Method for Automated Data Collection for 3D Reconstruction

Mark Zaslavskiy[1], Roman Shestopalov[2], Alexander Grebenshchikov[3], Danil Korenev[4], Evgeny Shkvirya[5]
[1,2,4,5]SPb ETU "LETI", Saint Petersburg, Russia
[3]SPbU, Saint Petersburg, Russia
mark.zaslavskiy@gmail.com, rpshestopalov@stud.etu.ru, alexander-grebenshchikov@mail.ru, danilkorenev2004@gmail.com, zhshkvir@yandex.ru

Abstract— **The paper presents an automated method to guide a human operator during RGB image shot for improving the quality of further 3D reconstruction for low-rise outdoor objects and a use case for method application. The method provides automation as a three-step process: local analysis of images performed during the shooting, global analysis, and recommendations performed after the shooting. Method steps filter out defective images, approximate future 3D-model using Structure-from-Motion (SfM), and map it to human operator trajectory estimation to identify object areas that will have low resolution on the final 3D model, requiring additional shooting. Method structure does not require any sensors except RGB camera and inertial sensors and does not rely on any external backend, which lowers the hardware requirement. The authors implemented the method and use-case as an Android application. The method was evaluated by experiments on outdoor datasets created for this study. The evaluation shows that the local analysis stage is fast enough to perform during the shooting process and that the local analysis stage improves the quality of the final 3D model.**

I. Introduction

Nowadays, 3D reconstruction has become increasingly available for many domains [1]. Cheap drones and photogrammetry enable 3D reconstruction for large-scale outdoor objects such as buildings using only RGB data [2]. However, real applications of the technology are complicated with the data collection task. The operator (through the device or manually) is required to create a sufficient set of RGB images which should have enough overlapping, evenly cover all areas of the object, and contain enough features for photogrammetry. The knowledge of shooting enough and sufficient image sets is often based on an operator's experience because, usually, even compliance with the criteria described above does not guarantee the quality of a reconstructed 3D model. Moreover, the technical process of obtaining images in the case of large-scale objects is a challenging task itself due to the great diversity of forms, landscapes, and outdoor conditions. Dependency on the operator experience and uncertainty of the result limits the 3D reconstruction technology application and raises the possible prices. Therefore there is a problem of low-cost guiding the operator during the shooting process to make a sufficient image set which will be enough for successful further 3D reconstruction.

The paper aims to create an automated method for guiding a human operator during RGB image shots and for improving the quality of further 3D reconstruction for low-rise outdoor objects by image analysis. The operator uses a smartphone with a mono RGB camera and an inertial measurement unit. The operator can walk freely around the object. Another requirement is that method calculations take place fully on the mobile device without additional computational units. The method combines analysis of individual RGB photos with analysis of intermediate Structure-from-Motion (SfM) reconstructed 3D models and estimated coordinates of each photo to determine recommendations for reshooting problematic areas of the object.

The novelty of the proposed method is defined by a combination of automatic analysis of captured images' applicability for further 3D reconstruction and by the generation of recommendations. The recommendations form guidance for the operator, which allows increased quality and resolution of a future 3D model by providing a set of the object areas requiring capturing additional images or recapturing improper ones.

II. Possible alternatives and solutions

The problem of the automation of operator guidance for 3D reconstruction has several solutions. Some methods use smartphones and tablets with additional hardware to the main camera, such as a depth sensor or visual markers. Others use an algorithmic approach based only on RGB data. In order to determine the best possible method, a review of existing solutions should be done.

Due to the desired goal of this paper, the posed problem can be considered a special case of the Next Best View (NBV) problem. The NBV aims to determine the minimum number of object survey points (viewpoints) and their relative position required to collect the information about the object to achieve a certain resolution level[3]. Therefore, the review should focus on NBV papers about the specific restrictions of this research.

A. Paper selection criteria

Possible solutions were searched among papers that are dedicated to NBV problem as the main topic and to 3D reconstruction, photogrammetry, or building detailed 3D models of real-world objects as a secondary topic. The intermediate list was processed to filter out papers that do not cover operator guiding or quality estimation for the future 3D model. Finally, the list was filtered by the publication date (not older than 2018) and the journal impact factor criteria (> 3.0). As a result, four papers were selected for further review [4-7].

