Realtime stereo matching using maximally stable extremal regions

Julius Gelšvartas[1,2] (corresponding author), Henrikas Simonavičius[2], Andrius Lauraitis[3], Rytis Maskeliūnas[3], Piotras Cimmperman[4], Paulius Serafinavičius[4]

[1]Automation Department, Faculty of Electrical and Electronics Engineering, Kaunas University of Technology, Studentų g. 50-154, Kaunas, Lithuania

[2]Rubedo sistemos, K. Baršausko g. 59a, Kaunas, Lithuania

[3]Department of Multimedia Engineering, Faculty of Informatics, Kaunas University of Technology, Studentų g. 50–414a, Kaunas, Lithuania

[4]Baltic Institute of Advanced Technology, Saulėtekio al. 15, Vilnius, Lithuania

julius.gelsvartas@gmail.com, henrikas.simonavicius@rubedo.lt, andrius.lauraitis@ktu.lt, rytis.maskeliunas@gmail.com, piotras.cimmperman@bpti.lt, paulius@serafinavicius.lt

Abstract

In this paper we present a novel stereo matching algorithm *Cyclops*. The algorithm uses Maximally stable extremal regions (MSER) for localizing parts of the image in rectified stereo image pair. Each detected MSER is normalized to a fixed size image. The normalized MSER regions are matched in the stereo image pair taking into account the epipolar line constraints. Matched MSER pairs are used to calculate the disparity image using the semi-global matching technique. *Cyclops* algorithm reduces stereo matching computation time and memory usage. The proposed algorithm is therefore useful for real time stereo vision.

Keywords: stereo vision, computer vision, image processing, maximally stable extremal regions.

1. Introduction

Unmanned aerial vehicles (UAV) require reliable sensors in order to operate autonomously. Current UAVs can already perform autonomous way-point following reliably. Obstacle avoidance, on the other hand, is still considered a challenge. UAV has to be capable of sensing its 3D environment to avoid obstacles. An ideal sensor should be very light, consume as little energy as possible, be usable outdoors and have high frame rate. Currently there are four types of sensors that are suitable for obstacle avoidance, namely lidar, structured-light, time of flight (TOF) and stereo cameras.

Structured-light sensors satisfy almost all requirements. These sensors have successfully been used for obstacle avoidance in indoor environments (Stowers, Hayes, & Bainbridge-Smith, 2011). Structured-light sensors project known patterns onto the environment in order to estimate scene depth information. This approach, however, has several limitations. First, the projected pattern becomes undetectable in direct sunlight. Second, these sensors have a limited detection range of up to 5 meters.

TOF cameras, on the other hand, are less affected by direct sunlight because they use modulated light. Some TOF cameras have longer range than structured-light cameras but at a cost of increased power consumption. Another limitation of TOF cameras is low resolution of currently available sensors. The main advantage of TOF cameras is their very high frame rate (usually more than 100Hz).

3D lidar sensors are currently the most widely used in outdoor environments. Lidar sensors offer very long range of 100m or more and accurate distance measurements. These sensors, however, are still very expensive.

All of discussed sensors are active, i.e. they are emitting light rays in order to determine the distance to the obstacles. The biggest disadvantage of active sensors is power consumption. Actively emitting light requires a lot of energy. Stereo camera, on the other hand, is a passive distance sensing technology. Using passive environment sensing has several advantages. First, this type of sensor can better adapt to lighting changes in the environment, because we can control their exposure time and gain parameters. Second, such system consumes less energy than active sensing.

Processing a pair of stereo images is a very computationally intensive procedure. Therefore, stereo camera images are usually processed on powerful computers that consume a lot of energy. In recent years latest embedded and mobile processors have become powerful enough to process stereo images. This paper investigates the potential of using NVIDIA's Jetson TK1 (NVIDIA, 2016) embedded platform for processing stereo images. A novel stereo matching algorithm that uses maximally stable extremal regions (MSER) (Matas, Chum, Urban, & Pajdla, 2004) is proposed. The proposed algorithm is compared with several state of the art stereo matching algorithms using the Middlebury Stereo Dataset (Scharstein, et al., 2014).

Stereo matching algorithms can be divided into two categories namely, local and global. Global algorithms can reconstruct the disparity images very accurately (Zhang, et al., 2015) but they are computationally intensive and have high memory requirements. Even highly optimized versions of these methods have long computation times and are not suitable for real time applications. These methods are not considered in this paper.

