, and Risto Ritala
Tampere University of Technology, Tampere, Finland

ABSTRACT

We construct a stereo vision system mounted on a micro air vehicle applying two high resolution consumer grade cameras. The system hardware and mechanical configuration are presented. A novel image processing algorithm specifically suited for high resolution stereo images is described, and its properties are discussed. Empirical data from outdoors flight experiments is presented, showing both successful and unsuccessful attempts at extracting depth information from the stereo images. The results highlight several important design aspects to consider when building a stereo vision system for use in micro air vehicles.

1 INTRODUCTION

Depth information is essential for representing real world structures in three dimensions. Depth information represented as a set of 3D points in a known frame of reference is called a point cloud. Point clouds or other 3D representations are essential for autonomous robot operations, since avoiding obstacles, planning routes or manipulating objects in the environment is impossible without 3D information about the surroundings.

Depth data can be obtained, for example, with laser range finders or time of flight (TOF) cameras. However, these methods may not be suitable for micro air vehicles (MAVs) due to weight and cost limitations.

Stereo vision provides point clouds by first finding the matching features from two images taken from two known but different locations. The difference between the matching features is called disparity. Disparity can be converted to a metric distance when the geometric transformation between the images is known. The features are usually block shaped areas in the images. The most difficult part in stereo vision is finding the matching features in the image pairs. Many of the current methods struggle with featureless surfaces, see e.g. [1]

Low resolution stereo imaging as used in [2] and [3] has the advantage of fast processing and low latency, critical for real time applications. For instance, low-resolution imaging can be used for attitude estimation and navigation when distance to the features is generally under 5m and surfaces have unique texture. On the other hand, these systems are not ideal for mapping areas with small featured surfaces from more than 5m away since the low resolution cameras cannot distinguish texture at distance. This performance is highly dependent on the construction of the stereo camera and the surface in question: plainly painted wall might not have enough features no matter how close the camera is, and the wall of the building with large features might have enough texture for low resolution stereo matching even from more than 20m away. High-resolution stereo imaging can be applied to obtain reliable depth information from feature poor environments such as sand, grass or asphalt surfaces. Another advantage of high resolution images is that they can also be used for tasks such as texture analysis or marker detection. The downside is increased computation time, which makes high resolution data challenging to use in real time applications. In this work, we apply a lightweight, low-cost, stereo camera system built from high-resolution consumer grade cameras to collect depth data. The stereo vision system is mounted on a MAV. Our goal is to apply the system to support the activities of an unmanned ground vehicle (UGV) by providing information on terrain features and formation in areas that the UGVs own sensors cannot reach. The remainder of the paper is organized as follows: in Section 2, we present the MAV hardware construction including the stereo camera system and give a brief overview of the theory of stereo imaging. In Section 3, the image processing techniques developed in this work specific to the use of high-resolution cameras are presented. Their differences to other state-of-the-art methods are discussed. In Section 4, empirical data collected from MAV test flights with the stereo camera system are presented. The data highlight several important design aspects that have an effect on stereo imaging performance. Finally, Section 5 concludes the paper.

2 METHODS

The MAV applied in this work is shown in Figure 1. It is built from carbon fiber honeycomb plate and carbon fiber tubing. The arms and their connections are from a commercial kit but the rest of the hardware has been cut from carbon fiber plates.

The system consists of subsystems which each handle their specific tasks. The main PC coordinates all the subprocesses. The layout and communication hierarchy is presented

[*]Email address(es): joonas.melin@tut.fi, mikko.lauri@tut.fi, risto.ritala@tut.fi

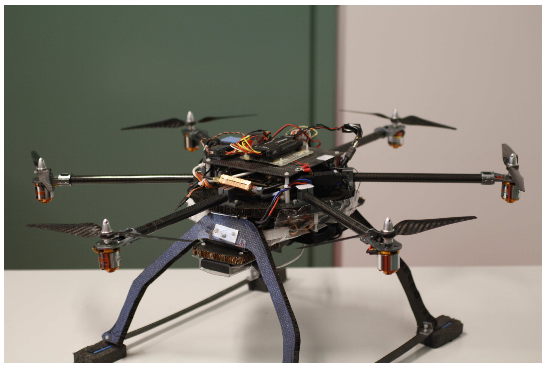


Figure 1: The MAV used in this work.

