Article Identifier
DOI: 10.22032/dbt.58859
URN: urn:nbn:de:gbv:ilm1-2023isc-075:8

60th ILMENAU SCIENTIFIC COLLOQUIUM
Technische Universität Ilmenau, 4 – 8 September 2023
URN: urn:nbn:de:gbv:ilm1-2023isc:1

Suitability study for real-time depth map generation using stereo matchers in OpenCV and Python

M. Richter[1], M. Rosenberger[1], R. Illmann[1], D. Buchanan[2], G. Notni[1,3]

Technische Universität Ilmenau1
Vision & Control GmbH[2]
Fraunhofer Institute for Applied Optics and Precision Engineering[3]

ABSTRACT

Stereo imaging provides an easy and cost-effective method to measure 3D surfaces, especially due to the availability of extensive free program libraries like OpenCV. An extension of the application to the field of forestry was aimed at here in the context of a project to capture the elevation profile of forest roads by means of stereo imaging. For this purpose, an analysis of the methods contained in OpenCV for the successful generation of depth maps was carried out. The program sections comprised the reading of the image stream, the image correction on the basis of calibrations carried out in advance as well as the generation of the disparity maps by the stereo matchers. These are then converted back into depth maps and stored in suitable memory formats. A data set of the image size 1280x864 pixels consisting of 30 stereo image pairs was used. The aim was to design an evaluation program which allows the processing of the described steps within one second for 30 image pairs. With a sequential processing of all steps under the used test system and the usage of a local stereo matcher a processing time of 4.37 s was determined. Steps to reduce the processing time included parallelizing the image preparation of the two frames of the image pair. Further reduction in total processing time was achieved by processing multiple image pairs simultaneously and using storage formats without compression. A total processing time of 0.8 s could be achieved by outsourcing the stereo matching to the graphics card. However, the tested method did not achieve the desired resolutions in depth as well as in the image plane. This was made possible by using semi-global matchers, which are up to 10 times slower but significantly more accurate, and which were therefore used for further investigations of the forest path profile.

Index Terms - OpenCV, Stereo Imaging, Python

1. INTRODUCTION

Early and targeted detection of road distress plays a crucial role in reducing costs and minimizing repair time. It also contributes to the enhancement of road safety by enabling timely prevention of further deterioration. The majority of studies and commercially utilized technologies focus on the assessment of paved roads used by thousands every day. A sensor system therefore not only has to be highly accurate to detect fine cracks but also fast enough to not obstruct the flowing traffic. This combination necessitates the utilization of rapid and costly measurement setups such as laser scanners [1], limiting the number of measurement vehicles. However, the extension of these measurement techniques to unpaved roads commonly found in forest environments is currently lacking in practice. Nonetheless, the evaluation and

maintenance of the forest road network are of utmost importance, particularly for efficient lumber removal, where heavy-loaded trucks must be able to traverse the roads safely.

The main causes of pavement damage on forest roads are primarily ruts, which account for about 67 % of the total damage, followed by general erosion of the surface layer with 24 % and potholes with 3 % [2]. These types of damage characteristically have a greater width compared to their depth. On asphalt roads however, cracks characterized by a high depth to surface area ratio are the most common type of damage, accounting for about 68% [1,3]. In contrast potholes and ruts account for less than 4% of the damage types [3].

Although it is possible to apply the same technologies used for paved roads to unpaved forest roads, the absence of high traffic volume, lower speeds, and distinct distress characteristics in contrast to paved roads present an opportunity to utilize slower yet more cost-effective methods, such as stereoscopic 3D. Additionally, comprehensive and freely available software packages like OpenCV can be utilized to analyze and extract three-dimensional data from the captured two-dimensional image pairs.

To develop a stereo system for this purpose we need to analysis the capabilities of the libraries provided by OpenCV in extend to their suitability in terms of accuracy, speed, and processing efficiency. The main focus is on achieving near real-time capability for path measurements and establishing the relationship between measurement speed and precision. The results of this analysis provide us with requirements and limitations for the mechanical setup.