Local methods can process images faster and have much higher frame rates. One example is semi-global block matching (SGBM) algorithm based on work done by (Hirschmüller, 2008). This algorithm has become very popular because it achieves good reconstruction accuracy and reasonable computation time. It has been demonstrated (Žbontar & LeCun, 2015) that semi-global matching algorithm can achieve very high reconstruction accuracy when combined with convolutional neural networks. The fast version of this algorithm achieves high frame rates. Achieving these results however requires a very powerful GPU card that has 16 times more GPU cores than NVIDIA’s Jetson TK1 processor. It has been shown that

high stereo matching frame rates can be achieved using powerful GPU processors (Kowalczuk, Psota, & Perez, 2013) and (Kim, Kang, & Kim, 2016). These algorithms would have significantly lower performance on embedded platforms such as NVIDIA's Jetson TK1 or they might not be portable at all due to high memory consumption.
Efficient large-scale stereo matching (ELAS) is another algorithm that has high frame rates (Geiger, Roser, & Urtasun, 2010). This algorithm creates a dense disparity prior image by triangulating robustly matched support points. Having a disparity prior allows to speed-up the computation of the final dense disparity image. This algorithm is very sensitive to incorrectly matched support points. Our proposed algorithm was inspired by ELAS but instead of matching support points we are using MSER matches for creating a disparity prior.
MSERs have already been successfully used in wide-baseline stereo matching setup (Matas, Chum, Urban, & Pajdla, 2004). Matching MSERs in this case is difficult because the epipolar geometry is unknown. Stereo camera images can be rectified therefore MSERs have to only be matched along the epipolar line. Another advantage of MSERs is that they can be detected very quickly on greyscale (Nistér & Stewénius, 2008) as well as colour images (Forssén, 2007).
Both SGBM and ELAS algorithms have good computation times for small resolution images. The computation time of these algorithms increases drastically with increasing image resolutions. Another limitation of SGBM and ELAS algorithms is their high memory usage. This becomes an issue when trying to run these algorithms on embedded devices. High memory usage can also complicate algorithm parallelization on GPU processors since they usually have limited memory capacity. *Cyclops* aims to address both computation time and memory consumption issues of SGBM and ELAS.
The aim of this paper is to present and evaluate a stereo matching algorithm *Cyclops*. The remaining paper is structured as follows. Section 2 describes the MSER based stereo matching algorithm and the experimental setup. The results are provided in Section 3. Discussion and conclusion are in Section 4 and Section 5 respectively.

2. Materials and methods

Cyclops takes as an input a pair of rectified stereo images and produces a disparity image. The algorithm consists of the following steps:

1. Extract MSERs in each stereo image.
2. Normalize each MSER region so that it could easily be matched.
3. Match MSER regions of left and right images.
4. Calculate the disparity image.

Each step is described in more details in the following sections.

2.1 Extracting MSERs

Each stereo image is processed independently to extract MSERs. MSER extraction algorithm available in OpenCV library (Bradski & Kaehler, 2008) was used. This algorithm can handle both greyscale and colour images. Colour image processing algorithm is slower but MSERs matching is less ambiguous in this case. Greyscale images can be processed faster but matching becomes more difficult. The choice of colour or greyscale image is a trade-off between accuracy and speed. Usually MSERs are detected on regular intensity images (MSER+). Additionally, MSERs can also be detected on the inverted image (MSER-). Performing MSER- detection produces a large number of additional regions for matching step. MSER- detection requires inverted image calculation and doubles the image processing time.
The main disadvantage of MSERs is that their distribution over the image might be sparse. Additionally, large untextured areas of the image might not be detected as MSER. This is

because MSER detection algorithms usually have maximal MSER size restrictions. Even when large MSERs are detected their matching can be complicated due to occlusions. This can be solved by dividing each image into horizontal sub-images such that

$$I_i = I([0{:}w), [i * h_{sub}{:}(i+1) * h_{sub}))$$

here I_i is the i-th sub-image extracted from original image I, w is the original image width and h_{sub} is the height of each sub-image. The image is divided into horizontal sub-images because the stereo images are rectified such that epipolar lines are aligned with the horizontal axes of the two images. Each sub-image is then processed individually using the same algorithm steps. Dividing image has two advantages. First, sub-images split very large MSER regions that are much less likely to be matchable due to occlusions. Second, since each sub-image is processed independently the number of matching candidates is reduced and matching step is faster. Image dividing procedure could further be improved by creating overlapping sub-images This would reduce depth discontinuities on image borders and splitting of small MSERs. An example image with detected MSERs can be seen in Figure 1.

