Literature Review

Catherine Yim
cyim@oxy.edu
Occidental College

1 Problem Context

Photogrammetry is the process of creating a 3D model out of a series of photos. This process has many applications such as city mapping, terrain model mapping, robotics, digital art and design, film and entertainment, and medicine just to name a few [6]. As computers and other technologies become more advanced and readily available, the technological gap between the latest findings in research and the implementation of these results in manufactured products continue to grow. This is due to the fact that innovations are likely created by research organizations, and the application of those new developments are used by the institutions that create them [19]. This leads to the similar solutions being discovered by different institutions. Therefore, the field seems to be advancing at a slower pace than it could be.

This project in specific is aimed to provide an accessible method for the user to create a small-scaled 3D model for personal use. Since there are few works available on the comparison of these different approaches [8], the goal of this project is to experiment with photogrammetry approaches and see how they compare to each other.

2 Technical Background

2.1 Understanding Homography

Computer vision is a term referring to the field that deals with information extraction from digital images [3]. In other words, it studies how a computer or system comprehends a photo. With that being said, homogeneous coordinates are useful when the computer or system is trying to make sense of 2D images (with 2 dimensional coordinates) in the context of a 3D world. Homogeneous coordinates are a coordinate system for projective spaces, and they allow us to map a point in the 3D world onto a 2D image and vise versa. For example, if there is a 2D point represented in Euclidean form $\begin{bmatrix}4\\8\end{bmatrix}$, this point can be expressed in a homogeneous form $\begin{bmatrix}4\\8\\1\end{bmatrix}$, with the addition of the z-coordinate as 1. This coordinate system is useful when mapping between two planar projections of an image.

2.2 Camera Matrix

In computer vision, a camera projection matrix is a 3 x 4 matrix which describes the mapping from 3D points in the world to 2D points in an image.

$$x = CX$$

where X is a 4 dimensional vector that represents a 3D point in a homogeneous coordinate system, and x is a 3 dimensional vector that represents the corresponding 2D point in an image. In this instance, C is the camera projection matrix. The camera projection matrix is mathematically represented as the following:

$$z_c \begin{bmatrix}u\\v\\1\end{bmatrix} = K \begin{bmatrix}R & T\end{bmatrix} \begin{bmatrix}x_w\\y_w\\z_w\\1\end{bmatrix} = M \begin{bmatrix}x_w\\y_w\\z_w\\1\end{bmatrix} \quad (1)$$

where a 2D point position in pixel coordinates is represented by $\begin{bmatrix}u & v & 1\end{bmatrix}^T$ and a 3D point position in world coordinates is represented by $\begin{bmatrix}x_w & y_w & z_w & 1\end{bmatrix}^T$. u, v are the x and y coordinates of the pixel in the camera, K is the intrinsic matrix (which I will talk more about in the next subsection), and KT form the extrinsic matrix (which will also be covered more in depth later in this paper). z_c represents the z-coordinate value of the camera or view-port's position. According to this model, if the matrix product were to be divided by this value, the user could then determine the value for the pixel coordinates.

2.3 Camera Parameters

The camera projection matrix is derived from the intrinsic and extrinsic parameters of the camera, and is often represented by the series of transformations. The extrinsic parameters define the camera's position and orientation. With this information, we can map the camera's position in the world coordinate system to the camera's coordinate system. The intrinsic parameters represents the camera image format, meaning it holds information on the camera's focal

length, pixel size, and image origin. For the purpose of this paper, I will refer to the joint calculation of extrinsic and intrinsic parameters under the umbrella term of Camera Calibration.

Intrinsic parameters of a camera is typically denoted by term K and is represented as such:

$$K = \begin{bmatrix} \alpha_x & \gamma & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2}$$

where α_x α_y represent focal length in terms of pixels, γ represents the skew coefficient between the x and the y axis, and $u0$ and $v0$ represent the x and y values (respectively) of the principal point. Extrinsic parameters of a camera is typically denoted by terms R, T and is represented as such:

$$\begin{bmatrix} R_{3x3} & T_{3x1} \\ 0_{1x3} & 1 \end{bmatrix}_{4x4} \tag{3}$$

R, T denotes the coordinate system transformations from real world coordinates to 3D camera coordinates. Specifically, T maps the position of the origin of the world coordinate system into coordinates of the camera-centered coordinate system. R is a standard rotation matrix that is commonly introduced in linear algebra as a transformation matrix used to rotate Euclidean matrices:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{4}$$

2.4 Image Matching

When converting a series of 2D images into a 3D model, typically, the first step is to image match each sequential photo. Image matching simply means to find where the photos overlap and how they relate to each other. The first step to image matching is detecting points in the image that appear to have unique features. How the algorithm does this is by examining the images' pixel properties (e.g. color or placement). These unique points are often called key points or feature points. Once the key points are detected, the next step is to gather more information on that specific pixel by computing its local descriptors. Local descriptors provide more information on the pixel by examining its surroundings pixels and their properties. A local descriptor is represented by a 1-dimensional vector that describes the visual appearance of the point. This gives the computer more information regarding the pixel's location in the context of the whole photo. Upon doing so, the algorithm can then match points from two images by comparing key points using these descriptors. To do this, we iteratively compare the descriptors of the images to discover pairs of descriptors that are similar [7].

3 Prior Work

The typical photogrammetry algorithm is comprised of 5 components: camera calibration, image reverse-distortion, key point identification, image matching, and triangulation. There have been many different methods of performing each task that have been invented throughout the years.

3.1 Calibration Techniques

There are three very popular calibration techniques used in computer vision: Zhang's method, Thai's Algorithm, and Heikkila Silven's method, all of which use an object of known dimensions as the base of calibration (most commonly a checkerboard) [16].

