Image Based Volume Estimation Using Stereo Vision

Nándor Bándi*, Rudolf-Bálint Tunyogi*, Zoltán Szabó†, Eszter Farkas† and Csaba Sulyok*
*Faculty of Mathematics and Computer Science, Babeș-Bolyai University
RO-400084 Cluj-Napoca, Romania
†Codespring
RO-400664 Cluj-Napoca, Romania
nandor.bnd@gmail.com; rudolftunyogi@gmail.com; szabo.zoltan@codespring.ro;
farkas.eszter@codespring.ro; sulyok.csaba@cs.ubbcluj.ro

***Abstract*—The current paper proposes an image based volume estimation method and its implementation. Firstly, the camera positions are estimated based on the feature points found on the images using bundle adjustment. Using the camera positions, a dense point cloud is generated, onto which a Poisson surface and a convex hull is fit. Afterwards an intersection of the surfaces is determined and refined. Finally, the volume of the final surface is calculated, and converted to metric scale using a reference object.**

The implementation of the software system consists of two main parts: an Android client and a server. The server is responsible for image processing whereas the Android client application provides the user interface for taking pictures, and rendering the three-dimensional reconstruction of the scanned object as a result.

***Index Terms*—3D reconstruction; bundle adjustment; point cloud processing; stereo vision; volume estimation**

I. Introduction

Nowadays computer vision-aided solutions are becoming more and more common. Their usage has appeared in many industries, for example in agriculture, healthcare and the automotive industry [1]. In most cases, a repeatable task is automated, ranging from estimating the amount and quality of produce in agriculture to autonomous driving.

In this paper a stereo vision based volume estimation method is presented. Previous solutions for volume estimation are often restricted to specific objects. Past works mainly approach the problem of food volume estimation. W. Lo, Sun, Qiu, and B. Lo [2] tackle the issue using deep learning view synthesis from a single depth map. Hassannjedad et al. [3] estimate food volume using video capture and interactive segmentation. Dehais, Anthimopoulos, Shevchik, and Mougiakakou [4] estimate the volume of food portions using two view reconstruction. Xu, He, Khanna, Boushey, and Delp [5] tackle the issue of food volume estimation using a model-based approach. Gao, Tan, Ma, Wang, and Tang [6] combine multiple sensors during reconstruction and estimate food volume. The proposed method is general in the sense that it only assumes good illumination of the object, presence of a visible texture on it and that it is located on a flat, homogeneous surface beside a reference object.

Estimating the volume of an object proves to be challenging in many respects: the first one is the three-dimensional reconstruction of the object, and the second is the determination of the exact metric scale of the reconstructed model. In this project the three-dimensional reconstruction is performed by stereo-vision based methods. Bundle adjustment [7] is used to estimate the camera positions. Patch-based Multi-view Stereo [8] is used to construct a dense point cloud based on the estimated positions of the camera. Background noise is removed using image segmentation. The convex hull of the filtered cloud is used as a first approximation of the volume. A Poisson surface [9] is fit to the dense cloud in order to refine the previously obtained approximation. The determination of the metric scale factor is performed by *maximum-likelihood* estimation of the size of a reference object. The real volume is determined by the volume of the refined surface and the metric scale. The implementation of the presented method consists of two parts: an Android application that aids the user with taking the pictures and shows the result of the method, and a server that performs the image processing.

II. Volume estimation

The estimation of volume is achieved via three-dimensional reconstruction and metric scaling of the object. The proposed method is based on a state-of-the art pipeline for three-dimensional reconstruction that consists of:

- AKAZE feature-point detection and matching
- bundle adjustment
- patch-based multi-view stereo
- poisson surface reconstruction

AKAZE features are computed using OpenCV [10]. Camera positions are reconstructed via bundle adjustment using the previously mentioned feature-points and Bundler, an implementation provided by Snavely et al. [7]. Binarization using Otsu thresholding, morphological closing and dilation are used for segmenting the object from the background. Dense reconstruction using Patch-based Multi-view Stereo is done using *PMVS2*, an implementation provided by Furukawa and Ponce [8]. Surface reconstruction is achieved using the Poisson surface reconstruction implementation of Kazhdan, Bolitho and Hoppe [9]. As an initial estimation the convex hull of the reconstructed point cloud S_{PMVS} resulting from PMVS2 is used (see Fig. 1), however its use has many disadvantages.