1) Plausible reconstruction of an approximated mesh model for next-best view planning of SFM-MVS [4]. The paper provides an approach for reducing time costs when constructing a dense model using the two-stage Structure-from-Motion - Multi-View Stereo (SfM-MVS) method. The method predicts the quality of a future dense model based on an approximate model obtained by the SfM method. Therefore, the quality of the final dense model can be predicted immediately after obtaining a cloud of sparse points using the SfM method. It will be possible to reshoot the current frames or add new ones for a more accurate dense model obtained by the SfM-MVS method. The method uses Delaunay triangulation from a sparse cloud of points and polygons and filtering tetrahedra. The following criteria predict possible errors and low resolution: the number of images per point of a sparse model, the area and sides size of a polygon, and the ratio of the base and median ratio.

2) Surface-driven Next-Best-View planning for exploration of large-scale 3D environments [5]. The method relies on a quadcopter with a depth camera. An approximate map of the object is loaded into the quadcopter; the method builds the route by solving the Traveling Salesman problem. Based on the data obtained, it dynamically adjusts to new circumstances; for example, it goes around the detected obstacles. Voxels are constructed using the Truncated Signed Distance Function method [11]. This algorithm converts information from the camera and depth sensor into voxels in 3D space. The Signed Distance Function (SDF) gives the distance from point X to the boundary of the surface. The method's advantage is a determination of whether a point is inside or outside the boundary of a surface.

3) Humanoid Robot Next Best View Planning Under Occlusions Using Body Movement Primitives [6]. The method uses a humanoid robot with a depth sensor as a shooting agent. The robot autonomously explores a point of interest (POI) in a selected area with an unknown environment. The robot's position on the surface is ultra-precise (deviation < 1 mm) and monitored by the motion capture system. The robot can solve two tasks: the most accurate study of the point of interest and the study of its entire environment. By relying on the exact coordinates of the robot and POI, the method automatically plots the route to the next viewpoint. If an obstacle is detected, the robot can tilt to study the POI better or determine a new point of view of the POI. When studying POI, the task of studying the environment is simultaneous. In difficult conditions, for example, incomplete visibility of the object, the robot automatically switches from the task of studying the POI to the task of studying the environment. While studying the environment, the solution simultaneously built a 2D surface map and a 3D model as voxels.

4) Mobile3DRecon: Real-time Monocular 3D Reconstruction on a Mobile Phone [7]. The method uses a neural network to estimate the depth of an object and a Simultaneous Localisation and Mapping (SLAM) algorithm to determine key points in space. The neural network chooses the most appropriate keyframe for the reference frame to simulate a frame from a stereo camera to determine the depth of the layers in the frame. The creation of a polygonal mesh occurs gradually. The CHISEL[10] algorithm creates a mesh in chunks for each scene. This approach does not guarantee the grid's constancy (consistency) and non-redundancy. Mesh generation is performed by a scalable voxel merge TSDF[11] to avoid voxel hash conflicts while progressively updating the surface mesh according to the TSDF variation of the newly merged voxels.

B. *Criteria for comparison*

For a correct assessment of methods described in papers [4-7], it is necessary to provide uniform comparison criteria. It should characterize the degree of automation, the need for additional preparation for shooting, taking into account the degree of detail of the model, and the ability to predict the quality of the future model:

- Method type: analytical or machine learning (ML-based).
- Method speed (on equivalent tasks) calculated using Geekbench[10]: quick (operating time less than 1 minute), slow (working time more than 1 minute).
- Achieved model resolution.
- Camera type.
- Shooting agent - a person, a robot, or an autonomous drone.
- Route planning. Presence or absence in the algorithm for constructing a route for shooting an object.
- Apriori information about the environment: the relative position of objects, landforms, and other factors.

Table I shows the comparison.

C. *Comparison results*

According to the results of a comparison, it follows that the [5] and [6], although they use an analytical method for solving the problem and plan a route for the agent to move, require a camera with a depth sensor and autonomous vehicles to move it. In addition, the speed of these algorithms is quite slow, and [6] is not applicable for large-scale objects. [7] requires a phone with a mid-range processor with a mono camera. However, for the software module's operation to determine this algorithm's frame depth, it is necessary to train a deep neural network. The model may change over time, which does not allow us to predict the reconstruction's resolution. The method [4] combines high speed with a transparent analytical approach for predicting errors and resolution of the 3D model without additional requirements to hardware.

