A Lightweight Synthetic–Baseline Stereopsis System for Autonomous Landing Site Selection in a Solar–VTOL Migratory UAV*

Stephen J. Carlson[1]

University of Nevada, Reno, 1664 N. Virginia, 89557, Reno, NV, USA
stephen_carlson@nevada.unr.edu

Abstract. This paper presents an initial prototype for a perception system for use on small UAVs for landing site detection. The system uses a down-facing monocular camera to collect images of an area of interest while translating horizontally in fixed-wing flight. Image pairs are compared using a plane–sweep algorithm to produce a disparity map of the scene, which is further translated and filtered to find viable landing site candidates. Such candidate sites would allow for an autonomous VTOL landing, enabling recurrent migratory mission behaviors. Steps for further development and implementation of the system in a real–time demonstration are outlined.

Keywords: Stereopsis · Structure-From-Motion · Perception · VTOL · Migratory-UAV

1 Introduction

1.1 Background

The UNR Robotics Workers Lab (RWL) does experiments in field robotics using terrestrial and aerial platforms [2,5]. The mission of the lab is to extend the capabilities of these robots in both range of physical abilities, and also in the autonomy behaviors that can be manifested in various environments [4,3]. The MiniHawk–VTOL is one example of the kinds of systems developed in this lab. The MiniHawk is a small UAV that has been developed in the RWL which is capable of both hovering and fixed–wing flight (VTOL) [9]. The MiniHawk has been used to demonstrate various behaviors such as re-configurable linking [6], landing–site detection using deep learning [1], solar recharge for recurrent migratory Missions [7,10,8], and marsupial teaming with ground robots. The project and its descendants are intended to extend the range and autonomy of UAVs such that human attendance or guidance is unnecessary. As such, these flying

* This material includes prior artwork and concepts that have been supported by the NSF Award: AWD-01-00002751: RI: Small: Learning Resilient Autonomous Flight. The presented content and ideas are solely those of the authors.

robots must be able to solve many problems on their own. This project aims to develop one of these behaviors with the MiniHawk as the parent robot platform.

The most difficult problem for terrestrial unattended (migratory) operations is the landing-site selection problem. When operating in an unstructured real–world environment, while it is possible to provide the system with a precomputed list of predetermined landing sites, or even a high–resolution terrain topology map, this constrains the operating region of the aircraft to only within this set, and the data storage requirements likely become intractable for a topology map of requisite detail. To be truly autonomous, the vehicle will have to make its own determination on landing site fitness as candidates are discovered through onboard sensing of the surrounding terrain. There are various sensory pathways for detecting terrain topology, such as Lidar or Radar, but the most mass–efficient and economical route is electro–optical sensing with cameras.