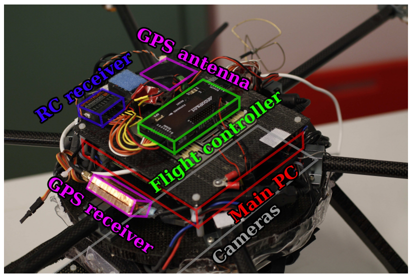


Figure 2: The physical layout of the components of the MAV system.

in Figure 3. The physical placement of the systems can be seen in Figure 2.

We are using the open source Arducopter [4] as our flight controller. An additional inertial measurement unit (IMU) is used for the camera attitude estimation. The IMU unit is built from Arm M4 microcontroller and Invesense MPU-6050 accelerometer and gyroscope. MAV location is sensed with a global positioning system (GPS) sensor. The GPS sensor consists of a Yuan10 receiver with RTKlib open source RTK-GPS software[5] in the main PC. The positional error of the GPS is usually smaller than 0.5m in motion, but can be as high as 10m when close to ground or buildings. The closer analysis of the positioning system can be found in [6].

The stereo camera is built by mounting two standard consumer cameras on a sandwiched honeycomb composite plate. This mounting ensures that the cameras share a common plane. A similar setup is used e.g. in [7]. Our stereo rig has a baseline of 219 mm, compared with 700 mm in [7].We use a small baseline because our MAV could not accommodate a larger camera rig. The cameras are running custom firmware that provides software triggering, raw image capturing and basic scripting capabilities. The cameras are triggered by a simple script running in the camera firmware. The camera rig is shown in Figure 4. Cameras are mounted in such a way that we get a maximal baseline with minimal space usage; however this results in a different shutter sweep direction between the cameras. Because we were using the raw images

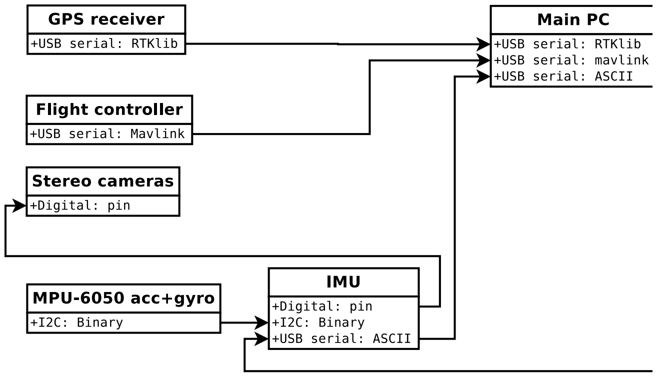


Figure 3: A schematic of the onboard hardware and their connections.

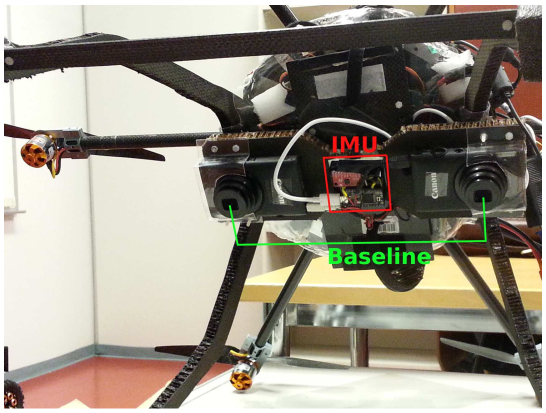


Figure 4: The onboard camera rig.

provided by the custom firmware, we were able to get 10bit images without cropping and other corrections done by the original firmware. However, this also meant that we had to do the factory calibration ourselves, which involved the geometric rectification of the images and removing the dead or stuck pixels from the images. The cameras also had varying triggering delay to the trigger signal. Constant delay was roughly 700ms which was compensated in the camera triggering code, but the variance of this delay proved to be a critical aspect on the experimental data. The triggering difference between the cameras for a sample of ten image pairs is presented in Figure 5.