TABLE I. METHOD COMPARISON

	Method type	**Method speed**	**Achieved model resolution**	**Camera type**	**Shooting agent**	**Route planning**	**Apriori information about the environment**
[4]	Analytical	Quick	High (error<1mm)	Mono camera	Operator	Missing	Missing
[5]	Analytical	Long	High (error ~13cm)	Camera with depth sensor	Autonomous flying drone	Dynamic. Changes as shooting progress	Preloaded map
[6]	Analytical	Long	No data	Depth sensor	Humanoid robot	Dynamic. The algorithm automatically switches from studying the object to studying the environment	The area of the terrain for the study is set
[7]	Machine learning	Quick	Depends on the duration	Mono camera	Operator	Missing	Missing

III. METHOD

A. The general idea and use case

The review showed that solutions for NBV are not fully applicable for solving low-cost automating shooting guidance for the operator. The methods mostly perform slow on mobile devices, require pre-trained ML models with a hypothesis of reconstructed objects, or require additional sensors. Moreover, the NBV solutions "as is" can provide only local recommendations for the next shot location and orientation, which might not be enough to guide the operator and does not increase the chance of successful 3D reconstruction. Approach [4] may seem promising for estimating mesh quality by the current image set. [6] is approximates mesh by using fewer computations and does not rely on any additional devices or information. However, in the case of large-scale outdoor objects, the solution [4] performance might not keep up with a large number of images. In contrast, each new image provides little information to improve the future 3D model quality. Instead, the analysis of each new image might be as fast as possible. It should not rely on all previous images but on only several last ones to keep the application responsible.

The ideas defined above shape a desired possible use case sequence for a solution - mobile application:

1) The operator opens the application on the mobile device.
2) The operator starts to take images of the object.
3) On each image taken, the application displays feedback about this particular image's local quality in the sense of a future 3D model and marks images that do not meet minimal conditions.
4) When the operator takes enough images or completes shooting the object from all sides, the application performs a global analysis of all images. It provides a set of recommendations for the operator.
5) After all recommendations are considered, and the application marks the image set as complete and acceptable, the operator sends the set to the detailed 3D reconstruction backend, which exists separately from the app.
6) The detailed 3D reconstruction backend builds the final 3D model from the image set using photogrammetry or another approach.

In order to implement the application, the following method for automating RGB image shooting is defined:

1) **Local analysis.** During the shooting process - analyze the current image to meet minimal requirements of future 3D reconstruction. It is proposed to use fast pre-processing algorithms designed for the last frame taken directly in the shooting process and aimed at filtering images that have obvious issues and most certainly will decrease the quality of a future 3D model.
2) **Global analysis**. After the shooting is complete, create an approximated mesh and evaluate its quality to predict possible problems during the full 3D reconstruction process. It is proposed to use information about the reconstructed camera trajectory and image coordinates (obtained during approximate 3D reconstruction). The data allows us to evaluate the degree of coverage of the object by images and the degree of overlap and check the loop closure.
3) **Recommendations.** Transform quality evaluation of the approximated mesh into recommendations to the operator about the additional shooting. Recommendations are formed based on aggregated global analysis data during the survey. The result of generating recommendations is a set of instructions (views) for additional object images, which the user should shoot. Such additional images will allow better model coverage for less human effort, avoiding sending incomplete image sets to the detailed 3D reconstruction backend.

The combination of three stages allows the operator to create a sufficient set of images while spending less time. The method filters out low-quality images and guides the operator.

The guidance allows the operator to create additional images in areas where the final 3D model might have a low resolution before the detailed 3D reconstruction.

B. Algorithms for local analysis

During outdoor shooting, some photo defects can appear due to external conditions such as lighting, weather, and obstacles. Blurry or overexposed images can reduce the number of tracked features and feature track length in the photo sequence. Moreover, features should appear in as many images as possible; therefore, checking the distance between corresponding features in adjacent photos is required. All mentioned checks can be performed right after a single photo shoot. In case of failures, it is almost certain that the image can reduce the quality of a future model, and therefore the shooting must be redone. In order to keep local analysis fast enough, the following algorithms were selected for the method: blurriness detection, closeness checking, and overexpose detection.

1) Blurriness detection. The algorithm for solving the problem was based on the article [12], where a comparative analysis of various algorithms was carried out. If a sharp image is recognized as blurry, the application can ask the user to repeat the shooting of the last image. If a blurry image is defined as sharp, the quality of the 3D reconstruction will be reduced, and the shooting process might need to be repeated from the beginning. Therefore it was decided that it was more important to recognize as many blurry images as possible with the risk of marking sharp images as blurry but not the other way around. Based on these requirements, the Laplacian operator algorithm was chosen to detect blurry images during the local analysis stage. The algorithm detects edges in a picture. It finds the variance of convolution with the Laplacian kernel. The image is considered sharp when the calculated value is higher than the threshold. Otherwise, the image is *blurry*.

2) Closeness checking. Because almost all photogrammetry solutions rely on the SfM method, a descriptor-based approach with image matching was chosen.

There are three stages of image closeness checking:

a) Descriptors searching on both images.

b) Matching descriptors from two images.

c) Check if corresponding descriptors are close enough (based on the threshold).