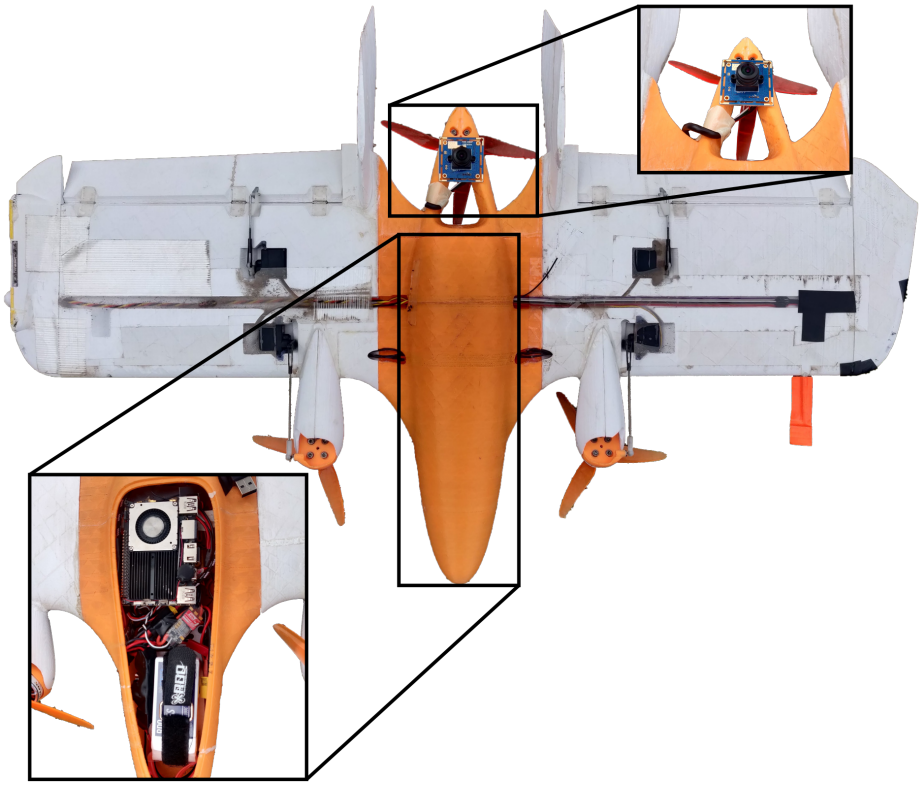

Fig. 1. The MiniHawk-VTOL with a RGB USB camera mounted for down-looking imaging, and the associated Khadas VIM3 SBC. Figure repeated from [1].

Prior work with the MiniHawk has shown the ability to detect landing sites using Deep Learning [1]. This method, however, is constrained by the prior

training of the network, and cannot be particularly generalized for all possible landing site solutions. In the previous experiment, the Deep Neural Network is trained from hundreds of examples which are human labeled, typically of large grass fields or sports surfaces, such as found at parks. The DNN was and is able to select these features consistently, but it is blind to any other viable landing surface, and furthermore, is completely unaware of the topology of the candidate surface. To have a generalized landing site detector, we need a visual perception system that can select based on the topology of the surface, not just the appearance.

A note on why we wish to implement our own vision system; while there are various commercial offerings which provide environment surface topology estimation using vision, these all are ill-equipt for our specific circumstances. Candidates such as the Intel RealSense D4xx series are only rated for up to 6 meters range, which is much too short for the necessary 100 meters that our vehicle typically will cruise at when scanning for landing sites. Our previous work with DNN landing site selection was accomplished using a lightweight USB RGB Web camera connected to a Khadas VIM3 single-board-computer, and this represents the rough limit for our payload capability. As we have control of the aircraft's stance and pose, we assume that we can use this to our advantage for the design of the vision system. Figure 1 shows the MiniHawk-VTOL with the down-facing monocular camera modification.

1.2 Problem Description

The primary solution for using vision to extrapolate topology (depth) is through matching features between two or more distinct images. Known as Multi-View Reconstruction Imaging, this is accomplished through either *stereopsis*, wherein the images are taken from two cameras affixed together with a known stereo baseline, or from multiple images taken by the same or different cameras at difference stances from the area of interest, known as *structure-from-motion*. Stereopsis, or stereo matching, is essentially a special case of structure-from-motion, where the camera pose is only translated, not rotated. This is represented in Figure 2, which shows the relationship between corresponding pixels in a pair of images for a given baseline distance between them. When stereo matching is performed in bulk, the result is known as a dense depth map.

A dense stereo reconstruction algorithm is used to create a depth map, also known as a disparity map. This algorithm finds the correspondence between each pixel in a rectified image pair. The prime assumption of this algorithm is that each image is rectified such that all rows between images are aligned, such that the correspondence between pixels becomes a 1D displacement search. This is a classical computer vision problem, with many algorithm implementations, but for this initial attempt at implementing our own system, we use the normalized cross–correlation algorithm. We assume that image pairs are captured from the aircraft in straight and level flight, such that each pair only differs in the baseline distance between aircraft position. Thus this becomes a simple stereo matching problem.