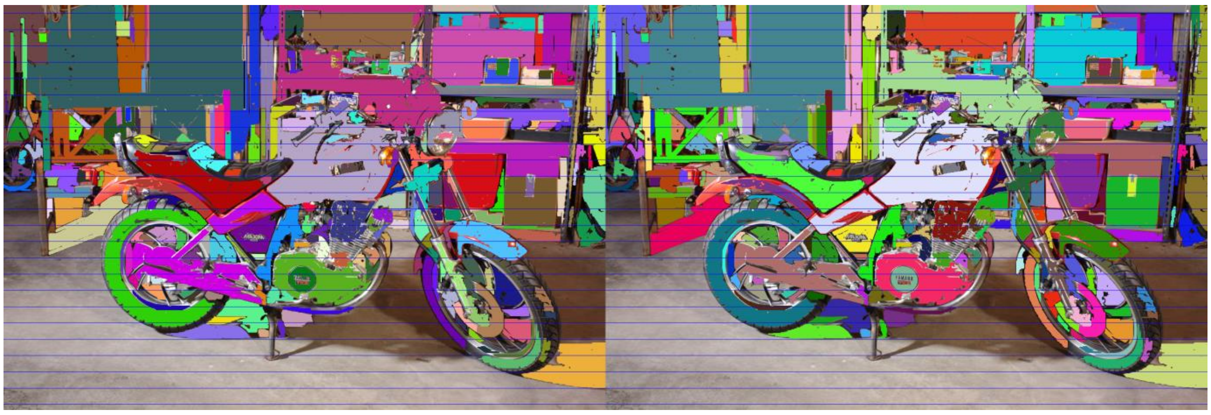
Figure 1. Stereo image pair with detected MSER coloured randomly. Blue lines show epipolar image lines every 20 pixels

2.2 Normalizing MSERs

Detected MSERs regions have varying sizes and cannot be directly compared and matched. To overcome this problem, the MSERs have to be normalized before matching. In the wide baseline stereo scenario MSERs are normalized by transforming the region boundary so that the covariance matrix of the region becomes diagonal (Matas, Chum, Urban, & Pajdla, 2004). Wide baseline stereo images are not rectified and the same region can be arbitrarily rotated in two images. Each normalized region is, therefore, used to extract rotational invariant (based on complex moments) descriptor. Region descriptors can then be matched. The main disadvantage of this approach is that it uses region boundary information but region internal pixel information is discarded.

Our method uses rectified stereo images. In this case an MSER region detected in left stereo image is typically going to have similar appearance in right stereo image. Normalization can be performed by scaling each MSER image into a fixed size image. More formally, the normalization input is a rectified image I and a list of detected MSER in that image. Each MSER is represented by a list of image coordinates that belong to that region $(p_1, p_2, \ldots, p_n)$ and a bounding box of that region b. Each MSER region is normalized independently. The normalization algorithm works as follows:

1. Temporary empty image I_l is created. This image has the same width and height as b.

2. All the MSER points $(p_1, p_2, \dots, p_n)$ are copied from I to I_l. All points in I_l that do not belong to the MSER region are set to black.
3. The image I_l is resized to a fixed size image I_f. Image I_f is the normalized MSER image that is used in MSER matching.

This normalization process is applied to both left and right stereo image MSERs. The normalization procedure is illustrated in Figure 2. The normalized regions can be easily compared and matched.