2. MATERIAL AND METHODS

2.1 Hardware and Software Setup

The present analysis of an evaluation program is based on Python 3.8.8 with OpenCV 4.5.4. To enhance the functionality of the OpenCV libraries, we incorporated additional community contributions and CUDA (Compute Unified Device Architecture) support. This enables access to the CUDA cores of NVIDA graphic cards and allows the offloading of various computations to the GPU (graphics processing unit) instead of the CPU (central processing unit). The tests were conducted on the Windows 10 operating system by Microsoft.

For the hardware setup, we utilized an eight-core CPU (AMD Ryzen 7 4800H) with 16 available threads, operating at a maximum clock speed of 4.2 GHz. The GPU employed is an Nvidia GeForce GTX 1650 Ti, equipped with 1024 CUDA capable cores. Additionally, the system is equipped with 32 GB of Random-Access Memory (RAM). The hard drive used provided a read speed of 1278 MB/s and a write speed of 400 Mb/s.

2.2 Stereo-matcher in OpenCV

To extract a 3D information from two 2D images some requirements must be met. Firstly, the individual cameras of the stereo setup must have the same setup in terms of focal length and sensor size. Secondly booth cameras need to be spaced apart when looking at a scene with an optional inclination. Due to the difference in viewing, objects in the scene shift between the two images, with an increase in shift the closer they are to the camera. To determine the dimension of the shift, we must detect the corresponding region in both images and calculate the difference in their position. This task is handled by the stereo matchers. There are two types of matching algorithms: global and local matchers. A global matcher tries to match each pixel in the left image with a corresponding pixel in the right image. A local matcher, on the other hand, reduces the processing time by only matching induvial image. However, the accuracy of the result decreases. OpenCV provides the local matcher developed by Kurt Konolige in the function cv2.stereoBM(). In this implementation the matching is done within a search block with a minimum size of 5x5 pixels. The search is conducted by scanning each line of the right image for a selected region of the left image. The matching of the search blocks is based on the decision criterion "Sum of Absolute Differences" (SAD). In this, the total absolute difference of each pixel within the search block is formed from the reference of the left image and the gray values of the search line in the right image. The area with the smallest added up difference in the search line is then assigned to the reference. Limitation of the algorithm are therefore areas on the search line where no gray value differences occur. This matcher works with a subpixel accuracy of 1/16 pixel. [4]. In addition to the CPU version, a CUDA-capable version is implemented. However, this has no subpixel accuracy and only limited search range for shifts of 192 pixels, limiting close distance detection capabilities and sensitivity to changes in depth. Instead of a pure global matching algorithm OpenCV includes a hybrid of the two types, the Semi-Global Block Matching (SGBM). The Approach is based on Heiko Hirschmüllers development of the Semi Global Matching (SGM), in which, the corresponding points are determined from 16 different direction for each individual pixel. The OpenCV Version allows a reduced processing time by using search block similar to the local matcher and different amounts of search lines. Depending on the selected mode the search is conducted by a minimum of 3 (mode=3), 5 (mode=0/default), 8 (mode=4) or 16 (mode=1) different angles. The amount of search lines increases processing time and memory requirements but theoretically allows for better matching results in complex scenes [4]. As with the local matcher, a subpixel accuracy of 1/16 is achievable. Similar to the local matcher, a CUDA capable version is also implemented here. However, this showed insufficient results for images larger than 840x640 pixels and was not considered for further investigations.

To assess the accuracy in the matching result we applied the three stereo matchers (local-CPU, semi-global-CPU and local CPU) and their sub-variants on the Tsukuba image pair (Figure 1). The scene depicted in the image pair contains many areas challenging for stereo matchers, like the low-texture lamp, reflections and partially obscured background due to the view angles.
In addition, a ground-truth disparity map (Figure 2) is provided which visualizes the calculated shifts an ideal stereo matcher would determine. The accuracy therefore can be determined by calculating the difference between the calculated disparities and the ground truth. A pixel was deemed correctly matched, if its disparity value between the calculated map and the ground truth was less than two pixels, which corresponds to a depth error of 2 cm at the maximum disparity of 80. For the setup of the matcher we, applied a block size of 5 x 5 pixel for both the local and semiglobal matcher. This selection is represented in the reduces size of the disparity map (Figure 2) in comparison to the original left image (Figure 1).