There are several ways to implement the first and the second step of the algorithm; the most effective are

a) Lucas-Kanade optical flow with forward-backward consistency check [13].

b) SIFT, AKAZE, ORB descriptors with brute force or FLANN-based matcher [14].

The corners used as descriptors in the Lucas-Kanade algorithm are more sensitive to scale and brightness changes than other types of descriptors mentioned above; however, this property of corners allows us to implicitly force the user to keep the distance to the target object and the level of brightness. Therefore, the first way of implementation was chosen. As a result, the following algorithm was implemented for image closeness checking:

a) Convert both images to grayscale.

b) Find corners on the first image using Shi–Tomasi corner detection algorithm [15]. These corners are called primary corners.

c) Find corresponding corners on the second image using the Lucas-Kanade optical flow method.

d) Calculate reverse optical flow (from the second image to the first); the secondary corners on the first image are the resulting corners.

e) For each primary corner, calculate the distance to the corresponding secondary corner and mark it if the distance is lower than a predefined threshold.

f) Evaluate the ratio of marked corners.

g) Two images are considered close enough if the result exceeds a predefined threshold.

3) Overexpose detection. The intensity thresholding method [16] was chosen for detecting overexposure due to the method's simplicity and high performance. Thresholding works by grouping pixels in an image into two classes, dark and light, depending on whether the pixel is below or above a predefined intensity threshold. After finding the bright parts of the image, it remains to find the ratio of their area to the total area of the image. The image is considered overexposed if the calculated ratio exceeds the predefined threshold.

C. Algorithms for global analysis

After taking shots of the object, there is a need to conduct a global analysis which will show the user the overall preliminary result. Since building a dense object model takes much time, it is appropriate to build an approximate mesh from a sparse point cloud. Moreover, by analysis of sparse point cloud mesh, it is possible to make predictions about dense object model quality. The SfM algorithm creates a sparse point cloud and an approximate mesh.

D. Algorithms for recommendation

The recommendation step improves the final 3D model quality by providing a list of locations around the object of interest (the recommendation list) where additional shooting is necessary. The algorithm consists of four steps:

1) Trajectory estimation.
2) Alignment of estimated trajectory and SfM data, identification of the image-level defects.
3) SfM mesh analysis, identification of polygon-level defects.
4) Recommendation list aggregation.

Trajectory estimation. After the global analysis step, there are estimated 3D point cloud and rotation-translation data for each image in the dataset. In order to form the recommendation, this data needs to be on a real-world scale and, therefore, to be possible to estimate future 3D model resolution using meters instead of abstract coordinates. For this purpose, the operator's trajectory is estimated using inertial measurement unit (IMU) - accelerometer, magnetometer, and gyroscope data. The estimation process consists of two connected steps: rotation and position calculation.

- IMU data-based rotation. The algorithm uses magnetometer+accelerometer measurements to calculate the rotation quaternion only when the user is not moving (at the moment of the shoot) because accelerometer readings during movement do not correctly represent the orientation. After all, this method assumes that the accelerometer shows the current gravity acceleration value. Integrating the gyroscope measurements with the Kalman filter and bias estimation is used when the user moves.
- IMU data-based position. Since the cheap accelerometers in mobile devices have poor accuracy, it was decided to use accelerometer readings in conjunction with a trajectory from the SFM algorithm.

It is essential to mention that IMU accuracy and related low-level API differ between different devices. Therefore, it is necessary to perform preliminary IMU calibration to keep the data uniform and avoid significant errors in the trajectory estimation step.

Alignment of estimated trajectory and SfM data, identification of the image-level defects. The alignment of IMU-based trajectory and SFM-based point cloud and trajectory is done by estimating the transformation (rotation, translation, and scale) between two sets of coordinates which minimizes pairwise distance using L-BFGS-B [17]. That allows us to find problematic images where calculated coordinates or/and rotations are significantly different from the estimations of SfM - the distance between two corresponding points or the angle between two corresponding rotations is higher than the predefined threshold. These image differences can arise due to image defects that have not been filtered out during the local analysis and lead to the misplacement of problematic images around the rough point cloud generated on the global analysis stage. Therefore it is necessary to include locations around those images in the recommendations list.

SfM mesh analysis, identification of polygon-level defects. During this step, the method analyzes an approximate mesh created by SfM on the global analysis stage. The analysis aims to detect polygon-level defects based on each polygon shape analysis: the number of images with this polygon (bigger is better), area of a polygon (smaller is better), length of polygon sides (smaller is better), and base to a median ratio (closer to 1 is better). After calculating those metrics, the threshold check is done. The desired resolution level chooses the threshold values for the final model (e.g., if desired resolution is 1 mm, the polygon area and the polygon side length should not be larger). All areas that do not pass the threshold are included in the recommendation list.