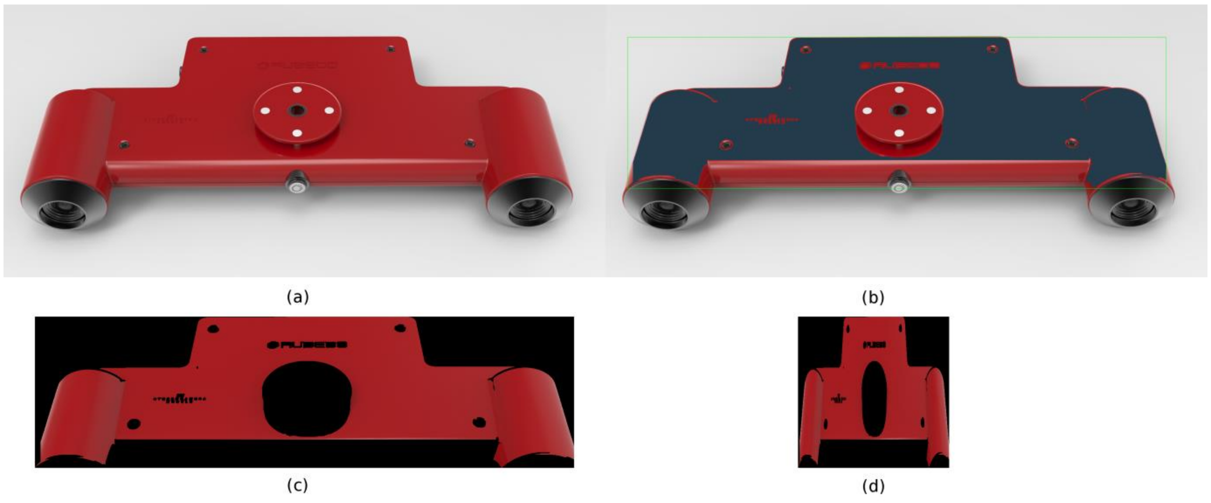

Figure 2. MSER region normalization procedure. (a) sample image $\boldsymbol{I}$, (b) detected sample MSER region (blue) and its bounding box $\boldsymbol{b}$ (green), (c) temporary MSER image $\boldsymbol{I_l}$, (d) normalized MSER image $\boldsymbol{I_f}$.

2.3 MSER matching

The matching is performed between two sets of MSER regions, namely $MSER_l = \{mser_{l1}, mser_{l2}, \dots, mser_{ln}\}$ and $MSER_r = \{mser_{r1}, mser_{r2}, \dots, mser_{rm}\}$. $MSER_l$ are all regions detected in left camera image and $MSER_r$ are regions from right camera image. Each region $mser_{li}$ is compared to all the regions in $MSER_r$. Majority of regions can be discarded as impossible matches by performing simple checks. We are performing the following checks to discard impossible matches:

1. We estimate the disparity of two regions by calculating the difference of the centre points of the MSER bounding boxes. Matching candidates whose estimated disparity is smaller than minimal disparity of larger than maximal disparity are discarded. The maximal and minimal disparity values are algorithm parameters.
2. Given two regions $mser_{li}$, $mser_{rj}$ and their bounding boxes b_{li}, b_{rj} we check that their bounding boxes intersect along the Oy axis (Ox axis is along image columns and Oy axis along image rows). The stereo images are rectified, therefore the bounding boxes of the matching regions must intersect.
3. The intersection of the bounding boxes b_{li} and b_{rj} along the Oy axis should be higher than a defined threshold.

MSER regions that pass all of the above checks are used for region similarity checks. Region similarity is measured between normalized $mser_{li}$ and $mser_{rj}$ images using the sum of absolute differences (SAD) metrics. For each region in $MSER_l$ we select one matching region from $MSER_r$ that has a minimal SAD value.

This approach can introduce false matches when a particular region has no similar regions. To reduce the amount of false matches the maximal SAD threshold is used. A check is performed to make sure that matching region SAD does not exceed maximal SAD threshold. Introducing a fixed maximal SAD threshold does not fully solve the problem. This is because SAD values are dependent on what area of MSER image is covered with region points. Regions that have square shapes will cover more of the MSER image and will have larger SAD values. Elongated regions on the other hand will have more black pixels and smaller SAD values. This is taken into account by scaling maximal SAD value by the percentage of MSER points in MSER image.

2.4 Calculating disparity image

The disparity image is calculated in the left image optical frame. Each pair of matched MSER regions $mser_{li}$ and $mser_{rj}$ is used to estimate the disparity value of the pixels belonging to $mser_{li}$. This paper proposes to use SGBM algorithm to estimate disparity values inside of the matched regions.

Matched MSER regions are well localized and SGBM algorithm only has to be applied within a very small disparity range. The input for the SGBM algorithm is obtained by extracting image regions of interest (ROI) from the original left and right stereo images. The ROI are defined by the MSER region bounding boxes b_{li} and b_{rj}. Disparity values are calculated for all the pixels in the ROI image creating disparity ROI image. Only disparity values that belong to the $mser_{li}$ are copied from disparity ROI image into the final disparity image.

2.5 Experimental setup

All experiments were performed on a NVIDIA’s Jetson TK1 development board. The evaluated algorithms only use CPU computations and the GPU processor was not used. The performance of the CPU cores was maximized by disabling the CPU power saving functionality. The board was powered using the default power adaptor provided in the development kit. The Linux for Tegra R21.4 operating system provided by NVIDIA was used. During the experiments only the disparity evaluation algorithm process was running. Each experiment was started via a ssh connection to the board.

Cyclops was compared with two high frame rate algorithms, namely SGBM and ELAS. These algorithms are at the top of the Middlebury Stereo Evaluation when evaluated according to the algorithm computation time. Both of these algorithms have publicly available implementation source codes. We used the ELAS algorithm implementation provided on the authors page and the SGBM algorithm implementation provided with OpenCV library version 3.1. The algorithms were evaluated using the tools and images provided by the Middlebury Stereo Dataset version 3. Evaluation was performed only on 15 training images from Middlebury Dataset. Both ELAS and SGBM algorithms were configured to have the same parameters as those used in Middlebury Stereo Dataset. The SGBM algorithm was configured to use 'Full DP' parameter set to false, this runs the faster version on SGBM algorithm. 'Full DP' parameter enables full-scale two-pass dynamic programming algorithm that consumes more memory and is slower.