Figure 1 left and right view of the Tsukuba pair. Distance between the cameras: 10cm, Focal length 615 pixel

Figure 2 ground truth disparity map of the Tsukuba pair

The processing time of the stereo matching was determined for the initial images size of the Tsukuba pair of 640 x 480 pixel. Additionally, we increase the size of the image up to 300% to 1280 x 960 pixel by utilizing OpenCV resize functions to assess the influence of the image size on processing time.

2.3 Pre- and post-processing

The main focus of this study rests on the development and optimization of an evaluation program to extract depth information from two 2D images. For this purpose, we developed a base program that handles all necessary steps for the depth extraction from the image pairs. While the discussed stereomatcher handles the core function of this program, some preparatory steps are necessary for images sourced from cameras.

Firstly, images sourced from cameras may exhibit distortions from the lenses used. Secondly, the discussed stereomatcher assumes that matching points only show a shift on the horizontal lane. As this heavily relies on the mounting of the individual cameras and the relative position of their sensors inside the housing, this cannot be assured. Therefore, rectification of the images is necessary beforehand (Figure 3). The libraries needed for the correction are also provided in OpenCV with the functions "cv2.initUndistortRectifyMap" and "cv2.remap". The necessary coefficients for both intrinsic and extrinsic calibration were acquired by performing a

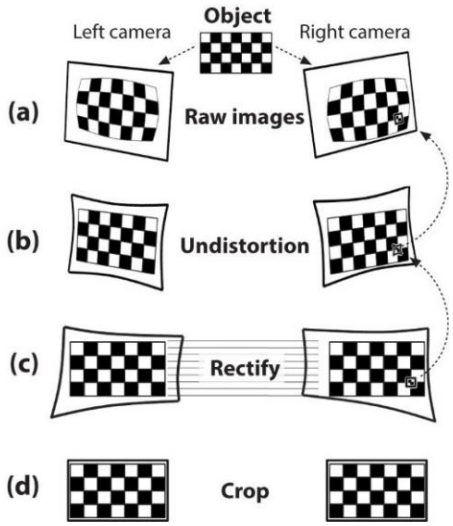

Figure 3 Preprocessing steps for stereo matching in OpenCV

preliminary calibration with a ChArUco target. As this calibration and the saving of the coefficients are done in advance, they are not part of this analysis.
After applying the corrections, the stereo matching can be performed, and the disparity map can be calculated. The conversion of the disparity map back to the 3D depth map is performed by "cv2.reprojectImageTo3D". Lastly, we save the depth information in a suitable format. Initially, the image format "png" was chosen, with the depth information encoded across two of the three color channels with the third canal used to store the disparity map.

In the starting configuration of this program, all described steps were performed in sequence. By analyzing the time required to process a dataset of image pairs, we further optimize the sequence in order to reduce the required time. The image set used for this purpose consisted of 30 stereo images with a resolution of 1280 x 864 pixel from our calibration set. In contrast to the evaluation of the stereomatcher the disparity search range was increased to 192 with the block size remaining at 5x5. Our aim was to a achieve a processing time of the whole dataset in less than second to mimic a real time evaluation.