Recommendation list aggregation. The last step of the recommendation process includes aggregating all recommendations into a short set of locations for reshooting. The aggregation is done by the closeness criteria of related problem object areas - the goal is two group many small close areas into several larger ones. The desired final model resolution defines the size of the areas to be grouped into larger ones. After the list is aggregated, the list of positions and angles for reshooting is generated for the user. Positions and angles are defined relative to existing defect-free images.

IV. Implementation

A. Architecture

For the approbation of the method, a prototype Android application was developed that partially implements the method as an interactive shooting procedure. The local analysis stage is fully implemented, and the global analysis and recommendations are partially implemented. The MVC (Model-View-Controller) architecture was used for the application. The Model implements the logic of storing survey data and executing the steps of the method, the View provides an interface for surveying and a project management interface (one project is equivalent to one survey of one single object), and the Controller connects the Model and the View.

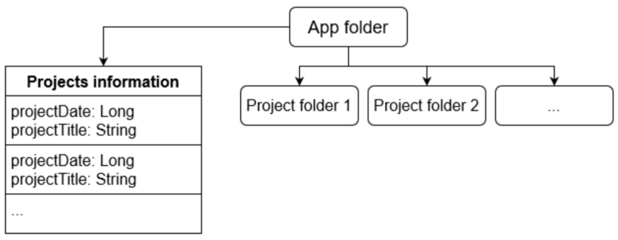


Fig. 1. Application files storage structure

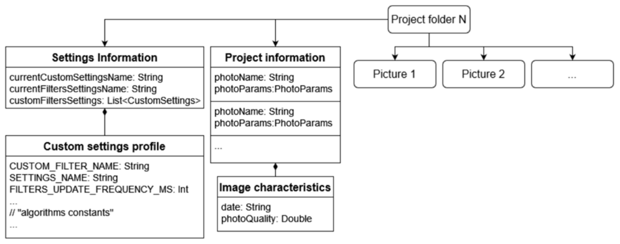
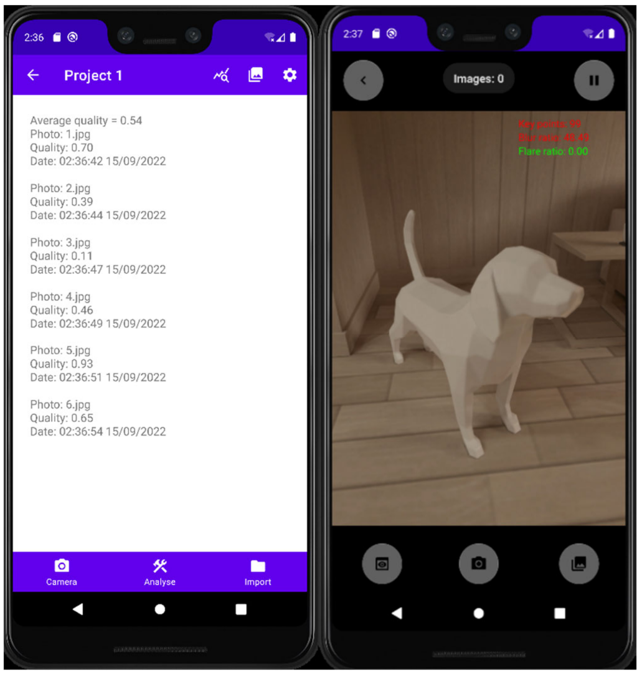


Fig. 2. Project files storage structure

The model structures user data using two entities: projects and snapshots. A project (Fig. 1-2) is a collection of settings for shooting and analyzing images and a set of images. The snapshot is the actual RGB image in JPG format, its metadata, and intermediate results of the execution of the method steps. The Model also stores the calibration data of the smartphone sensors (camera, accelerometer, magnetometer, and gyroscope). Data is stored using the device's file system - projects are mapped to directories, and metadata is presented as JSON files.

The main elements of the View (Fig. 3) include the sensor calibration screens, the project screen, the survey screen, and the image set analysis screens. The project screen is a central element since the user can access automated data collection for future 3D reconstruction: switching to the camera, starting image analysis, importing images from the smartphone's memory, setting algorithm parameters, accessing the image gallery, and statistics on images. The main space of the screen contains a brief description of each image - the name, quality metric, and timestamp.