Cyclops first extracted MSER regions on sub-images whose width was set to $h_{sub} = 50$ pixels (px). Each extracted MSER region was normalized to I_f image whose size was fixed at $50px \times 50px$. The region intersection threshold used during matching step was set to 0.9. Finally the maximal SAD threshold was set to 120000.

The algorithms were evaluated using the following criteria:

1. Algorithm computation time. This was measured without taking into account the time taken to load input images and save the result disparity image. The computation time was calculated for each image individually. The results section provides the average algorithm computation time per image in seconds.
2. Total memory usage of the process that was executing stereo matching algorithm. This experiment was performed on Vintage stereo image pair from Middlebury Stereo Dataset. The measurements were obtained using the Massif tool from Valgrind profiler available in Ubuntu distribution.
3. Percentage of bad pixels whose disparity error is greater than 2.0. The provided measure is average percentage of bad pixels per image.
4. Percentage of invalid pixels per image. These are pixels where the stereo matching algorithm was not able to estimate the disparity value. The provided measure is average percentage of invalid pixels per image.
5. Average error of bad pixels per image. The average disparity value error of pixels that have been classified as bad pixels.

The algorithms were evaluated using three different resolution image sets, namely full, half and quarter resolution images. The full resolution images have the size of up to 3000 × 2000 pixels and maximal disparity values of up to 800 pixels. Middlebury dataset provides the maximal disparity values of each image individually. Half resolution images are two times smaller with up to 1500 × 1000 pixels and no more than 400 disparity values. The quarter resolution images are again two times smaller than half resolution images. In the provided results the used resolution is indicated by adding an underscore and a letter to the algorithms name. Letter Q indicates quarter resolution images, H half resolution images and F full resolution images. For example, SGBM algorithm that was run using the half resolution images would be shown as SGBM_H.

3. Results

The most important criteria applied to evaluate the algorithms is the algorithm computation time. *Cyclops* algorithm reduces the search space and is therefore expected to reduce the computation time. The computation time results can be seen in Figure 3.

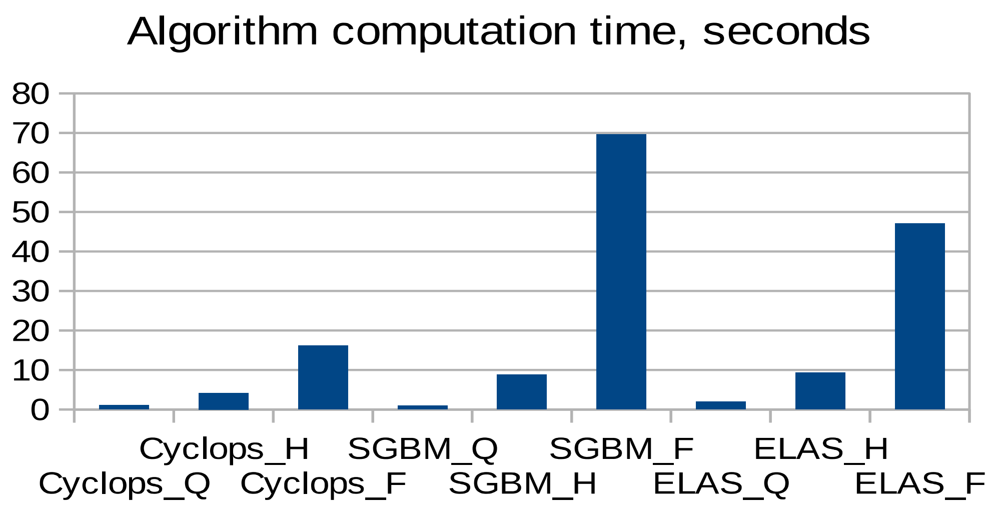

Figure 3. The average algorithm computation time per image, seconds

All algorithms processed quarter resolution images in very similar computation times. The computation time of the SGBM and ELAS algorithms rapidly increases to almost 10 seconds for H images and more than 40 seconds for F images. The computation time of *Cyclops* algorithm increases a lot slower. *Cyclops* algorithm is approximately two times faster than SGBM and ELAS for half resolution images. For full resolution images the speed up is even more significant, approximately three times faster than ELAS and four times faster than SGBM algorithm.
The percentages of time spend on algorithm steps are shown in Table 1. *Cyclops* spends more than 50% of computation time detecting MSER regions and more than 40% computing disparity image values. This experiment was performed on F resolution Vintage stereo image pair. Other image pairs and resolutions had similar results. MSER regions can be detected in linear time (Nistér & Stewénius, 2008). This is the main reason why *Cyclops* computation time increases slower compared to ELAS and SGBM.