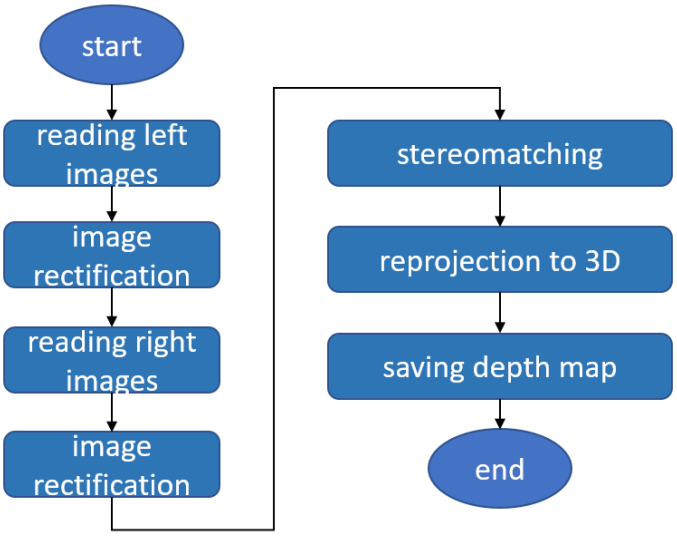

Figure 4 Initial sequence of the evaluation program. All steps in order

3. RESULTS

When comparing the calculated disparity maps of the Tsukuba pair among each other, a clear improvement of the SGBM (Figure 5, right) over the two local matchers (left and middle) can be determined. Especially the background and lamp shade are visually very close to the ground truth. When calculating the absolute difference (Figure 6) almost all larger areas show high compliance (dark areas). The disparity maps calculated with the local matcher in comparison show low compliances in these areas. Low compliance in both can mostly be found in areas concealed by the different angles of sight and the and areas with steep depth variations surrounding more homogeneous regions.
The change in mode for the SGBM had no significant influence on the conformity, reaching around 75 % in all cases. The lowest conformity with the ground-truth demonstrated the GPU version of the local matcher with 38.6 %, while the CPU version achieved a conformity of 42 %. Comparing them side by side its apparent that the variance between ground truth and the disparity map is higher in the GPU version.

Figure 5 Disparity maps of the Tsukuba pair calculated with the local matcher via CPU (left) local matcher via GPU (middle), and the semi global matcher via CPU with the default 5 search directions(right)

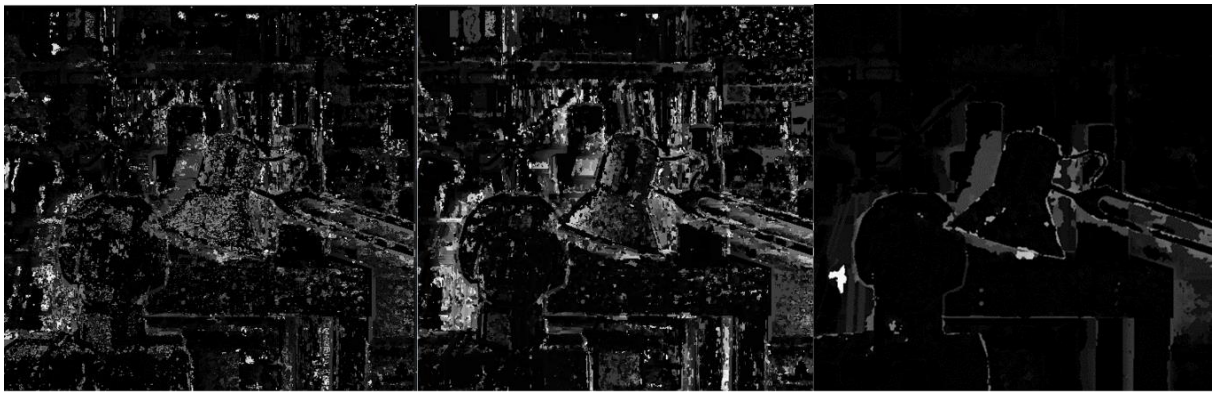

Figure 6 Differences between the calculated disparity map from Figure 5 and the ground truth from Figure 2