Average quality = 0.54
Photo: 1.jpg
Quality: 0.70
Date: 02:36:42 15/09/2022

Photo: 2.jpg
Quality: 0.39
Date: 02:36:44 15/09/2022

Photo: 3.jpg
Quality: 0.11
Date: 02:36:47 15/09/2022

Photo: 4.jpg
Quality: 0.46
Date: 02:36:49 15/09/2022

Photo: 5.jpg
Quality: 0.93
Date: 02:36:51 15/09/2022

Photo: 6.jpg
Quality: 0.65
Date: 02:36:54 15/09/2022

Fig. 3. Project and camera screen

There are several basic controls on the camera screen: a button for showing the previous image (if any), a button for taking a new one, a gallery screen shortcut, and a button for local analysis. Local analysis works as a "hint" for the operator, which in text form in different colors (green and red), informs the user about the results of testing the current frame.

Showing the last image taken allows the user to make a new image close to the previous one by overlaying a semi-transparent last image on the current frame with a colored frame. Based on the image similarity algorithm, the frame can have three colors - green (high probability of matching), yellow (sufficient probability of matching), and red (low probability of matching).

B. Threshold setup

The choice of threshold values is important since it determines the effectiveness of the operator's work. Thresholds that are too low will create the risk of building a poor-quality final model. Thresholds that are too high will increase by shooting time. For flexibility, the user interface contains a view for customizing the thresholds.

The values of local analysis thresholds may vary depending on the object and the requirements for the final model. Of course, the global analysis and recommendation steps can partially compensate for the poor quality of the image with default thresholds by re-shooting. However, it is important to have a method for selecting the values:

1) Shoot the object in the necessary conditions to obtain a sufficient number of shots.
2) Divide the dataset into three groups using expert judgment by the number of visible issues (blurry, flare, big angle difference):
    a. "Suitable" - no issues.
    b. "Drawback" - one type of issue exists.
    c. "Unsuitable" - several types of issue exists.
3) Apply local analysis algorithms to the sample, and get values for each image.
4) Make a distribution and determine the boundary (lower or upper, depending on the algorithm) that separates "Suitable" images from other groups.

The threshold setup approach is not aimed to use frequently (on each object of 3D reconstruction). Instead, the approach is designed to provide initial values of threshold, which can be further adjusted to meet the requirements (desired final 3D model quality and resolution) for the particular object of interest. The approach can be automated by applying different optimization methods for the threshold selection problem described above.

C. Implementation details

The application uses Kotlin and open-source libraries. The execution of local analysis algorithms (determine_keypoints, detect_blurry, detect_flare, and angle_check algorithms) is performed using the OpenCV library. The local analysis procedures during shooting are implemented by a separate thread that periodically analyzes the current frame in the camera's visibility area. The construction of a sparse point cloud and its processing for global analysis and recommendations is performed using the BoofCV Java library.

V. EXPERIMENTAL EVALUATION

The experimental evaluation aims to justify the method by applying its simplest element (local analysis) to real-world data. Justification is based on two hypotheses: 1) the local analysis is fast enough to be computed during the shooting process on the mobile device 2) the local analysis allows for improved final 3D model quality by filtering out poor-quality images.

A. Dataset

To perform experimental evaluation, two datasets [18] of a low-rise outdoor object (tree trunk, radius 0.7 m) were prepared. The first one ("Bad") contains 37 images of the tree created with intentional issues (blurriness, flare, large angle between images). The second dataset ("Good") consists of 50 images created to provide the best possible images for future 3D reconstruction. Both datasets were shot using a POCO X3 NFC [19] smartphone and the same trajectory - a circle with a radius of 3.85 ± 0.2 m. The expert evaluation shows that the "Bad" dataset contains 24 "Suitable" images, 5 "Drawback" images, and 8 "Unsuitable" images. The "Good" dataset contains 46 "Suitable" images and 4 "Drawback" images without any "Unsuitable" images.

Both datasets were processed using local analysis algorithms. Threshold values were selected as follows:

- Extract_keypoints 15000,
- Detect_blurry 1000,
- Detect_flare 0.2,
- Angle_check 0.1.

Fig.4-5 shows the distribution of the values.

B. Local analysis computation time

For evaluating the computation time of local analysis, an experiment with both datasets [18] was performed. Each image from each dataset was processed five times through the application installed on the Realme C3 smartphone [20]. The resulting time was measured by the measureTimeMillis [21] function. Experimental results are presented in Table II.