Table 1. Percentage of time spend on algorithms steps

Algorithm step	Percentages of total time spent, %
MSER detection	50.18
Normalization	5.18
Matching	0.81
Disparity computation	43.83

Cyclops algorithm not only has a smaller algorithm computation time but also reduces the overall memory usage of the algorithm. This was tested on Vintage stereo image pair images. The results of algorithm memory consumption are shown in Figure 4.

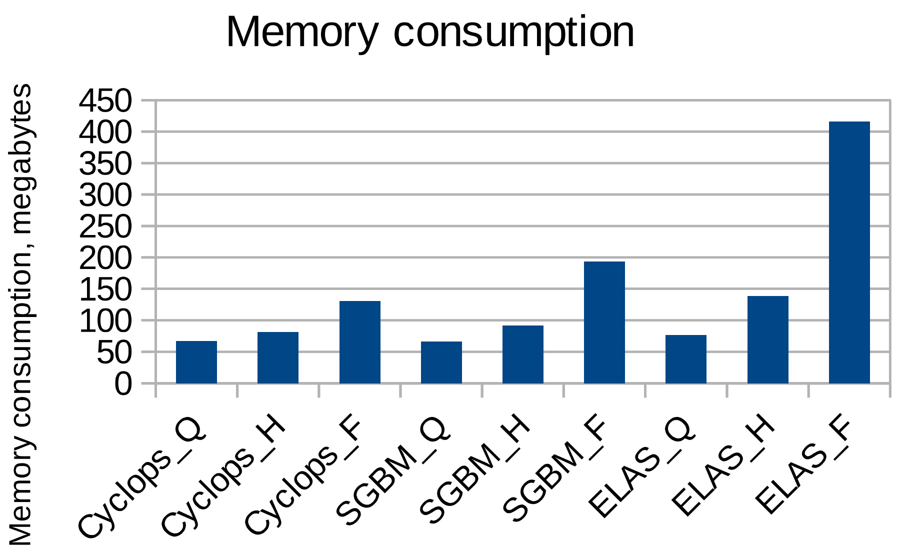

Figure 4. Algorithm total memory consumption, megabytes

Overall *Cyclops* algorithm used less memory to process H and F resolution images. Overall ELAS algorithm had more than 40% higher memory consumption. All algorithms used almost the same amount of memory for Q resolution images. On H resolution images *Cyclops* uses less memory than both SGBM and ELAS. This reduction in memory usage is even more significant on F resolution images, namely 68.6% less memory than ELAS and 32.3% less memory than SGBM. Detailed results of computation time and memory consumption are provided in Table 2.

Table 2. Algorithm computation time (seconds) and memory consumption (megabytes)

Algorithm name	Time, seconds	Memory consumption, megabytes
*Cyclops*_Q	1.08	67.17
*Cyclops*_H	4.2	81.66
*Cyclops*_F	16.22	130.68
SGBM_Q	1.01	66.36
SGBM_H	8.82	91.73
SGBM_F	69.67	193.07
ELAS_Q	1.99	76.51
ELAS_H	9.37	138.27
ELAS_F	47.04	416.02

Next the average percentage of bad and invalid pixels in the produced disparity images is evaluated. The results of this experiment are shown in Figure 5.

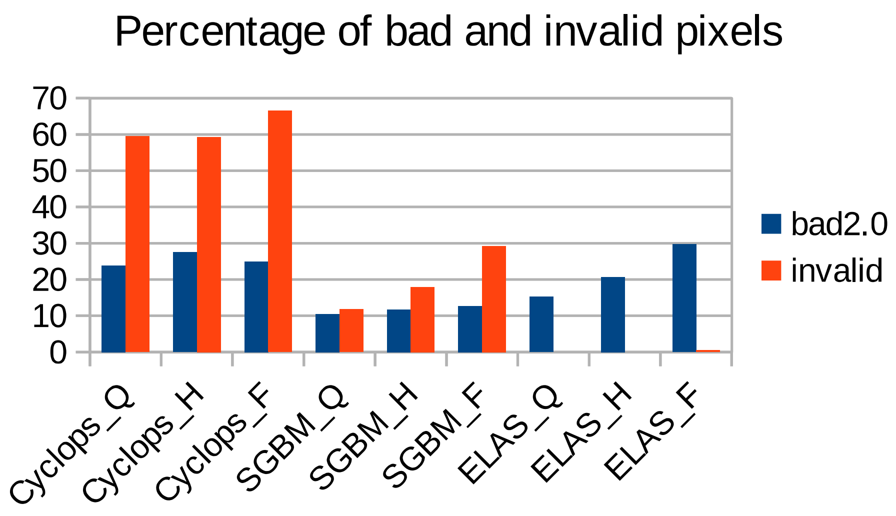

Figure 5. Average percentage of bad pixels whose error is larger than 2.0 (blue) and average percentage of invalid pixels (red)