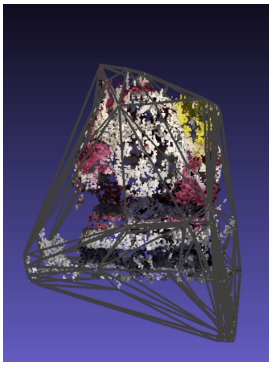

Fig. 1. The convex hull of the reconstructed point cloud

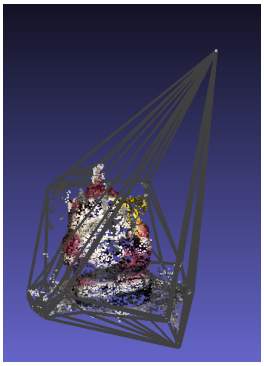

Fig. 2. Noise sensitivity of the convex hull: a single outlier point drastically changed the estimated volume

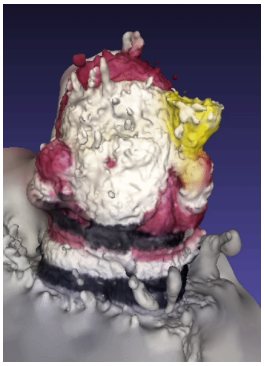

Fig. 3. The reconstructed Poisson surface

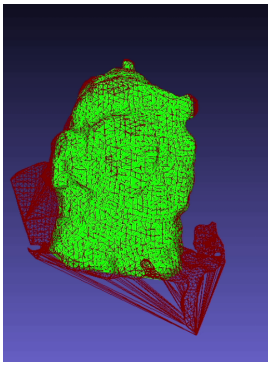

Fig. 4. Volume refinement via refined convex hull.

In the case of convex objects the volume of the convex hull is representative of their volume, not so for concave ones. Furthermore, convex hulls are very sensitive to noise, as the presence of a single outlying point can increase the estimation error significantly (see Fig. 2). In this section a better estimation using the reconstructed Poisson surface is proposed (see Fig. 3).

Given the object is wholly reconstructed the volume of its convex hull is an upper bound of its real volume, and can be written as

$$V_c = V + \varepsilon, \varepsilon \geq 0 \tag{1}$$

where V is the real volume and ε is the error of the convex hull estimation. V can be estimated by approximating a binary indicator function $\chi : C \rightarrow \{0, 1\}$ that maps a point of the convex hull to 1 if the given point is part of the object, and to 0 otherwise. The χ indicator function determines the ε error factor as this is equal to the volume of the subspace of the hull that is mapped to zero. The Poisson surface is an estimate of the boundary of the object, and can be thought of as an implicit approximation of χ. The surface is merely an estimate as it is not always closed, the bottom of the object can not be observed and reconstructed. The points above the surface can be mapped to 0 and the others to 1. This method provides a better estimate as its precision is limited only by the reconstruction precision of the Poisson surface and the noise level of the convex hull. Shadow spots can induce a significant amount of noise into the dense point cloud S_{PMVS} and increase the size of its convex hull. The prior approach is not a good estimate in this case, since the convex hull is not fit to the bottom of the object. The point cloud S_{BA} computed by bundle adjustment is less noisy due to the fact that it is generated by AKAZE feature-points that do not appear in these problematic spots. The subset of points in S_{BA}

$$P_{BA} = \{\, p \mid \; p \in S_{BA}, \;\; p \in C\,(S_{PMVS})\}$$

that are inside the convex hull of S_{PMVS} results in a less noisy convex hull which further reduces the ε error factor (see Fig. 4).