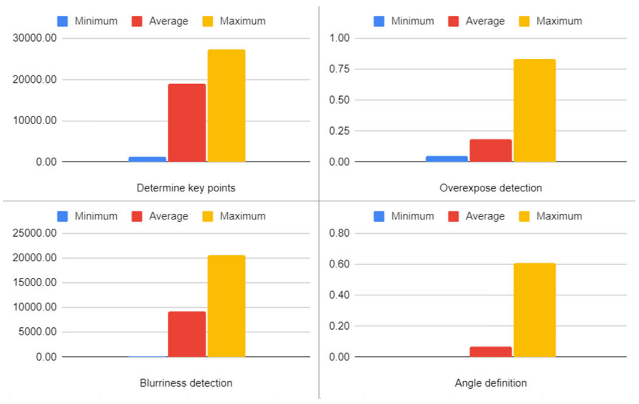


Fig. 4. Local analysis for the "Bad" dataset.

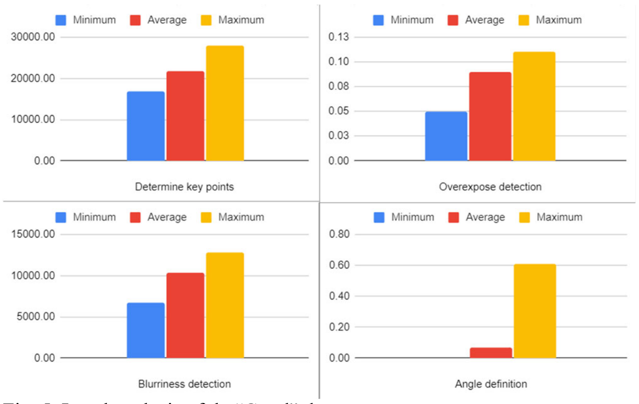


Fig. 5. Local analysis of the"Good" dataset

TABLE II. LOCAL ANALYSIS AVERAGE COMPUTATION TIME

Dataset	**Average time, extract keypoints, ms**	**Average time, detect blurry, ms**	**Average time, detect flare, ms**	**Average time, angle check, ms**	**Total average time, ms**
Bad	1458	16	66	257	1798
Good	1784	18	74	331	2208

Results show that on average the total computational time per each image is around two seconds. However, heuristics can improve it: calculate algorithms from fastest to slowest until there are two failed thresholds.

C. Local analysis impact on 3D model quality

In order to evaluate the quality of the filtration through the local analysis and the viability of the threshold setup approach, an experiment of 3D reconstruction was performed on "Bad" and "Good" datasets using Meshroom [22] software. Each dataset was reconstructed using the default photogrammetry pipeline. As a result, two different 3D models with textures were created (Fig 6-7).

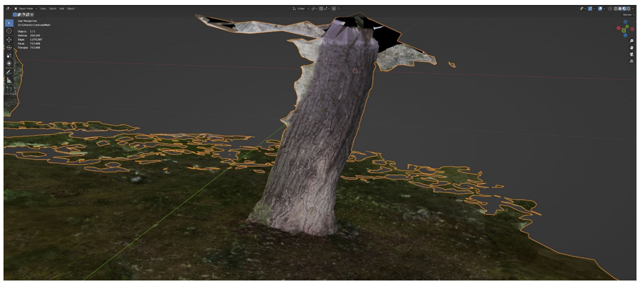


Fig. 6. 3D model for "Bad" dataset

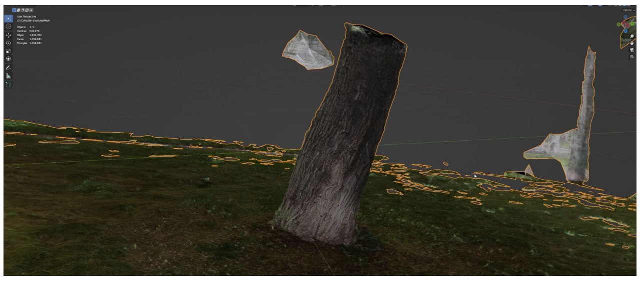


Fig. 7. 3D model for "Good" dataset

Both models contain the object (tree); however, the "Bad" model consisted of 713000 polygons instead of 1094000 for the "Good" dataset model which indirectly shows lower resolution for "Bad" case. The "Bad" model also contains more visible issues in the tree area, such as part of the background attached to the tree trunk. Therefore it can be concluded that filtering out by the local analysis can improve the quality of the future 3D model.

VI. CONCLUSION

In the paper, an automated method for guiding a human operator during RGB image shots and improving the quality of further 3D reconstruction for low-rise outdoor objects by image analysis is presented. The proposed method contains three steps and allows for automated data collection using average smartphones equipped with an RGB camera and IMU. Automation is achieved by providing two-level guidance for the operator: during the shooting process by filtering out defect images and after building a dataset by estimating future 3D model issues. The method can be configured to achieve the desired quality for the final 3D model by using a system threshold for each step. The method does not require any backend and can be implemented as an autonomous application without a backend server.