The SGBM algorithm had similar percentage of bad pixels across all three resolution images. The percentage of invalid pixels increased approximately by 6% and 11% with increasing resolution. The main reason is probably the fact that the algorithm parameters were constant across different resolution images. Changing parameters for different resolution images might change the results but results for these algorithms in Middlebury dataset uses constant algorithm parameters.

The ELAS algorithm contained no invalid pixels because the default algorithm parameters for the Middlebury dataset performed disparity image hole filling. There was one exception where full resolution image contained a single invalid pixel region that was not filled by the algorithm. The cause of this inconsistency was not investigated further. The percentage of bad pixels of ELAS algorithm was slightly larger (4.7%) than SGBM and increased by 5% and 9% for higher resolution images.

Cyclops algorithm has high percentage of invalid pixels. There are two main reasons for this. First, the detected MSER regions do not cover large untextured regions. This problem was partially addressed by dividing the image into sub-images. Second, no post processing or hole filling algorithms were used to cover invalid pixel areas. Adding such post processing algorithms would lower or eliminate invalid pixels but could also potentially increase the percentage of bad pixels.

The percentage of bad pixels of *Cyclops* algorithm was higher than SGBM and ELAS algorithm for quarter and half size images. This metric was not changing significantly for different resolution images.

Another important metric for evaluating stereo matching algorithm quality is the average disparity error of bad pixels. The results of this evaluation are provided in Figure 6.

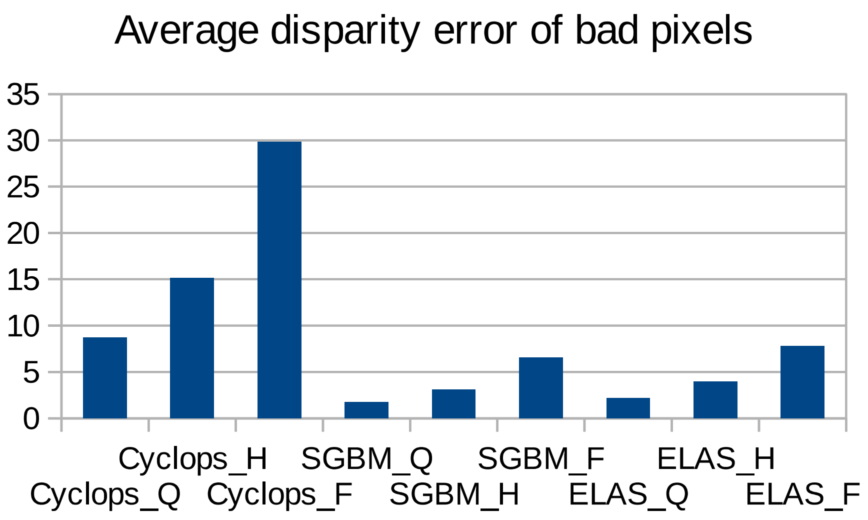

Figure 6. The average disparity error of bad pixels

Both SGBM and ELAS algorithm had a very similar disparity error values. All algorithms had an increasing disparity error when the image resolution increased. The average disparity error of the proposed algorithm was higher than SGBM and ELAS. The main source of error in *Cyclops* algorithm was the SGBM algorithm that was used to estimate the final disparity values. The main cause of this is that only the bounding boxes of the matched MSER regions are provided for SGBM algorithm. The MSER regions usually do not contain enough edges or texture that improves SGBM matching. This issue can be addressed by providing SGBM algorithm with regions that are larger than the MSER bounding boxes. Another potential solution is to use more accurate stereo matching algorithm in the final algorithm stage. The main purpose of *Cyclops* algorithm is to reduce the search space of the algorithm used in final disparity evaluation stage. Reduced search space makes it possible to use more advanced algorithms in the final disparity evaluation stage and have high overall algorithm computation frame rates. The more in detail results are provided in Table 3.