III. Metric scaling

Metric scaling is achieved using an Aruco marker [11] as a reference object. The size of the marker is determined by *maximum likelihood* estimation of its corners. The *maximum likelihood* estimation of a corner is the centroid of the reconstructed corner samples. The centroid is further augmented by assigning more weight to the samples that are visible on more images since these are likely to be of higher precision.

IV. Architecture

The software system consists of an Android client and a backend server. The role of the Android client is to help the user take photos and then upload the images to the web server for processing. In addition, the user can use the Android client to follow the progress of the image processing, and then view the resulting volume and three-dimensional model. The second part of the system is the image processing backend server. This can be accessed via an API server written in Node.js.

A. Backend server

The web server is implemented in accordance with the REST[1] API standard, the role of which is to provide the necessary endpoints for the Android client. The server is composed of image processing services that communicate through an ActiveMQ message broker. As the architecture of the server is microservice-oriented, scalability is achieved by running each service in a separate virtualized container, of which additional instances can be launched on demand.

B. Android Client

In this section, the Android application is presented.

A project within the application encapsulates the images taken and volume estimation result of the object. It is possible to create a project within the Projects menu, for which the user needs to enter a name, the metric size and identification of the marker and the resolution of the images. After creating the project, navigating to the main view displays an image similar to Fig. 5. The name of the project and the number of images

[1]Representational state transfer

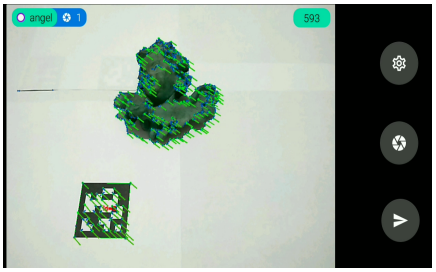

Fig. 5. After the first image is taken, the application indicates the amount of displacement since the last image

taken are displayed in the upper left corner. Feature points are counted and tracked during the taking of images. The feature points are marked with blue dots on the display, and the number of detected feature points is shown in the upper right corner. These are Shi-Tomasi corners [12] as AKAZE feature-points could not be tracked in real-time. These feature points are only indicative to help the user track the movement of the camera. After taking the first photo, the application helps the user make the right amount of camera movement by drawing the optical flow vectors. These connect the positions of the feature points in the last and current images, the length of which is also represented by colors. If the movement between the images is not adequate, the vectors will turn red. The goal is to keep the camera displacement within an optimal interval, i.e. these vectors remain green. A timeline available within the Status screen shows the progress and all the steps of the image processing. Once the processing finished, the calculated volume is displayed, which is also visible in the project view (see Fig. 6). On the left side there is a list of previously created projects, and on the right side are the results for the current project: the name of the project is displayed, along with the number of images and the volume of the object. The resulting textured three-dimensional model is displayed on a black canvas.

V. Experiments and Results

The estimation accuracy of the proposed method was tested. The hypothesis is that given enough pictures of an object the presented method can estimate its volume. This section presents the test environment, methodology and results. During testing the objects were placed on a homogeneous flat surface,

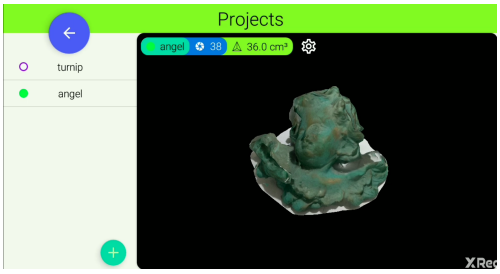

Fig. 6. The list of projects is shown on the left side and the result of the currently selected project is shown on the right side

Fig. 7. Representative objects for testing the presented method

with white background and lighting from above. The images were taken in two different ways. In the first method, the object was placed on a plate rotating at a constant speed, the camera was in a fixed position, and the images were taken at regular intervals. In the second method, the object remained in a fixed position and the camera moved freely around the object. In regards to the hardware, the server has an *Intel® Core™ i3-6100U* CPU with two 2.3GHz cores and 8GB of RAM. A *Xiaomi Redmi 5* type phone with a 12MP camera and a *Canon Powershot G16* type camera were used to take the test images.