The proposed method was implemented as an Android application, which provides the user interface and business logic for the general method use case and implements the local analysis step. A threshold setup approach was defined for the method evaluation, and two datasets of a low-rise outdoor object were created. During experiments on the datasets, it was shown that local analysis implementation is fast enough to use during shooting and the method improves the quality of the final 3D model.

The future work includes evaluating the quality of the IMU-based trajectory estimation and calibration, global analysis and recommendation steps performance, accuracy, and impact on the final 3D model.

REFERENCES

[1] Transparency market research,3D Reconstruction Technology Market Outlook 2031, Web: https://www.transparencymarketresearch.com/3D-reconstruction-technology-market.html

[2] R. Tarek, and A.Gorodetsky. "Review of Unmanned Aerial System (UAS) applications in the built environment: Towards automated building inspection procedures using drones." *Automation in Construction* 93 (2018): pp. 252-264.

[3] L. M. Womg, C. Dumont, & , M. A. Abidi (1998, October). Next-best-view algorithm for object reconstruction. In *Sensor Fusion and Decentralized Control in Robotic Systems* (Vol. 3523, pp. 191-200).

[4] R. Moritani, S. Kanai, H. Date, Y. Niina, R. Honma. Plausible reconstruction of an approximated mesh model for next-best view planning of SFM-MVS, *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume XLIII-B2-2020, 43.

[5] G. Hardouin, F. Morbidi, J. Moras. Surface-driven Next-Best-View planning for exploration of large-scale 3D environments // *IFAC PapersOnLine* 53-2 (2020) pp 15501–15507.

[6] R. Monica, J. Aleotti, D. Piccinini. Humanoid Robot Next Best View Planning Under Occlusions Using Body Movement Primitives. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2493-2500 .

[7] X. Yang, L. Zhou, H. Jiang, Z. Tang, Y. Wang, H. Bao, & G. Zhang (2020). Mobile3DRecon: real-time monocular 3D reconstruction on a mobile phone. *IEEE Transactions on Visualization and Computer Graphics*, *26*(12), pp. 3446-3456.

[8] M. Klingensmith, I. Dryanovski, S. S. Srinivasa, and J. Xiao. Chisel: Real time large scale 3D reconstruction onboard a mobile device using spatially hashed signed distance fields. *In Robotics: Science and Systems, 2015*, (Vol. 4, No. 1).

[9] Motion Segmentation of Truncated Signed Distance Function based Volumetric Surfaces, Web: http://www.doc.ic.ac.uk/~bglocker/pdfs/perera2015wacv.pdf

[10] Geekbench 5, Web: https://browser.geekbench.com

[11] D. Werner, A. Al-Hamadi &, P. Werner . Truncated signed distance function: experiments on voxel size. In *International Conference Image Analysis and Recognition* (pp. 357-364). Springer, Cham.

[12] R. A. Pagaduan, M. C. R. Aragon, , &R. P. Medina. Iblurdetect: Image blur detection techniques assessment and evaluation study. In *Proceedings of the International Conference on Culture Heritage, Education, Sustainable Tourism, and Innovation Technologies-CESIT* (pp. 286-291).

[13] A. Plyer, G. Le Besnerais and F. Champagnat, "Massively parallel Lucas Kanade optical flow for real-time video processing applications." *Journal of Real-Time Image Processing* 11.4 (2016): pp. 713-730.

[14] S. A. K. Tareen and Z. Saleem, A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. In *2018 International conference on computing, mathematics and engineering technologies (iCoMET)* (pp. 1-10). IEEE.

[15] M.Bansal, M. Kumar,M. Kumar and K. Kumar, (2021). An efficient technique for object recognition using Shi-Tomasi corner detection algorithm. *Soft Computing*, *25*(6), pp. 4423-4432.

[16] Mabaso, M. A., Withey, D. J., & Twala, B. (2018). Spot detection methods in fluorescence microscopy imaging: a review, Web: http://researchspace.csir.co.za/dspace/handle/10204/10606

[17] C. Zhu, R. H. Byrd, P. Lu and J. Nocedal, (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, *23*(4), pp. 550-560.

[18] Kaggle, Tree datasets, Web: https://www.kaggle.com/datasets/alexgrebenshcikov/3Dreconstructordatasets

[19] Poco x3 NFC specs, Web: https://www.po.co/global/poco-x3-nfc/specs/

[20] Realme c3 specs, Web: https://www.realme.com/ph/realme-c3/specs

[21] Kotlin, measure-time-millis function documentationWeb: https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.system/measure-time-millis.html

[22] AliceVision | Photogrammetric Computer Vision Framework, Meshroom, Web: https://alicevision.org/