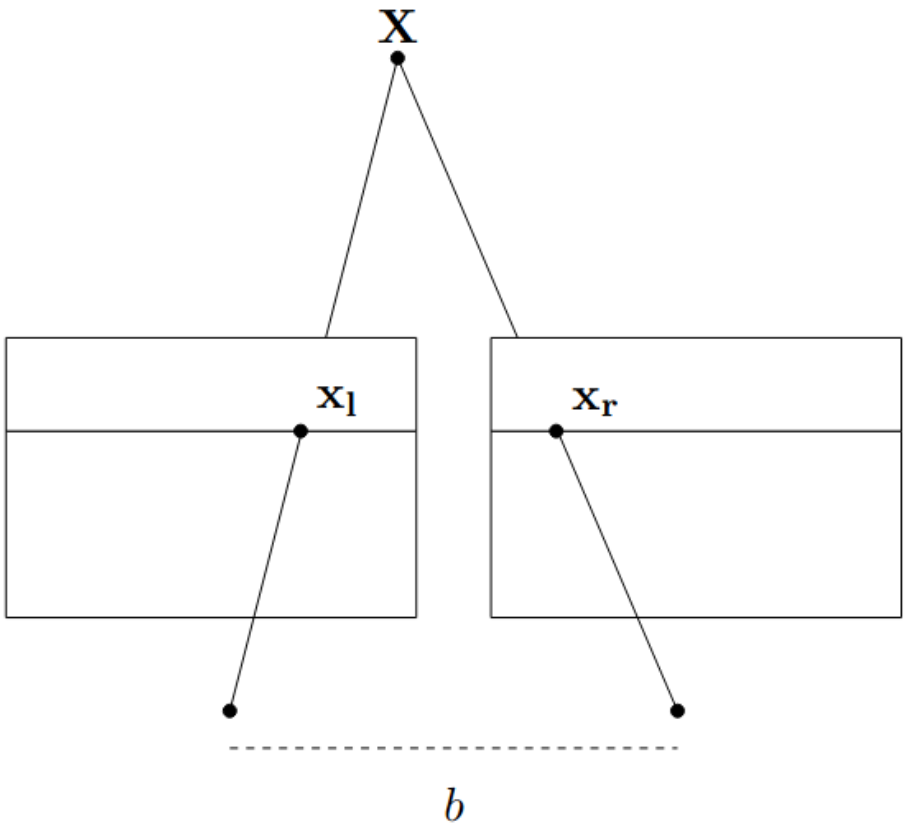

Fig. 2. Illustration of Stereo Matching from [12]. For a given point X, the corresponding pixels in the *left* and *right* images are X_l and X_r, with a baseline distance b.

The normalized cross–correlation stereo matching algorithm is taken from [12]:

$$ncc(I_1, I_2) = \frac{\sum_x (I_1(x) - \mu_1)(I_2(x) - \mu_2)}{\sum_x (I_1(x) - \mu_1)^2 \sum_x (I_2(x) - \mu_2)^2}$$

where I_1 and I_2 are two different image patches, with the mean of each patch as μ_1 and μ_2. The patches are correlated along the x-direction, and recorded to the associated pixel in the output image. The end result is a 3D structure with width and height determined by the input images, and the depth determined by the number of displacements tested in the x-direction. This process is called plane sweeping, since each displacement step corresponds to one plane in this cube. The Normalized Cross–Correlation Plane Sweep Algorithm is shown below.

Algorithm 1 Normalized Cross–Correlation Plane Sweep

$\mu_1 \leftarrow UniformFilter(I_1, width)$
$\mu_2 \leftarrow UniformFilter(I_2, width)$
$n_1 \leftarrow I_1 - \mu_1$
$n_2 \leftarrow I_2 - \mu_2$
$S \leftarrow 0$
$S_l \leftarrow 0$
$S_r \leftarrow 0$
$D \leftarrow 0$
for d in $DisparityRange$ **do**
  $S \leftarrow UniformFilter(Roll(I_1, d) \cdot n_2, width)$
  $S_l \leftarrow UniformFilter(Roll(I_1, d) \cdot Roll(I_1, d), width)$
  $S_r \leftarrow UniformFilter(n_2 \cdot n_2, width)$
  $D \leftarrow \frac{S}{\sqrt{S_l \cdot S_r}}$
end for
return $ArgMax(D, 2)$ ▷ Return the Max Score Index per pixel

Each instance of $UniformFilter()$ is a $2D$ window filter with a stencil of size $width$. The complexity is $\mathcal{O}(n^2)$ as the image size increases, with three summations over the local patch stencil. The effect of increasing the disparity test range is linear. The problem is NP-Complete [13], and is solvable but slow without parallelization.

2 Approach & Implementation

A CPU and GPU Version of the Normalized Cross–Correlation Plane Sweep were written, in C++ and HIP code, respectively. Both share an optimization in how the $ArgMax()$ is calculated; rather than search an entire 3D cube, the best score and associated disparity offset are updated at the same time as the correlation score is calculated. This flattens the cube to only a 2D image, but at the loss of the ability to search the volume for local maximia that might represent the true or best correlation value.