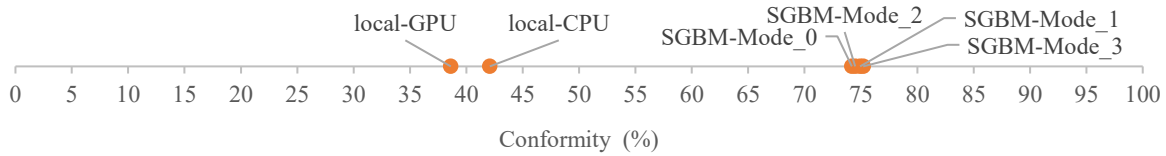

Figure 7 Conformity of the calculated disparity maps with the ground-truth

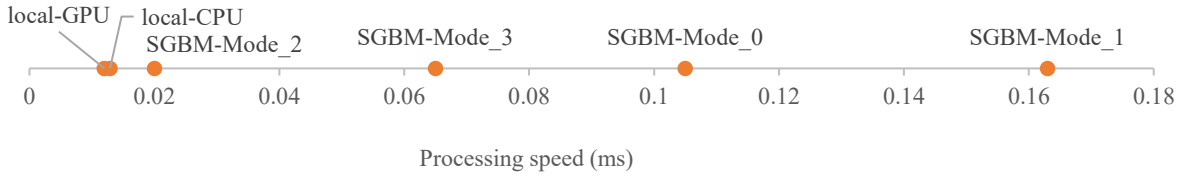

Figure 8 Processing speed of the various block matcher types of OpenCV

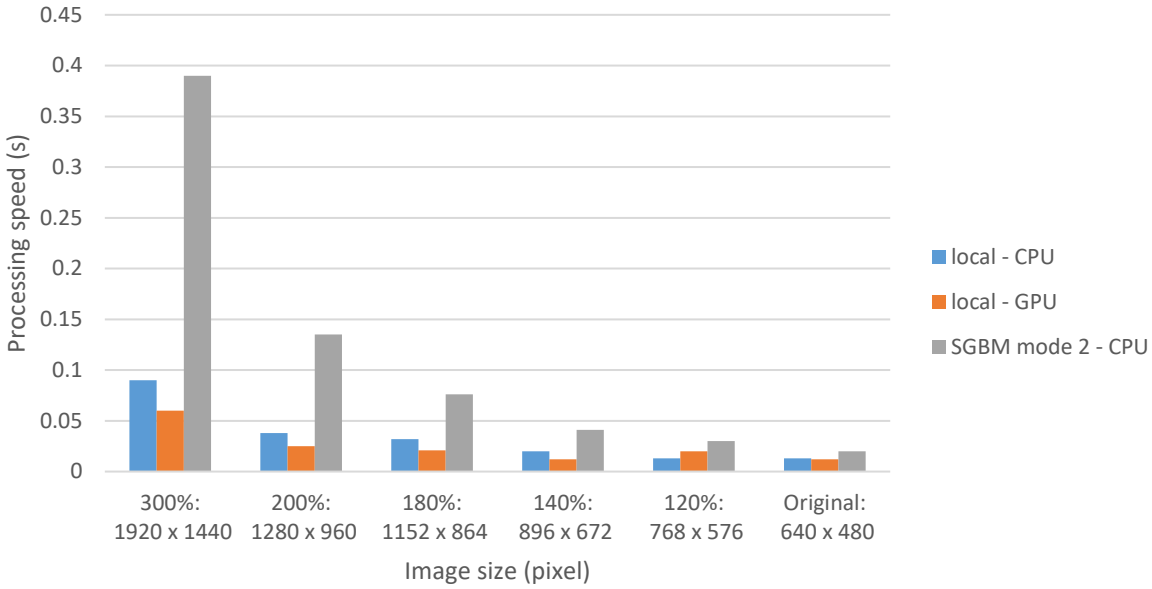

Figure 9 processing speed of the three fastest matcher types in relation to the size of the image

Conversely, the influence on processing time is larger. Here both local matchers process the images within 0.0129 s with the GPU version being slight faster at 0.012 s. The SGBM modes show a larger increase in time, with each mode almost doubling the processing time. The three-angle mode 2 processed the images within 0.027 s, while the five-angle mode achieves this in 0.065 s. The eight and sixteen-angle mode result in times of 0.105 s and 0.163 s in comparison.

