Web Application for Reconstructing Sculptures as 3D Models from Images

Mary Cris Joy Cantimbuhan and Asst. Prof. Maureen Lyndel Lauron

I. Introduction

A. Background of the Study

Back in the 1960s, computer engineers have been creating three-dimensional (3D) models. In our current age, it is used in many things such as animation, games, visualization and 3D printing [1].

Using multiple images in reconstructing 3D models has been around for decades. There are many approaches on this, such as *silhouette exploitation, pathwork* and *photo consistency with homography* [2]. All of these are still relevant and some of them are used together with other techniques.

Structure from Motion (SfM) is a technique to obtain camera parameters for 3D model reconstruction. It has been used since 1999 [3], until today. OpenMVG uses this as its basis [4].

The study will focus on reconstruction of sculptures using SfM-MVS (Structure from Motion-Multi-View Stereo). Where SfM by OpenMVG will be used in reconstructing a basic 3D point cloud, and OpenMVS will be used in creating mesh and applying texture [5].

A web application will be developed as the presentation of the end project.

B. Statement of the Problem

Sculpture is a form of art usually in 3D. There are places that are famous for their sculptures, such as Paete, Laguna. Given that, not everyone can personally go there to appreciate the work. In addition, some sculptures are not always available for people to investigate due to their historical value. [6]

With that, people are only left with pictures. But these images by themselves do not always provide the information needed. A virtual version of the sculpture would help to give a better visual information.

In this day and age, a web application is a good way to enable users to access different services. The user do not have to do heavy computations in their computer, and they do not have to install anything *complicated*.

C. Objectives

Generally, the study aims to develop a web application that will accept images from users, and then produce a 3D model from those images. Specifically, this study aims:

- To develop a web application for 3D reconstruction from images
- To use SfM-MVS workflow in creating 3D models
- To assess the applicationś ability to reconstruct 3D models from images within the limitations and to cater the user's needs.

D. Scope and Limitations

The studyś goal is to develop a web application which accepts series of images, then outputs a 3D model of the object. But not all images are useful for this. The following contains the characteristics of invalid images:

- grayscale
- transparent
- dark
- reflective

The focus is to create 3D models of sculptures. Other objects may also be modelled, but the quality is not assured.

Part of the limitations is that the series of photos must be taken in one direction, and must be of good quality. Suggested camera is one with 8MP and sensor size of 4.7mm. Number of megapixels does not automatically equate to image quality, so sensor size must also be considered [7]. Many cameras of smartphones nowadays have that specifications, so a professional camera is not required.

II. Literature Review

This section contains literature and studies associated with 3D modelling techniques, together with a useful insight on what framework to use in the application itself.

A. Related Studies

Reconstructing 3D models from images has been around for decades, that many approaches have existed to do this [2].

Mulayim, Atalay and Yilmaz [8] provided a clear explanation of 3D reconstruction from images. Their method is silhouette based, but they also incorporated photo consistency in their study. They have a controlled imaging environment, where they used a round table. They projected the silhouettes and carved off the excess voxels using photo consistency.

Patchwork is a representation method introduce by Zeng, Paris, Quan and Sillion [9]. Patches are stitched together in coarse level, and smaller patches are combined in fine level. This method captures concavities, and produces seamless model even with the existence of hidden regions. It also preserves details in large scale modelling.

In 2010, Lee and Yilmaz [10] focused on homography exploitation and photo consistency for these plays a major

role in 3D reconstruction. Their method assures construction of convex and concave areas, without camera calibration and pose estimation. According to their results, their method performs better than the silhouette based ones.

B. Libraries

OpenMVG's global reconstruction is an open-source implementation of Moulon, Monasse and Marlet's study in 2013 [11]. They used structure from motion (SfM) to estimate camera poses and image orientation. It means that the horizon does not have to be in the same place in the pictures, as long as continuity of the photos is assured. They were able to get precise translation directions and accurate camera positions. They follow the usual SfM pipeline of *image analysis, feature computation, matches detection, and reconstruction*. The main change is in the reconstruction part, where the mentioned adjustments are applied.

Although a 3D point cloud is already present after OpenMVG's SfM pipeline, the result is obviously still lacking in terms of being "whole". It can be improved by using OpenMVS. It reconstructs a dense version of the point cloud from previous SfM pipeline, then creates a mesh. The mesh is refined to acquire specific details that are probably not acquired in previous step. After that, texture is applied using the images used in the SfM pipeline [5].

C. Web Application Frameworks

There are many 3D modelling websites in the Internet available today. Some of them are SketchUp, Tinkercad, Clara.io and Mixamo. All of these has the ability to show its users 3D models, since their goal is to enable its users create models. With that, a web application for the current problem needs a 3D engine to support it.

BabylonJS is a JavaScript framework for building 3D games [12]. Many demo applications are available in their website. It supports loading OBJ files, which is an option file format in the 3D reconstruction. It is relatively new compared with its famous competitor, ThreeJS, but it has a fast-growing community for newcomers. In addition, BabylonJS supports any browser that is WebGL compatible, such as IE 11+, Firefox 4+, Google Chrome 9+, and Opera 15+.