The tests measure the effects of modifying the following principal variables:

- number of images
- image resolution
- quality of the object

The number of images ranges from 6 to 100. The images have a resolution of 2048×1536, 3200×2400 or 4000×3000. The quality of the object is ordinal in nature, indicating how the object meets the initial assumptions. Five objects of different qualities were tested and a total of 384 tests were completed. The number of test cases for objects is relatively evenly distributed. Within these test cases, the number of images and the different image resolutions are also evenly distributed. The results of the estimation of volume of three representative objects that are different from each other in terms of quality are presented below (see Fig. 7).

Object 1: Celery root

The first object is the root of a celery, the surface of which is textured, consequently it can be easily and accurately reconstructed due to the large number of feature point matches as can be seen in Fig. 8. and 9. Fig. 10. shows the relative

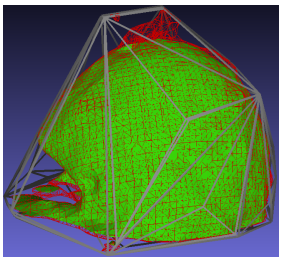

Fig. 8. The relative error of the hull is 0.28, after refinement it is reduced to 0.04

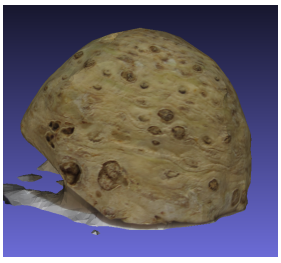

Fig. 9. The final textured model

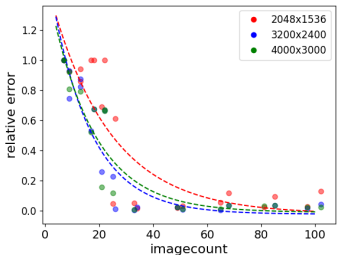

Fig. 10. The relative error decreases with increasing number of images and higher image resolution

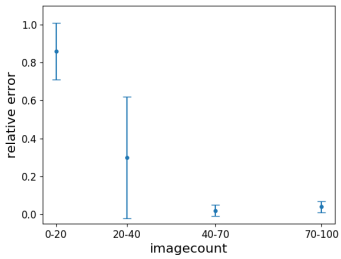

Fig. 11. The mean and standard deviation of the relative error is decreasing with the increasing number of images

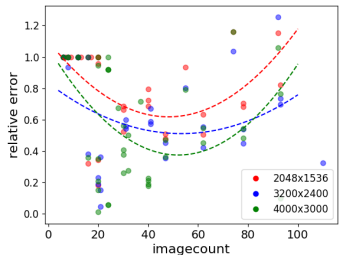

Fig. 14. The estimation is inaccurate and unstable

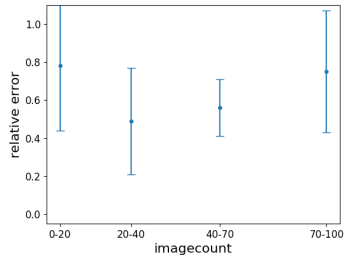

Fig. 15. The error mean and standard deviation increase significantly

estimation error as a function of image number and image resolution, respectively. The relative error is greatly influenced by the number of images and the number of feature point matches. The relative error correlates negatively with the number of images ($\rho = -0.8,\ p < 0.001,\ n = 62$) and with the number of feature point matches ($\rho = -0.8,\ p < 0.001,\ n = 62$) where ρ is the correlation coefficient, p is the p-value, n is the sample count. Considering the non-linear relationship of the variables, Spearman correlation was calculated. The number of feature point matches is largely influenced by the number of images ($r = 0.9, p < 0.001, n = 62$) because as the number of images increases, the reconstructed cameras are more closely positioned, which results in more common feature-point observations (see Fig. 16). The use of images with higher resolution also results in the detection of more feature points, decreasing the error. If the number of images is less than 20, a relative error of 0.86 is to be expected.