To perform any kind of mapping, we need to define our coordinate systems and be able to transform data from the sensor plane into the map coordinates. Image processing must be done in a frame relative to the sensor plane and map information is most conveniently represented in a global frame of reference. In the case of the stereo camera, we need at least three coordinate systems: the image, camera and global coordinate systems. The origin of the image coordinate system is at the principal point, which is in the middle of the sensor. Image coordinates have units in pixels. The

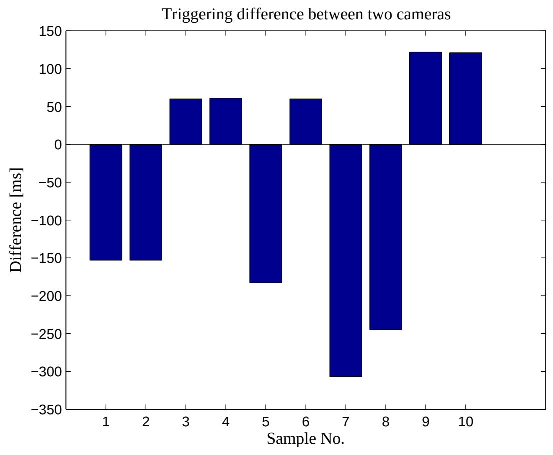


Figure 5: Triggering time differences measured from ten image pairs.

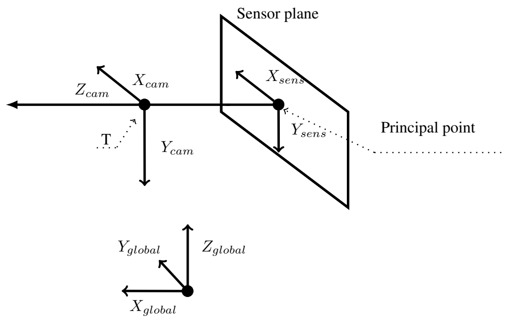


Figure 6: The coordinate systems applied.

camera coordinate system has its origin at the focal point of the camera. The global coordinate system is a fixed earth-centered frame. The coordinate systems are presented in Figure 6, where the coordinate axes with subscript sens, cam and glob represent the image, camera and the global coordinate frames, respectively.

Stereo camera is calibrated to rectify the images so that both of the images appear as if they only have translation in horizontal (X) direction. This means that if a feature is found at coordinate $[x_1, y_1]$ on the left image, it will found on the right image at a coordinate $[x_1 + d, y_1]$ where the term d is called the disparity. Stereo camera calibration also includes the traditional single camera calibration where the lens distortions are compensated.

There are several methods for finding disparities between images. One of these methods is called block matching. Block matching works by selecting a small rectangular block from the left image and slides it horizontally over the right image, calculating correlation over each position. An alternative version of this typically used in low-resolution systems is the sum of absolute differences, where instead of correlation, difference between the blocks in the images is used. The point of maximal correlation or the minimal point of summed absolute differences is chosen as the disparity between the images at the origin of the block.

Once the disparity between the images has been calculated at several points, the pixel units in disparities d are converted to metric depth z according to[8, p. 175]

$$z = \frac{d_t f}{d}. \tag{1}$$

This equation is valid in the special case when the cameras are on the same plane. The baseline d_t is the distance between the camera centers. When this information is combined with the focal length f and disparity d we can calculate depth in meters. The focal length f is determined by camera calibration in pixel units.

By this process, we have obtained the metric depth data on the camera coordinate frame. The data is then converted to the global coordinate frame. This requires knowing the rotation and translation of the cameras when the images were taken. Sensor data from GPS, IMU, etc. are used to estimate the rotation and translation parameters.

3 Image processing

Image processing was first tested with readily available tools like the machine vision toolbox [9]. However, these tools were not always able to handle high resolution data or resulted in low quality image matching. Due to this, we implemented our own stereo matching algorithm specifically suitable for high-resolution images.