Together wth BabylonJS, AngularJS and NodeJS can also be used in the web application. Angular can take care of the front-end processes [13]. Node.js, on the other hand, will be on the server-side, since it was built on Chrome's V8 JavaScript Engine.

D. Definition of Terms

- 3D - three-dimensional
- IDE - integrated development environment
- MVS - multi-view stereo
- SfM - structure from motion
- voxel - equivalent to a point in the 3D space

III. Materials and Methods

This section contains the full methodology of the study

A. Materials

Listed below are the materials that will be used for the study:

1) A laptop with the following specifications for development:
   - RAM: 3.7 GB
   - Processor: Intel® Core™ i3-4005U CPU @ 1.70GHz x 4
   - OS: Ubuntu 14.04.5 LTS
   - OS-type: 64-bit
2) A smartphone with the following specifications for taking pictures in testing phase:
   - RAM: 2GB
   - CPU: 1.2GHz
   - OS: Android OS v6.0.1 (Marshmallow)
   - Display: 5.5"
3) Node.js, Angular, and BabylonJS will be used for the development of the web application.
4) OpenMVG and OpenMVS libraries shall be used for the 3D reconstruction.

B. Methods

3D Model Reconstruction:

1) Image Analysis: The width, height and EXIF focal length are extracted from the images themselves. EXIF focal length is the *focal length* field in an image header. Width and height are both in pixels, so that will be acquired from the *resolution* field. Regarding the focal length (it must be in pixels for future computations), it can also be added by the users themselves as an input information, or be estimated by the application, given that the camera sensor size is known. The sensor size information is present in many products, so it is easier to provide than the focal length itself.

In the case of focal length not being part of userś input, it will be estimated as [14]:

$$focal\_length = max(width, height) * \frac{EXIF\_focal\_length}{sensor\_size}$$

Principal points on *X* and *Y* axes will also be estimated:

$$ppx = \frac{width}{2}$$

$$ppy = \frac{height}{2}$$

The camera that will be used is a pinhole camera with three parameters, where the k_n are the radial distortion coefficients:

$$X_d = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) X$$

A view is created for every image, and each view will be checked for valid intrinsic data. The initial intrinsic matrix is shown below:

$$K = \begin{bmatrix} focal\_length & 0 & ppx \\ 0 & focal\_length & ppy \\ 0 & 0 & 1 \end{bmatrix}$$

Using the parameters, the intrinsic data is computed. Intrinsic data are deemed *valid* if the adjustments used for pinhole camera with three parameters fixed the distortions in the image. There are specific matrices to show axis skew, focal length, and principal point offset [15]. All images with valid intrinsic data are documented in a JSON file.

2) Feature Computation: The JSON file will be loaded here. Using that data, an image describer is created using SIFT Image Describer [16]. The optional masking is also done here, if the user wanted to mask the images. A global mask, or a mask for every image will do.

Finally, features and descriptors are extracted for every view and are stored as *.desc* and *.feat* files.

3) Matches Computation: The geometric model to be computed is the fundamental matrix. The intrinsic data with the views are loaded.

The image describer used in previous step will be used again here. The descriptors extracted in the previous step will be matched with each other using Approximate Nearest Neighbor L2 matching [17]. Those are the putative matches. The pair list are then matched exhaustively and then using photometric matching [18].

Next is the geometric filtering of the putative matches using the upper bound of a Contrario Model Estimation threshold [19]. Pair with poor connection are removed from the list. The geometric filtered matches are saved, together with the adjacency matrix created from the matches.

4) Global Reconstruction: This part is heavily based on the study by Moulon et al [11]. Again, intrinsic data of the views are loaded. The collected features and matches from the above steps are also read for the configuration of the reconstruction parameters. False relative pairwise rotations are removed. After that global rotation and translations are computed.

Using triangulation, link tracks are put in triplets for validation. 3D point cloud and camera poses are ensured after the two-step Bundle Adjustment. At this point, the SfM pipeline is done. But the result is still composed scattered and noisy dots.

5) Export to OpenMVS Format: OpenMVS has specific parameters to be filled, so there is a separate step for this. All intrinsic data are collected, with the camera (pinhole camera). Undistorted images, poses and structure must be defined so OpenMVS would not have to compute for them again. This step creates a *.mvs* file.

6) Reconstruct Dense 3D Point Cloud: This is an optional step, but doing this would produce more detailed, and accurate model. The parameters added in the *.mvs* file are validated first, before processing. Images with no valid neighbors are removed, best views are selected as reference image.

All 3D points visible in the reference image are extracted. The depth and normal maps are estimated using triangulation of sparse 3D point cloud and interpolating normal and depth of all points in the images. Small segments are them removed, and small gaps are filled as much as possible. Lastly, all valid depth amps are combined in one 3D point cloud.

7) Create Mesh: Using Delaunay tetrahedralization [20] and graph-cut method [21], mesh is reconstructed with the consideration of point cloud input (either the one from OpenMVG, or from the dense 3D point cloud reconstruction). The Delaunay triangulation of the point cloud input is reconstructed one-by-one in the point insertion.