The accuracy increases between 20 and 40 images, the expected error is 0.30, but the method is unstable as the standard deviation of the error is 0.32. In the case of larger image numbers, the accuracy of the method significantly improves, as the mean of the error and its standard deviation decrease to less than 0.05 (see Fig. 11).

Object 2: Box

The second object is a box of medium quality, as its surface is less textured and partially reflective. As a result of this, the number of feature point matches is less, which makes the result more inaccurate and unstable (see Fig. 12).

The relative error is still negatively correlated with the number of images and feature point matches ($\rho = -0.79,\ p < 0.001,\ n = 94$), and ($\rho = -0.78,\ p < 0.001,\ n = 94$) respectively. Compared to the previous object, on average the error mean and standard deviation increased by 0.05 and 0.04, respectively (see Fig. 13).

Object 3: Angel statue

The third object is an angel statue, which is of poor quality, as there is little texture on its surface and there are unobservable gaps at its bottom. The correlation between the error and the number of images is negligible ($\rho = -0,24,\ p < 0,05,\ n = 106$). Compared to the previous object, on average the relative error is bigger by 0.28 and its standard deviation by 0.09 (see Fig. 15). Due to the small baseline between the cameras and the small size of the object an increasing tendency of the error in the case of higher image count can be observed (see Fig. 14). Another significant source of error are the imperceptible gaps on the bottom of the object. In the absence of a reconstruction, the method estimates the bottom of the object to be convex, and consequently the estimated volume becomes bigger.

Running time

The parallelization of feature-point identification, matching and image segmentation significantly reduces the execution time. The running time of the method is linear in terms of the number of images, in the case of higher image resolution, the rate of increase of the running time is higher (see Fig. 17). As an example for 100 images with a resolution of 2048×1536, the execution time is 446 seconds, whereas in the case of

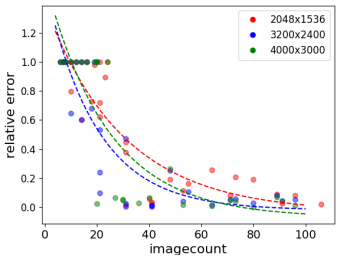

Fig. 12. The error also decreases exponentially

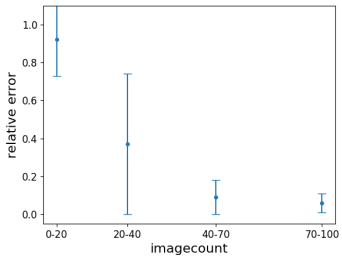

Fig. 13. The mean and standard deviation of the error are larger than in the case of the first object

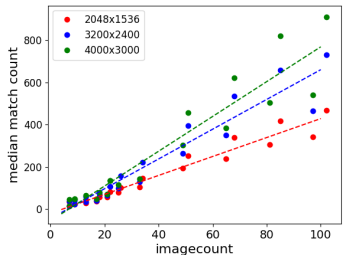

Fig. 16. The number of feature point matches increases with the number of images and higher image resolution

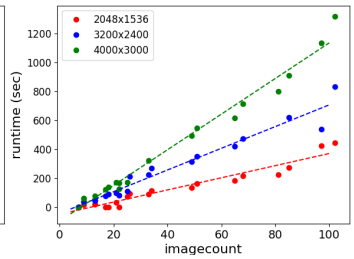

Fig. 17. Execution time grows linearly with respect to the number of images

TABLE I
VOLUME ESTIMATION ACCURACY

	Relative error				
Object	Mean	Std deviation	Min	Max	n
Celery root	0.033	0.032	0.003	0.129	28
Box	0.095	0.115	0.002	0.473	46
Porcelain figurine	0.481	0.294	0.077	1.000	30
Angel statue	0.598	0.261	0.099	1.255	56
Glass	0.879	0.247	0.074	1.000	26

TABLE II
EXECUTION TIME

	Average Running Time (seconds)		
Object	2048×1536	3200×2400	4000×3000
Celery root	122	234	399
Box	136	267	398
Porcelain figurine	87	192	258
Angel statue	89	216	258

4000×3000, with the same number of images, the running time is 1319 seconds.