We apply a block matching algorithm to find the disparities between the images. Our approach works iteratively, alternating the block size and using the information gained on previous steps to constrain the search areas. The idea behind this method was to utilize all available data with little concern at computation times and concentrate on computational efficiency at a later stage. The algorithm is described in Figure 7. The process begins by loading the images from the disk, followed by the camera calibration procedures which involve the geometric rectification and correcting for faulty pixels. After the calibration steps, we use principal component analysis (PCA) to reduce the RGB color channels to an monochrome image. The disparity between the images is searched with the block matching algorithm. When the disparity is found, the disparity image is converted to point cloud format in global coordinates.

3.1 RGB to monochrome conversion

Stereo vision algorithms are generally implemented on monochrome images as this reduces calculation time. Therefore, our RGB data needs to be converted to monochrome. Traditionally, the RGB image is converted to monochrome by summing the color channels with different gains, roughly equivalent to discarding hue and saturation information from the image. We applied windowed PCA [10] over the whole image in this work. This method finds optimal multipliers

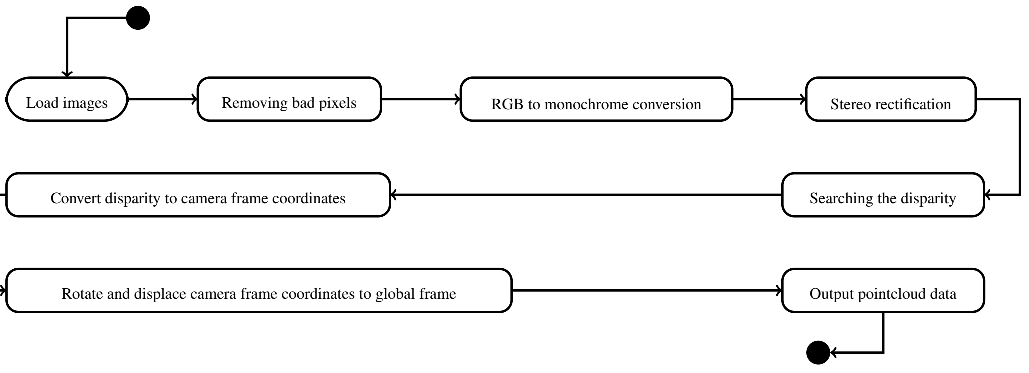

Load images
Removing bad pixels
RGB to monochrome conversion
Stereo rectification
Searching the disparity
Convert disparity to camera frame coordinates
Rotate and displace camera frame coordinates to global frame
Output pointcloud data

Figure 7: The algorithm for calculating point clouds from stereo images.

for each color channel by computing the PCA transformation to each window so that local contrast is maximized. Comparison of the traditional monochrome conversion and PCA transformation can be seen in Figure 8. The figure shows the worst case scenario for the traditional RGB conversion and has been designed to illustrate a special case when the traditional conversion is not viable. The figure also includes a comparison with a real photograph. Differences are visible in places where colors such as red and green are close to each other. Traditional conversion results in low contrast between the two colors whereas PCA transformation enhances the differences.

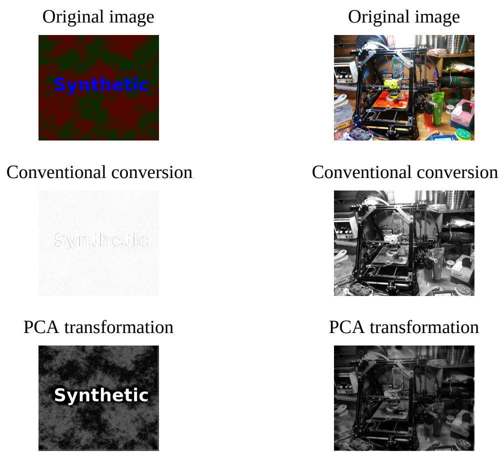


Figure 8: RGB to monochrome conversion for a synthetic (left column) and real image (right column). The source image is shown on the top row. The middle row shows the traditional RGB conversion of the source image and the bottom row the converesion with the windowed PCA transform.

The processing takes 5 seconds for an 8Mpix image with and I7-3990 processor using a window size of 40 pixels. The algorithm was implemented usingMatlab's builtin PCA transformation.