With an increase in images size these processing times expectedly started to increase steadily with an increased images size. The increase to 200% more than doubled the time for the local matcher while tripling the default SGBM times for the CPU computations. The GPU version also doubled from 0.012 s to 0.025 s. A 400% increase in image size increased the processing time further to 0.072 s for the GPU version, while the CPU version of the same matcher increased to 0.16 s. The semi global matcher in its fastest configuration in contrast needed 0.7 s

The second part of this evaluation factored in the necessary pre and post processing steps for the stereo matching. Based on the time analysis we initially utilized the CPU-version of the local block matcher. The sequential processing of these steps resulted in a total processing time for the 30 image of 4.37 s. With preprocessing is taking up 32 % and post processing 54 % of this time. The next steps to reduce processing time were to parallelize the process as much as possible, this included the parallel preprocessing of the image loading and rectification, halving their respective times (Figure 11, left) to a total of 3.67 s. A additional parallelization to process

eight image pairs simultaneously cut the processing time further down to 1.56 s (Figure 11, right).

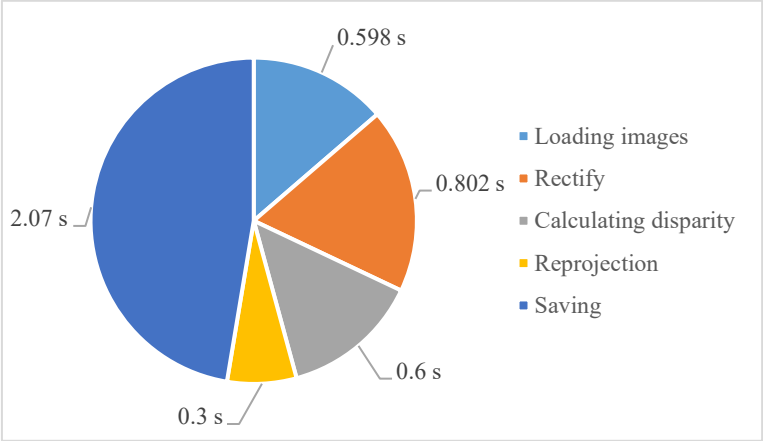

Figure 10 Averaged processing time of the 30 image pairs with sequential program flow, combined time: 4.37 s

In this context, the loading and saving time remain significant factors influencing the overall processing time. The implementation of multithreading requires all 30 image pairs to be loaded before distributing them across the threads. Therefore, the adjustable factors for optimization are limited to the saving format and block matcher.

Initially, the PNG format was chosen due to its small data size of a few kilobytes. However, to further minimize processing time, we investigated different saving formats without compression. These formats allowed us to directly save the floating-point depth maps. As a result, we switched to the ".tiff" image format, which reduced the saving time from 0.645 s to 0.045 s. It is important to note that these alternative formats generate larger file sizes. In the case of depth maps, the file size increased from 409 KB to 4.21 MB per depth map.

This optimization effort enabled us to achieve a processing time of 1.06 s using only the standard CPU libraries, falling just slightly short of our objective of one second (Figure 12, left). By leveraging the faster GPU version of the local matcher, we were able to further reduce the processing time to 0.8 s (Figure 12, right).

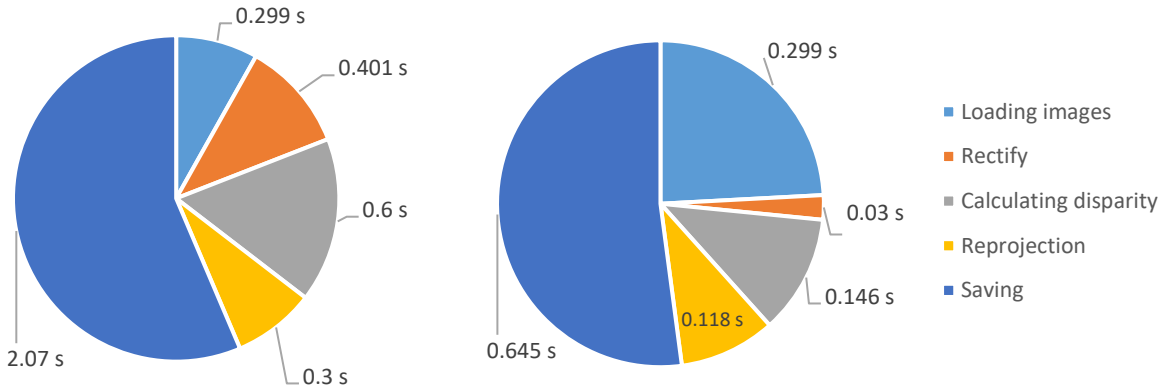

Figure 11 Averaged processing time for each step.
Left: with parallelization of the loading and rectification, processing time:3.67 s
Right: with parallelization of the rectification to savings across 8 threads, processing time:1.43 s