Summary

During testing the efficiency, accuracy, and limitations of the method have been analyzed. The method is robust for high-quality objects with an accuracy of more than 90% given adequate number of images and resolution. Accuracy for objects with non textured and imperceptible parts is significantly reduced regardless of the number of images or their resolution.

Table I summarizes the accuracy of the estimation of the objects, assuming the number of images is acceptable, that is at least 30. The objects are listed in decreasing order with respect to their quality. Table II summarizes the execution time of the presented method. The expected running time is 221 seconds with the testing hardware, although this depends on the number and resolution of the images. In the case of the glass, the running time is not indicated as the processing was not completed due to the low number of feature point matches.

VI. CONCLUSIONS AND FUTURE WORK

A method has been presented which, based on modern reconstruction techniques, is able to estimate the volume of certain objects with at least 90% accuracy if the necessary conditions are met. Additionally, an application that aids the user with taking the pictures, communicates with the centralized server, informs the user in real-time about the state of the image processing, and then displays the estimated volume together with the reconstructed object, has also been presented.

This method can be improved in many ways. Segmentation can be improved with the help of object recognition algorithms, so the estimation could work robustly in the case of images that contain a complex background. The reference object could be omitted using sensor fusion methods that can estimate the position of the camera on a metric scale (Extended Kalman filter, Visual inertial odometry) [13]. It is important to mention, that these methods could only be applied if the estimate they give is sufficiently accurate, because as the scaling problem is poorly conditioned, any detection error would have a significant effect on the final estimation. The reference object can also be omitted in the case of a closed system, where images are taken by several calibrated cameras in fixed positions. The application could provide a more immersive user experience, for example by re-projecting the reconstructed model using the Aruco-marker.

REFERENCES

[1] R. SZELISKI, *COMPUTER VISION: algorithms and applications*. SPRINGER NATURE, 2020.

[2] P. W. Lo, Y. Sun, J. Qiu, and B. Lo, "Food volume estimation based on deep learning view synthesis from a single depth map," *Nutrients*, vol. 10, p. 2005, 12 2018.

[3] H. Hassannjedad, G. Matrella, P. Ciampolini, I. D. Munari, M. Mordonini, and S. Cagnoni, "A new approach to image-based estimation of food volume," *Algorithms*, vol. 10, no. 2, p. 66, Oct 2017.

[4] J. Dehais, M. Anthimopoulos, S. Shevchik, and S. Mougiakakou, "Two-view 3d reconstruction for food volume estimation," *IEEE Transactions on Multimedia*, vol. PP, pp. 1–1, 12 2016.

[5] C. Xu, Y. He, N. Khanna, C. J. Boushey, and E. J. Delp, "Model-based food volume estimation using 3d pose," *2013 IEEE International Conference on Image Processing*, 2013.

[6] J. Gao, W. Tan, L. Ma, Y. Wang, and W. Tang, "Musefood: Multi-sensor-based food volume estimation on smartphones," *CoRR*, vol. abs/1903.07437, 2019. [Online]. Available: http://arxiv.org/abs/1903.07437

[7] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism," *ACM Transactions on Graphics*, vol. 25, no. 3, p. 835, 1 2006.

[8] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multi-view stereopsis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1362–1376, 2010.

[9] H. Hoppe, "Poisson surface reconstruction and its applications," *Proceedings of the 2008 ACM symposium on Solid and physical modeling - SPM 08*, 2008.

[10] Official opencv documentation. [Online]. Available: https://docs.opencv.org/2.4/

[11] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, p. 2280–2292, 2014.

[12] J. Shi and Tomasi, "Good features to track," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*, 1994.

[13] J. Civera, A. J. Davison, and M. M. M. José, *Structure from motion using the extended kalman filter*. Springer, 2012.