3.2 Block matching

We use a block matching algorithm for solving the disparity between the images. A diagram of our block matching algorithm is shown in Figure 9. This algorithm corresponds to the Searching the disparity part of the whole image processing procedure shown in Figure 7. The block matching algorithm works iteratively by first calculating an initial estimate for the image using low resolution and large block sizes. The following iterations will use decreasing block sizes and higher resolution. The decision to replace the results from the previous rounds is done based on cost functions that determine the quality of the match. The algorithm is essentially correlation based block matching with heuristic decision making between different block sizes. The algorithm begins by setting initial values for disparity and the uncertainty of each pixel. In the next phase, the algorithm chooses the block size and resolution to be used in calculation. The search window in the other image will scale according to the uncertainty for that particular disparity. Then, normalized cross correlation is used to calculate the correlation of the block over the search window. This is followed by calculating the different cost functions or descriptors for the disparity. The rest of the algorithm computes heuristic values based on the cost functions for deciding if the disparity calculated in this iteration is better than the one calculated last time. The decision logic is illustrated in the lower part of the Figure 7. For example, in the case where two out of four of the cost functions indicate that the new disparity value would be better than the previously calculated disparity, the old value will be updated with a new disparity value if the window size used was smaller for the new window. The old estimate will also be updated if three or more of the four cost functions indicate that the new estimate is better. The whole algorithm will iterate as long as the percentage of points updated is over a stopping threshold.

The cost functions used in this work fall into two categories. The first category contains pixel based functions, and the other category contains functions that need to be converted to pixels from an unitless quantity, such as correlation.

Uncertainty in pixel units can be converted to a metric uncertainty through the same equations that are used for converting the disparity.

Pixel based cost functions penalize the match if it has too many peaks that are caused e.g. by repetitive texture. Results from larger block size will be preferred if they do not have the multi peak correlation. Standard deviation with the neighboring disparity values is used to penalize noisy pixel values. In other words, smooth estimates are preferred. One cost function is the pixel size. Because we drop the image resolution for the larger block sizes, the pixels will be larger. This results in different pixel sizes between block sizes and the cost function encodes a preference of smaller block sizes over larger blocks. The non pixel based cost functions were the correlation of the found peak, and the signal to noise ratio of the peak. They were converted to the pixel scale by constant multipliers, which were experimentaly determined. This ratio is calculated by dividing the correlation peak height with the standard deviation. This ensures that high correlations that clearly stand out from the surroundings are preferred over a high correlation only, which might result from simply flat colored surfaces.

The algorithm we developed resembles the semi-global block matching (SGM) technique used by Hirschmuller et al. [11]. The key difference between our algorithms is in the cost functions used to quantify the match quality. Hirschmuller et al. use only local standard deviation to penalize bad matches whereas our algorithm employs multiple cost functions to describe the match quality. The iterative structure where results from different block sizes are used also differs. We are using results from multiple levels to ensure that even large surfaces without many distinctive features will get matched correctly. With a small window size, these large surfaces will not be properly matched as the small window does not contain enough texture for finding the match. Using large windows mitigates this problem although it creates new problems with small details being lost on the edges. This is why decision heuristics are implemented to favor good matches from smaller block sizes if it is found on the area of the larger block.

3.3 *Performance*

The algorithm was implemented so that it could be distributed to multiple computers acting as a calculation cluster. The software was running on Matlab and the calculation was distributed through serialized structures sent through ZeroMQ [12], an open source library providing a transport layer for distributed applications.

Average calculation times in a cluster with two Intel i7 3770 processors and Nvidia GTX680 graphics cards was 15-20 minutes for complete processing of one image pair. The current code has not been performance optimized, but it will scale well with the addition of more hardware to calculation. As the algorithm runs iteratively, initial results are available in under one minute with the current setup. For the rest of the time, the estimate is refined to produce a more accurate end result.