The GPU Implementation uses a default $BLOCK\_SIZE$ of 15 threads per dimension. As the problem is 2D, this results in 225 threads per block, and the number of blocks is determined by the input image size. The value of 15 is selected as this is the largest shared Greatest Common Factor for the image sizes that are tested against, being $1920 * 1080$ and $450 * 375$, while also remaining below the 1024 thread limit for $BLOCK\_SIZE^2$ threads.

Runtime performance metrics are registered in the code using the *ctime.h* library for the CPU implementation, and the *hipEvent* API for the GPU implementation.

Instructions for cloning and running the project are supplied in the project repository on GitHub, located at https://github.com/StephenCarlson/GPU_Stereopsis. In particular, the Runtime Notes Document details how to run the program and the expected behavior.

3 Experimental Results

The Stereo Match Plane Sweep program was tested against two different image pairs. The first is the "Cones" pair from the Middlebury Dataset [11], which is provided for testing stereo vision algorithms and provides a well–characterized ground–truth for comparison. The left and right input images are shown in Figure 3. Selected snapshots of the Disparity Map and Disparity Scores are shown in Figure 4. The Disparity Scores image shows the correlation coefficient for each pixel, where a value of 0 (black) is no correlation, and 255 (white) is a perfect correlation match for the patch around that pixel. The Disparity Map represents the approximate depth, where a lighter-valued pixel is "closer", and a darker value is "further" or infinite distance. Note that this is a non-linear relationship and must be corrected for the camera's intrinsic optical characteristics, so this a "raw" disparity, not distance.

Fig. 3. "Cones" input image pair. From the Middlebury Stereo Dataset [11]

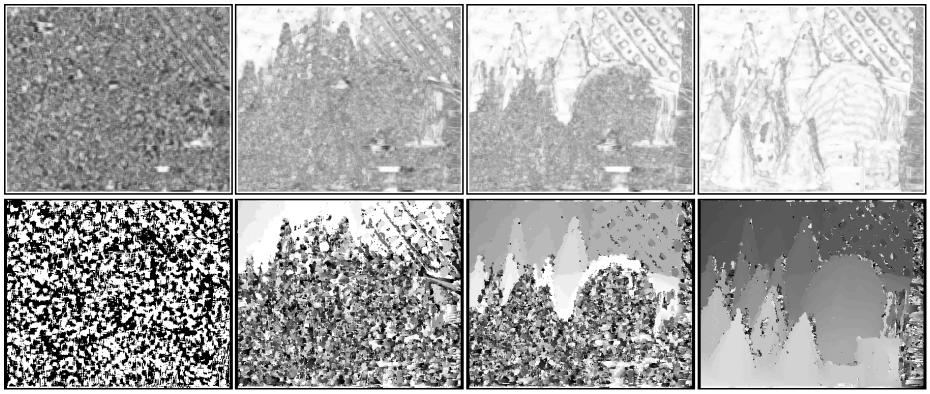

Fig. 4. Cones results for $d = 0$, $d = 20$, $d = 30$, $d = 59$ (left-to-right). Top Row is Correlation Score, Bottom Row is Disparity Offset.