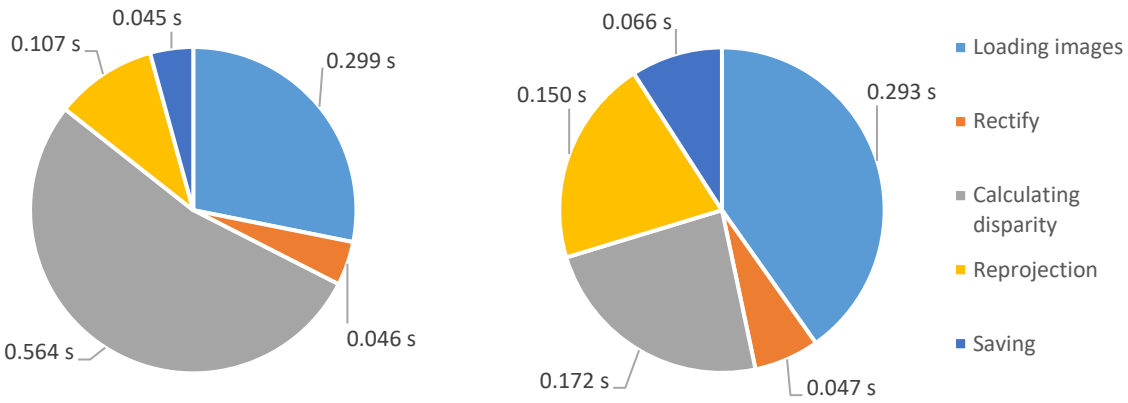

Figure 12 Final averaged processing time for each step.
Left: CPU only processing with improved save time by switching from png to tiff, processing time:1.06 s
Right: same optimization as left but utilizing the GPU version of the stereo matcher, processing time:0.8 s

4. DISCUSSION

In our assessment, we successfully achieved a significant reduction in the processing time required for 30 pairs of images to nearly one second or less. This performance holds the potential for near real-time evaluation of the forest path. However, the conditions under which we achieved these low processing times present obstacles to practical implementation.

First, we were only able to achieve these low processing times by relying on the local matchers, as even the fastest implementation of the semiglobal matcher has double the processing time compared to the local matchers at an image size of 640 x 480 pixels, and triple at a larger resolutions like 1280 x 960 pixels (Figure 9). These implementations are shown to be of low accuracy, especially at larger distances as shown in Figure 6. Since the GPU version in the middle lacks the 1/16 subpixel resolution of the CPU version, these errors are additionally amplified, resulting in the lowest conformity of 38% and higher deviations from the ground truth. The lack of sub-disparities severely limits the depth gradations that can be mapped when back-projecting the disparities onto a 3D map, resulting in a loss of geometric information.

The semiglobal block matchers on the other hand, all achieve high compliance rates of around 76% and exhibit minimal errors in the background and on homogeneous surfaces. The negligible differences observed among the subtypes may be a consequence of the small image size and the absence of larger homogeneous surfaces, thereby limiting the significance of the number of search directions.

For practical implementation, these factors imply that the semiglobal matchers are more effective at greater distances to the target. Consequently, we can position the cameras at a higher altitude on a test vehicle while preserving the depth gradation. However, a system using the GPU version would need to be positioned roughly 16 time closer to the ground to achieve the same depth gradation as the CPU variants. This closer proximity could require the use of additional cameras to cover the entire forest trail. It also increases the risk of the lenses being hit by gravel, depending on the distance to the ground.