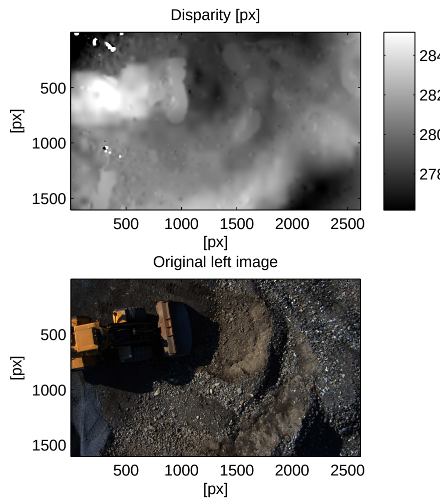


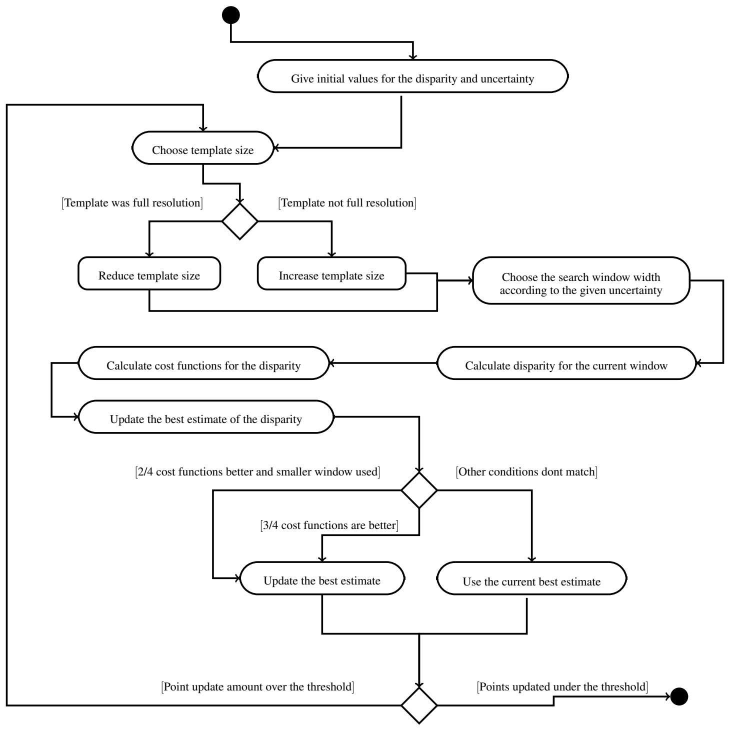
Figure 10: Original image and disparity extracted from the stereo pair in good conditions.

4 Experimental results

In this section, we present some of the best and worst results obtained with our algorithm. Comparing the end results highlights important design aspects that must be taken into account when designing a stereo vision system for use in a MAV. We also present a comparison of our algorithm to the SGM technique of [13]. Figure 10 presents the performance of the disparity calculation in ideal conditions when there is sufficient texture and the triggering of the cameras does not have large time difference. This results in a smooth disparity estimate with only a small amount of missing data due to bad matching.

Figure 11 presents the point cloud extracted from the disparity image seen in Figure 10. Points on the lower left edge are missing due to estimate being too inaccurate. Small errors are also present on the upper right corner where the spike in height is not present in the real terrain. Axes are scaled to 10 m. There is no visible warping on the point cloud caused by the inaccurate triggering of the cameras.

Figure 12 present the system's performance in poor conditions. The matching itself is good but there is a clear trend visible towards the lower left corner. This is a good example of a case where the assumed camera calibration is invalid because cameras were triggered at different times and the position and attitude of the cameras does not correspond to the calibrated positions. This often results in a valid looking disparity image, but the point cloud reveals heavy distortions. Compensating was attempted by forcing the mean disparity to correspond to the height of the MAV but this correction only applies to the incorrect baseline and does not fix the possible



Give initial values for the disparity and uncertainty
Choose template size
[Template was full resolution]
[Template not full resolution]
Reduce template size
Increase template size
Choose the search window width according to the given uncertainty
Calculate cost functions for the disparity
Calculate disparity for the current window
Update the best estimate of the disparity
[2/4 cost functions better and smaller window used]
[Other conditions dont match]
[3/4 cost functions are better]
Update the best estimate
Use the current best estimate
[Point update amount over the threshold]
[Points updated under the threshold]

Figure 9: An activity diagram for the disparity search algorithm. Each of the operations is performed for the whole image.