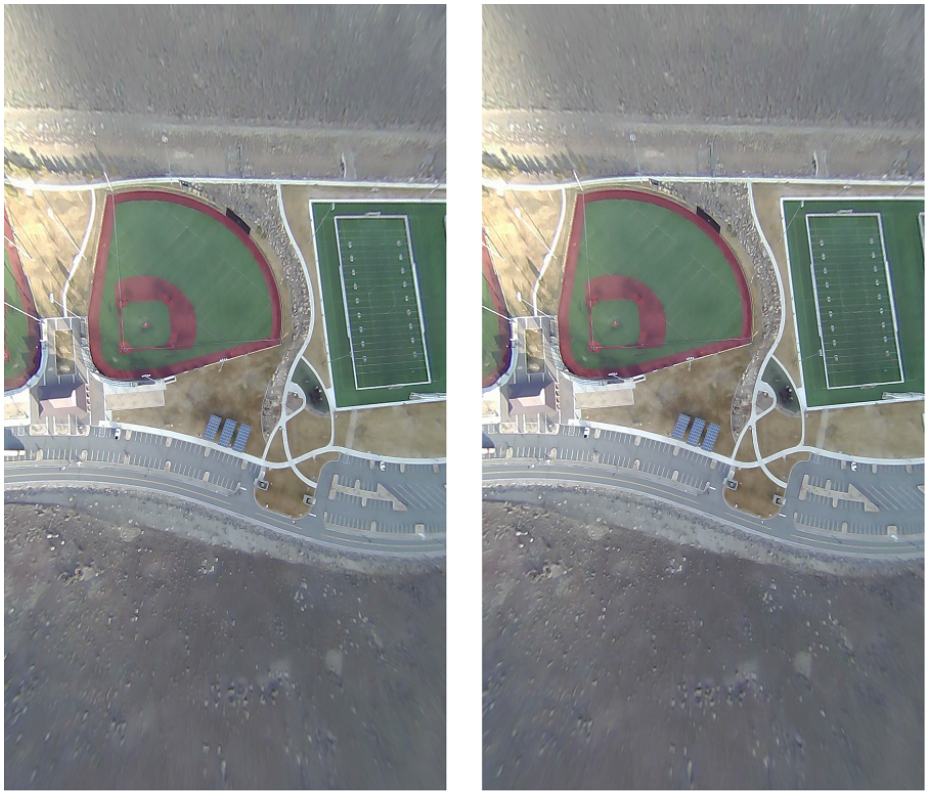

Fig. 5. Golden Eagle Park image input pair. Taken from data collected for [1].

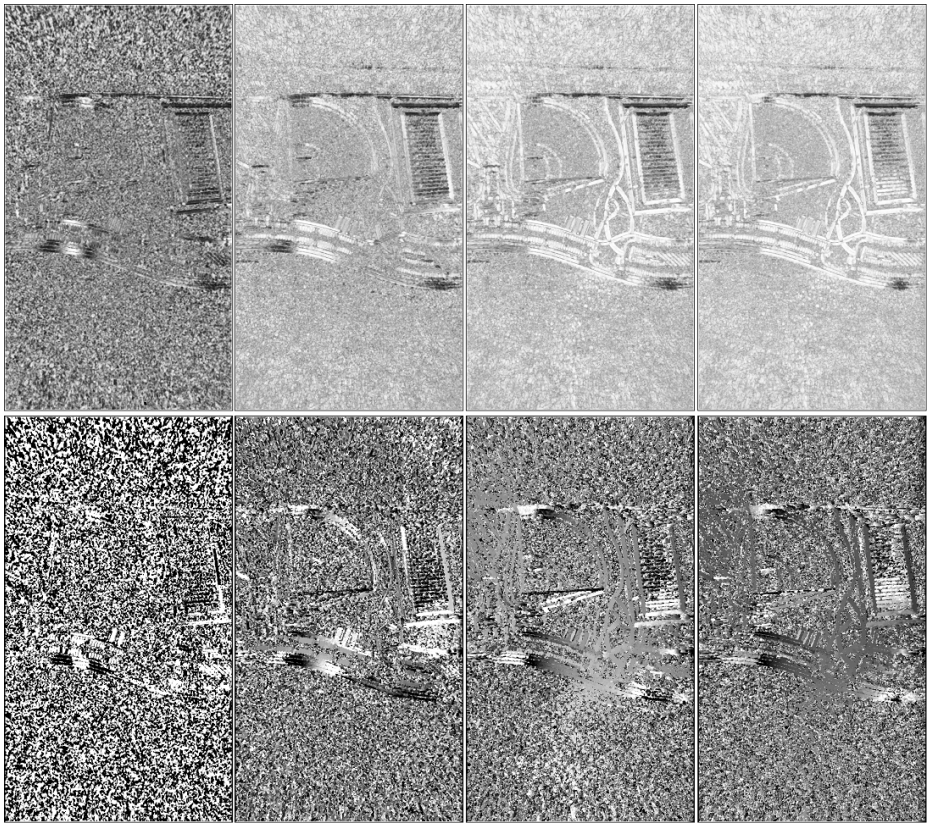

Fig. 6. Golden Eagle Park results for $d = 1$, $d = 15$, $d = 30$, $d = 49$ (left-to-right). Top Row is Correlation Score, Bottom Row is Disparity Offset.