Table 3. Percentage of bad pixels whose error is larger than 2.0, percentage of invalid pixels and average disparity error of bad pixels

Algorithm name	Bad 2.0, %	Invalid, %	Average disparity error, px
Cyclops_Q	23.87	59.45	8.69
Cyclops_H	27.46	59.31	15.16
Cyclops_F	24.98	66.55	29.81
SGBM_Q	10.54	11.81	1.72
SGBM_H	11.69	17.91	3.09
SGBM_F	12.67	29.19	6.52
ELAS_Q	15.26	0	2.18
ELAS_H	20.70	0	3.94
ELAS_F	29.70	0.52	7.77

4. Discussion

This paper presents a novel rectified stereo image matching algorithm *Cyclops*. MSERs are used to quickly match regions in stereo image pair. The main advantage of matching regions is that significantly less computations have to be performed. This reduces the algorithm computation time and memory usage. The reduced computation time and memory usage is most significantly visible when processing high resolution images. The proposed algorithm is able to process F resolution images in 16.22 seconds using approximately 130 megabytes of memory on an embedded computer.

Further improvements are necessary to address *Cyclops* limitations. First, the reconstructed disparity image is usually sparse. This can either be solved by post processing the computed disparity image or by choosing other region detectors. These solutions would significantly reduce the percentage of invalid pixels. Second, the produced disparity image usually has significant disparity errors that can be higher than $8px$. This could be addressed by using a more accurate disparity estimator in the final algorithm step. Using a more accurate algorithm would not significantly increase the overall algorithm computation time. This is mainly because the image regions that are used in the final algorithm stage are well localized and should only have small disparity value changes within the region.

It is worth noting that currently the algorithm is executed sequentially. Parallelizing the algorithm could further improve its computation times. Parallelized algorithm could achieve 2 to 10 times speed up. The most trivial parallelization technique would only require computing each sub-image in parallel. The current algorithm splits the image into sub-images and processes each sub-image independently but sequentially. A more advanced approach would require parallelizing individual algorithm steps.

Parallelized version of this algorithm could be used for obstacle avoidance. The NVIDIA's Jetson TK1 would be an ideal platform for creating a stereo vision based obstacle avoidance system. TK1 has a small form factor as well as low power consumption. High levels of invalid pixels would not have very big impact on obstacle avoidance as long as the algorithm is able to detect foreground obstacles. Disparity errors would pose more problems for obstacle avoidance but this can be solved by choosing a more accurate disparity computation algorithm.

5. Conclusion

This paper shows that *Cyclops* algorithm can achieve fast computation times on high resolution images. Reducing memory consumption is also important because free memory can be used by other algorithms such as obstacle detection.

Acknowledgements

This research was funded by a grant (No. REP-1/2015) from the Research Council of Lithuania.

References

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library.* O'Reilly Media, Inc.

Forssén, P.-E. (2007). Maximally stable colour regions for recognition and matching. *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on* (pp. 1-8). IEEE.

Geiger, A., Roser, M., & Urtasun, R. (2010). Efficient Large-Scale Stereo Matching. *Asian Conference on Computer Vision (ACCV).*

Hirschmüller, H. (2008). Stereo processing by semiglobal matching and mutual information. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 30*(2), 328-341.

Kim, S., Kang, S.-J., & Kim, Y. (2016). Real-time stereo matching using extended binary weighted aggregation. *Digital Signal Processing*, 51-61.

Kowalczuk, J., Psota, E., & Perez, L. (2013). Real-time stereo matching on CUDA using an iterative refinement method for adaptive support-weight correspondences. *IEEE transactions on circuits and systems for video technology, 23*(1), 94-104.

Matas, J., Chum, O., Urban, M., & Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing, 22*(10), 761-767.

Nistér, D., & Stewénius, H. (2008). Linear time maximally stable extremal regions. *European Conference on Computer Vision* (pp. 183-196). Springer.

NVIDIA. (2016, 04 04). *Jetson-TK1 Embedded Development Platform*. Retrieved from http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html

Sanghun, K., Suk-Ju, K., & Young Hwan, K. (2016). Real-time stereo matching using extended binary weighted aggregation. *Digital Signal Processing, 53*, 51-61.

Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nešić, N., Wang, X., & Westling, P. (2014). High-resolution stereo datasets with subpixel-accurate ground truth. *Pattern Recognition* (pp. 31-42). Springer.

Stowers, J., Hayes, M., & Bainbridge-Smith, A. (2011). Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor. *Mechatronics (ICM), 2011 IEEE International Conference on* (pp. 358-362). IEEE.

Žbontar, J., & LeCun, Y. (2015). Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. *arXiv preprint arXiv:1510.05970*.

Zhang, C., Li, Z., Cheng, Y., Cai, R., Chao, H., & Rui, Y. (2015). Meshstereo: A global stereo model with mesh alignment regularization for view interpolation. *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2057--2065). IEEE.