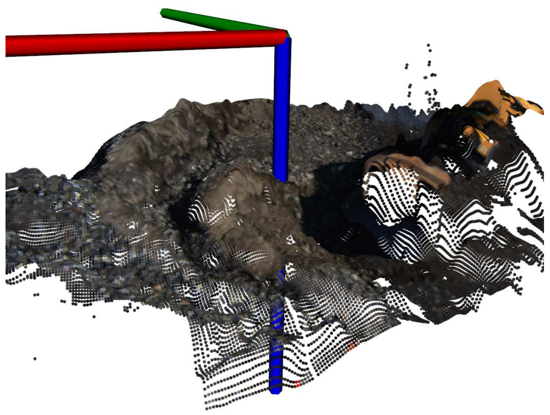


Figure 11: Point cloud extracted from disparity image.

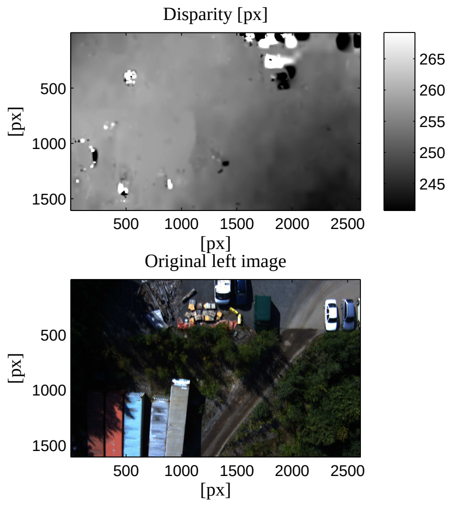


Figure 12: Original image and the disparity extracted from a stereo image pair in poor conditions.

camera rotation between the images. Knowledge of the exact camera triggering times would be required to correct for these errors.

Figure 13 shows the point cloud extracted from the disparity image in Figure 12. The effect of assuming an invalid calibration is clearly visible in this figure, as can be seen from the distorted surface. In reality, the ground plane seen in the figure should be flat.

We tested our algorithms performance against an implementation of SGM [14]. According to the software documentation, the implementation is based on [13]. The comparison is presented in Figure 14 We used the image pair that was earlier presented in Figure 10. We used a block size of 17px, with search window size of $64x304$px with other values left at their default values. It can be seen that our method man-

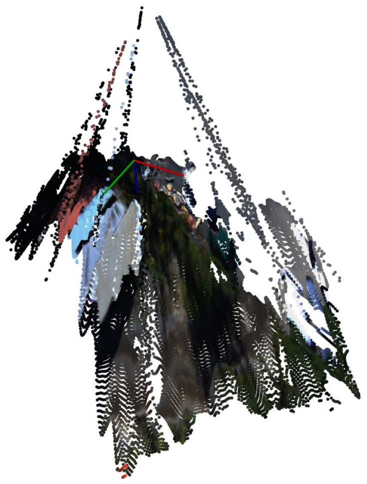


Figure 13: A point cloud extracted from a disparity image when an invalid camera calibration is applied.

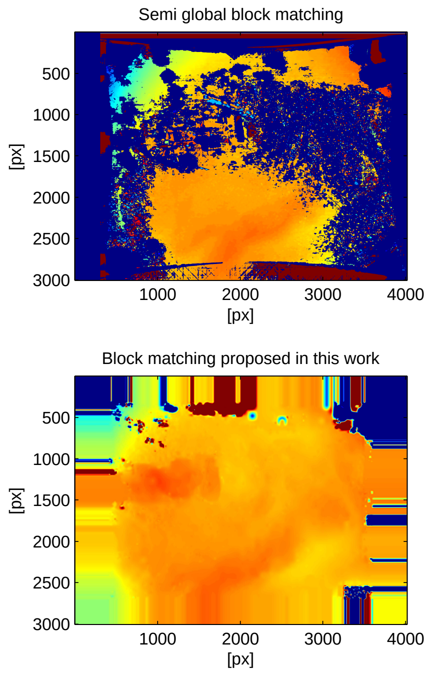


Figure 14: Comparing Matlab implementation of semi global block matching to our algorithm.

ages to find a more complete disparity estimate. However, it is worth noting that SGM took 12seconds to calculate the results on a single PC as our method took roughly 15 minutes running on two processors and one GPU. The result indicates that there may be cases where our algorithm outperforms SGM. However, a fair comparison requires testing with more input images and is a topic for future work.