Each edge is weighted, and the graph-cut algorithm splits the tetrahedrons to acquire surfaces. An optional mesh refinement can be done to smooth out the surface, but this will take more time. Given that the unrefined and refined meshes does not have much difference, the unrefined mesh can serve the purpose of 3D virtual representation of an object.

8) Apply Texture: Each surface in a view is rendered. Every face is assigned the best view of them, together with a mean color. Texture patches are assigned in every view, plus the faces which will use the patch.

An array of faces viewed by each image is extracted. The projection area is computed in each face. Outliers, or views where the projection is very different from the other views are sized down, or are removed. Border pixels of texture patches are also removed after an amount of dilation. After that seam leveling and merging of texture patches with overlapping rectangles take place. The patches are arranged to fit the smallest possible texture image. Some texture patches' sizes are increased in cases where a few increase will fit the current position.

Final texture patch coordinates are then computed, and the final model will have a separate *.png* file containing all the texture patches.

The Web Application: Node.js will be used in the back-end. Angular and BabylonJS shall work together in the front-end.

REFERENCES

[1] (2016, August) What is a 3d modeling? things you've got to know nowadays. [Online]. Available: https://archicgi.com/3d-modeling-things-youve-got-know/

[2] S. Paris. (2011, December) Methods for 3d reconstruction from multiple images. [Online]. Available: https://people.csail.mit.edu/sparis/talks/Paris_06_3D_Reconstruction.pdf

[3] T. Jebara, A. Azarbayejani, and A. Pentland. (1999, May) 3d structure from 2d motion. [Online]. Available: http://www1.cs.columbia.edu/~jebara/htmlpapers/SFM/sfm.html

[4] (2013, February) Openmvg. [Online]. Available: https://github.com/openMVG/openMVG

[5] (2015, May) Openmvs. [Online]. Available: https://github.com/cdcseacave/openMVS

[6] The art institute of chicago. [Online]. Available: http://www.artic.edu/visit/museum-guidelines/why-we-dont-touch

[7] L. Shu. (2013, January) Why small sensor size may leave your photos feeling inadequate. [Online]. Available: https://www.digitaltrends.com/photography/image-sensor-size-matters/

[8] A. Y. Mulayim, U. Yilmaz, and V. Atalay. (2003, February) Silhouette-based 3-d model reconstruction from multiple images. [Online]. Available: https://www.cv.tu-berlin.de/fileadmin/fg140/Silhouette-based_3D.pdf

[9] G. Zeng, S. Paris, L. Quan, and F. Sillion. (2006, November) Accurate and scalable surface representation and reconstruction from images. [Online]. Available: http://ieeexplore.ieee.org/document/4016556/

[10] H. Lee and A. Yilmaz. (2010, January) 3d reconstruction using photo consistency from uncalibrated multiple views. [Online]. Available: https://pdfs.semanticscholar.org/b681/3af8ba286121f27cbdaa426fc24bb5a81aad.pdf

[11] P. Moulon, P. Monasse, and R. Marlet. (2013, December) Global fusion of relative motions for robust, accurate and scalable structure from motion. [Online]. Available: http://imagine.enpc.fr/~moulonp/publis/iccv2013/

[12] D. Catuhe. (2013, July) Babylonjs. [Online]. Available: https://www.babylonjs.com/

[13] (2010, October) Angular. [Online]. Available: https://angular.io/
[14] N. Snavely. (2008, January) Estimating the focal length of a photo from exif tags. [Online]. Available: http://phototour.cs.washington.edu/focal.html
[15] K. Simek. (2013, August) Dissecting the camera matrix, part 3: The intrinsic matrix. [Online]. Available: http://ksimek.github.io/2013/08/13/intrinsic/
[16] (2015, December) Introduction to sift (scale-invariant feature transform). [Online]. Available: https://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html
[17] N. Ailon and B. Chazelle. (2006, May) Approximate nearest neighbors and the fast johnson-lindenstrauss transform. [Online]. Available: https://pdfs.semanticscholar.org/4da1/dada177523e379da86c75396db4db7f0b2cf.pdf
[18] M. Kolomenkin and I. Shimshoni. (2006) Image matching using photometric information. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.5040&rep=rep1&type=pdf
[19] P. Moulon, P. Monasse, and R. Marlet. (2013) Adaptive structure from motion with a contrario model estimation. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-37447-0_20
[20] G. Miller, D. Talmor, and S.-H. Teng. Delaunay tetrahedralization and parallel three dimensional mesh generation. [Online]. Available: https://www.cs.cmu.edu/~glmiller/Publications/TalmorMiTeng93.pdf
[21] M. Jancosek and T. Pajdla. (2015, September) Exploiting visibility information in surface reconstruction to preserve weakly supported surfaces. [Online]. Available: https://www.researchgate.net/publication/275064596_Exploiting_Visibility_Information_in_Surface_Reconstruction_to_Preserve_Weakly_Supported_Surfaces