Furthermore, the investigated image resolutions of 1280 x 864 pixels are relatively low for a system of this nature. A sensor with these dimensions imposes constraints on either the cameras

field of view or the resulting image resolution. Particularly when utilizing local matchers, it becomes crucial to employ a high-resolution setup to effectively capture the coarse-grained texture present on the forest road. Insufficient resolution and the resulting blur, leads to the loss of these fine textures resulting in a decline in matching accuracy, similar to the reduction observed in homogeneous surface. Therefore, to ensure optimal performance of local matchers, it is essential to maintain a sufficiently high resolution that preserves the fine details and textures found on the forest road. Achieving this would require either the utilization of more cameras or larger sensors. However, increasing the sensor size contradicts our real-time goal, since all investigated matcher types roughly doubles their processing time when the resolution is increased from 1280x960 pixels to 1920x1440 pixels (Figure 9).

Finally, our optimization strategy focused on maximizing the parallelization of processes. This approach resulted in a heavy utilization of the CPU during the calculations and achieved a utilization rate of 90-100%. Consequently, there is minimal available capacity for integrating additional components required to process the image stream directly from the cameras instead of reading it from memory beforehand. Since even the GPU version of our implementation showed performance spikes, adapting our program to less powerful systems such as the Jetson TK, which only has a quad-core 2.32 GHz processor, does not seem practical without further optimization and reliance on the integrated GPU and multiple systems. This approach would also depend heavily on the less accurate GPU stereo matcher.

5. CONCLUSION

The results of the study show that real-time forest path recognition in three dimensions is possible with OpenCV libraries. However, this performance depends on using the fastest matcher in OpenCV and low-resolution images. Despite the advantage of improved processing speed, the disadvantages appear in the form of significantly lower matching accuracy compared to the slower semi-global matcher. In addition, the lack of subpixel disparity further affects the quality of the results, as the achievable depth resolution is limited. Furthermore, increasing the camera resolution or integrating additional cameras is not possible without additional high-performance hardware.

In our particular case, opting for the slower semiglobal block matchers, with processing speed up to ten times lower, allows us not only to achieve the desired matching accuracy and depth resolution, but also to focus on a stereo setup with high spatial and depth resolution while having complete flexibility in the configuration of the measurement setup. Therefore, we decided to abandon the goal of real-time evaluation of the forest path and evaluate the path after the measurement run. This approach gives us the opportunity to perform further post-processing steps on the generated depth maps, such as correcting the slope position and merging the individual 3D maps into a continuous path profile.

6. ACKNOWLEDGMENTS

This project was funded by the Federal Ministry of Food and Agriculture based on a resolution of the German Bundestag. The work is related to the project “Contura-TU-I” (2220NR061B).

REFERENCES

List and number all bibliographical references at the end of the paper. The references can be numbered in alphabetic order or in order of appearance in the document. When referring to them in the text, type the corresponding reference number in square brackets.

[1] Mathavan, Senthan; Kamal, Khurram; Rahman, Mujib, “A Review of Three-Dimensional Imaging Technologies for Pavement Distress Detection and Measurements”, IEEE Transactions on Intelligent Transportation Systems, Institute of Electrical and Electronics Engineers Inc , New York, pp. 2353–2362, May 2015

[2] Girardin, Patricia; Valeria, Osvaldo; Girard, François,” Measuring Spatial and Temporal Gravelled Forest Road Degradation in the Boreal Forest”,Remote Sensing, MDPI, Basel, p.457, 2022, Nr. 14(3)

[3] Loprencipe, Giuseppe; Pantuso, Antonio, “A Specified Procedure for Distress Identification and Assessment for Urban Road Surfaces Based on PCI”, MDPI, Basel, p. 65, 2017, Nr. 7(5), S. 65

[4] Kaehler, Adrian; Bradski, Gary,”Learning OpenCV 3. Computer Vision in C++ with the OpenCV Library”.O’Reilly Media, Sebastopol, 2017

CONTACTS

M.Sc. Martin Richter email: richter.martin@tu-ilmenau.de

Dr. Ing. Maik Rosenberger email: maik.rosenberger@tu-ilmenau.de