5 Conclusion

We applied high-resolution, low-cost consumer grade cameras to build a lightweight stereo camera system mounted on a MAV. Specialized algorithms were developed to extract depth information from the high-resolution images. We were able to extract high quality depth data from low textured surfaces like dirt roads, sand piles and grass. High resolution stereo vision was useful in identifying features and making the system compact without making the field of view too narrow.

In flight altitudes exceeding 10m, the quality of the point clouds decreased. This was due to the features being too small. This can be seen as increasingly noisy depth estimates for low featured surfaces. Another drawback was caused by the use of the consumer grade cameras which did not trigger accurately at the same time. This resulted in the stereo calibration assumed in the image processing phase being inaccurate. Inaccuracy in camera calibration was often not critical when the altitude was low as the disparities were small, but when altitude was more than 10m, the results degraded heavily. We identified several important factors that should be taken into account when designing a stereo camera system for use in a MAV. The single most important factor is to trigger the cameras at precisely the same time. Even differences as small as 100ms can cause significant errors when the cameras are moving at typical MAV speeds. Using multiple block sizes proved to be useful while matching images with low texture. High computation times were expected as the goal of this work was not to make a real time stereo vision system, but to explore the possibilities of the high resolution stereo vision. This work is a summary of one of the authors Masters thesis [15]. The algorithm development and data gathering was done during year 2013. In future work, we will concentrate on using machine vision cameras with accurate hardware-level triggering. We are also planning to test how low textured surfaces can be matched using low resolution cameras with a narrow field of view. In addition, computation is faster for lower resolution images. Smaller cameras also enable us to use a longer baseline to compensate for the smaller resolution.

Acknowledgements

The authors would like to thank Mr. Heimo Ihalainen for discussions and valuable insights related to this work.

References

[1] H. Sunyoto, W. Van der Mark, and D.M. Gavrila. A comparative study of fast dense stereo vision algorithms. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 319–324, June 2004.

[2] K. Schmid and H. Hirschmuller. Stereo vision and imu based real-time ego-motion and depth image computation on a handheld device. In *Robotics and Automation*

(ICRA), 2013 IEEE International Conference on, pages 4671–4678, May 2013.

[3] K. Schmid, T. Tomic, F. Ruess, H. Hirschmuller, and M. Suppa. Stereo vision based indoor/outdoor navigation for flying robots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3955–3962, Nov 2013.

[4] Arducopter project. Arduino-based autopilot for mulitrotor craft. Available online: http://copter.ardupilot.com/, 2014.

[5] T. Takasu. RTKLIB: An open source program package for gnss positioning. Available online: http://www.rtklib.com/, 2014.

[6] R. Hellevaara. RTK-GPS on a UAV Platform: Determining Coordinates from an Image (in Finnish) . Master's thesis, Tampere University of Technology, Finland, 2013.

[7] M. Warren and B. Upcroft. High altitude stereo visual odometry. In *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[8] O. Faugeras. *Three-dimensional computer vision : a geometric viewpoint*. MIT Press, Cambridge, Mass, 1993.

[9] Peter I. Corke. *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011.

[10] C. Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006.

[11] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):328–341, 2008.

[12] iMatix Corporation. ZeroMQ intelligent transport layer for distributed apps. Available online: http://zeromq.org/, 2014.

[13] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814. IEEE, 2005.

[14] MATLAB. *version 8.3 (R2014a)*. The MathWorks Inc., Natick, Massachusetts, 2014.

[15] J. Melin. Surface reconstruction using high resolution stereo vision in a micro air vehicle. Master's thesis, Tampere University of Technology, Finland, 2014.