After developing the program for the "Cones", a rectified pair of images was taken from a previous flight over Golden Eagle Park by the MiniHawk-VTOL on a previous experiment. Selecting an appropriate pair was difficult since the aircraft would couple any rotational motion to the captured video, and without the time to develop a image synchronization mechanism to account for this motion, only a very select few consecutive frames could be found that had little if any apparent motion other than pure translation over the ground. The best pair that was found are shown in Figure 5. The results of running the Plane Sweep algorithm on the Golden Eagle set are shown in Figure 6.

Running against the Golden Eagle image pair, the running time for each version is shown below in Table 1. The CPU implementation consistently took over a minute to compute the disparity map.

Table 1. Performance and Running Time

Test Condition	BLOCK_SIZE	Run Time (ms)
CPU - 1920x1080	–	67319
GPU - 1920x1080	5	123
GPU - 1920x1080	15	77
GPU - 1920x1080	30	67

4 Discussion

As shown in Table 1 above, the GPU pipeline greatly accelerated the stereo matching process, by well over two orders of magnitude. Granted, both implementations stand to be further optimized, but the GPU implementation at this point is sufficient for better than 10Hz of depth map updates.

The quality of the outputs is greatly dependant on how well the stereo image pairs are rectified and aligned. The Golden Eagle case has only a few regions of excellent correlation, and the corresponding depth map offset value appears to agree with the expected surface distance and topology. In the "Cones" case, the images are already well aligned, and the results are much cleaner with a much more rich collection of well–correlated pixels. Adding a stage in the pipeline that aligns the images is one of many features that can be implemented.

Finally, of greatest interest is determining how to accomplish the same pipeline scheme in the architecture on the Khadas VIM3 and other single–board computers. We hope to use this in conjunction with the DNN work we have demonstrated in [1] to build a robust landing site detector, such that the MiniHawk-VTOL can perform truly autonomous migratory behaviors in the field.

References

1. Arora, P., Carlson, S.J., Karakurt, T., Papachristos, C.: Deep-learned autonomous landing site discovery for a tiltrotor micro aerial vehicle. In: 2022 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 255–262. IEEE (2022)
2. Arora, P., Papachristos, C.: Mobile manipulator robot visual servoing and guidance for dynamic target grasping. In: International Symposium on Visual Computing. pp. 223–235. Springer (2020)
3. Arora, P., Papachristos, C.: Launching a micro–scout uav from a mobile robotic manipulator arm. In: 2021 IEEE Aerospace Conference (50100). pp. 1–8. IEEE (2021)
4. Arora, P., Papachristos, C.: Mobile manipulation–based deployment of micro aerial robot scouts through constricted aperture-like ingress points. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 6716–6723. IEEE (2021)
5. Arora, P., Papachristos, C.: Mobile manipulation–based deployment of micro aerial robot scouts through constricted aperture-like ingress points. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 6716–6723 (2021)
6. Carlson, S.J., Arora, P., Papachristos, C.: A multi-vtol modular aspect ratio reconfigurable aerial robot. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 8–15. IEEE (2022)
7. Carlson, S.J., Karakurt, T., Arora, P., Papachristos, C.: Integrated solar power harvesting and hibernation for a recurrent-mission vtol micro aerial vehicle. In: 2022 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 237–244. IEEE (2022)
8. Carlson, S.J., Papachristos, C.: Migratory behaviors, design principles, and experiments of a vtol uav for long-term autonomy. ICRA 2021 Aerial Robotics Workshop on "Resilient and Long-Term Autonomy for Aerial Robotic Systems" (2021)
9. Carlson, S.J., Papachristos, C.: The MiniHawk-VTOL: Design, modeling, and experiments of a rapidly-prototyped tiltrotor uav. In: 2021 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 777–786. IEEE (2021)
10. Carlson, S.J., Papachristos, C.: Solar energy harvesting for a land-to-recharge tiltrotor micro aerial vehicle. In: 2022 IEEE Aerospace Conference (AERO). pp. 1–8. IEEE (2022)
11. Scharstein, D., Szeliski, R.: High-accuracy stereo depth maps using structured light. In: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. vol. 1, pp. I–I. IEEE (2003)
12. Solem, J.E.: Programming Computer Vision with Python: Tools and algorithms for analyzing images. " O'Reilly Media, Inc." (2012)
13. Wikipedia: Computer stereo vision, https://en.wikipedia.org/wiki/Computer_stereo_vision