3.1.1 Tsai's Method

Tsai's calibration model was proposed in 1987 and assumes that some parameters of the camera are provided by the manufacturer [16]. This is advantageous because it reduces the amount of estimations made during this process. This two-step algorithm requires at least 8 feature points per image and uses a set of at least 8 linear equations based on the radial alignment constraint, in the first part of the algorithm. The second portion of the process uses an order radial distortion model while no decentering distortion terms are considered. However useful this algorithm may seem, it can only be used to calibrate a camera if a grid point coordinate is already known.

3.1.2 Heikkila Silven's method

Heikkila Silven's method was developed in 1997 [16]. It uses a closed-form solution (the Direct Linear Transformation) to gather information regarding the camera's parameters. It then applies a nonlinear least-squares estimation employing the Levenberg-Marquardt algorithm to compute for distortion parameters. Unlike Tsai's method, this one uses two coefficients for both radial and decentering distortion.

3.1.3 Zhang's Method

Zhang's method requires the user to take at least 5 images of a checkerboard of known dimensions. The checkerboard must be planar, meaning it must be 2-dimensional to discard the z-coordinate variable. The checkerboard must be rotated or translated in the camera space after every photo. Zhang's method computes the projective transformation between the two images by identifying the corner points of the board. With that projective transformation matrix, this algorithm finds the intrinsic and extrinsic parameters of the camera using a closed-form solution (similar to Heikkila Silven's

method), and the rest of the unknown distortion terms found using a linear least-squares solution.

3.2 Key Point Detection Methods

Key point detection techniques are ideally fast and accurate when dealing with image transformations (e.g. rotation, lighting, and noise). Some of the most well-known and used key point detectors are Scale Invariant Feature Transform (SIFT), Speed up Robust Feature (SURF), Binary Robust Independent Elementary Features (BRIEF), and Oriented FAST and Rotated BRIEF (ORB). SIFT was created by David G. Lowe in 2004 [12]. It was the first of the 4 detection methods to be introduced, and is still widely used due to its efficient performance when handling object recognition. However, the downside of this method is that it requires a large computational complexity [7]. SURF is a popular alternative to SIFT. It was developed by Herbert Bay et al. in 2006 as a solution to SIFT's computational complexity issue [1]. Another alternative to SIFT was introduced by Calonder et al. under the name Binary Robust Independent Elementary Features (BRIEF) [4]. ORB is another popular key point detection method developed by Rublee et al., and it was marketed as an efficient alternative to both SIFT and SURF [17]. SURF, BRIEF, and ORB are all known to perform faster than SIFT, but it is still widely used in photogrammetry algorithms. The reason for this is that, despite its disadvantageous computational complexity, SIFT performs the best when facing various image transformations, meaning its robustness is yet to be challenged [7].

3.3 Connecting Key Points

The most basic method of key point matching is the Brute-Force approach. This method aims to find key points in two images based on their descriptors by iteratively sifting through each point. Although brute force matching is currently considered the standard way of image matching, it is disadvantageous due to its long run-time. If there are an N number of key points, this process would have a run-time complexity of $O(N)^2$ since each key point is being compared to each other. For this reason, among others, other key point matching methods, such as binary space partitioning, are slowly starting to be implemented in various parts of the photogrammetry field, such as urban-spacial-pattern-detection [13].

A method of enhancing this process is the incorporation of matched-epipolar projections [15]. This add-on is the process of resampling pairs of stereo images, essentially, reorienting one of the images to match the other along the y-axis. First, a small number of key points are calculated (the most prominent ones) are matched to each other. The lines that connect these two points are called the epipolar lines. These images are then translated to have the epipolar lines run parallel with the x-axis. This makes it so that the only differences between the two images are in the x-direction. Since the search is now reduced from 2 dimensions to 1 dimension, we can now calculate more key points and match them with the Brute-Force method in a shorter time. This method is meant to be more accurate than the Brute-Force method since there is less room for error in a 1 dimensional search as opposed to a 2 dimensional one [15].

3.4 Different types of 3D models

There are many different types of 3D models and constructions. Two of the most common are point clouds and meshes.

3.4.1 Point Clouds

A point cloud is, as the name suggests, a cluster of points in a 3D space meant to represent the surface of an object. This cloud of points can consist of millions of high density points in order to represent highly complicated structures [10]. Points in a point cloud have no particular order, however, the points are related to one another. This means that any region on the cloud can be measured from the information it holds regarding its surrounding geometry [10]. This is advantageous when creating representation of real-life objects since you can get a fairly accurate approximation of all measurements for a sample measurement.

3.4.2 Meshes

3D meshes are fundamental processes in many applications of condition such as shape retrieval, deformation, and texture mapping [2]. It is effectively the process of connecting a polygonal surface into different regions by either the polygon's vertex or face. Meshes can either be geometrically constructed or semantically constructed. Geometry in this case is the objective structure of the composition. Semantic construction means that the shapes are connected with the consideration of the objects' real-world applications. For example, let's consider a wall clock. Geometrically, this structure would be a circular structure with two lines protruding from the center of the circle towards the circumference. Semantically, this structure would hold the meaning of a device that provides time-of-day. In the case of geometric construction, the algorithms are based on low level geometric information to connect the polygons, and in the case of semantic construction, the algorithms are based on higher level geometric notations [9].

4 Methods

4.1 Project Idea

For this project, I made a a basic photogrammetry algorithm. This algorithm's architecture follows the basic architecture mentioned earlier in this paper: camera calibration, image reverse-distortion, key point identification, image matching, and triangulation.

4.2 Input

4.2.1 Object Photo-taking

There are two methods of photo-taking. The user could station the object on a rotatable surface, such as a tray or turntable, with the camera in a fixed position and capture a photo of the object every θ. The user could also have the object stationed in a singular position and have the camera rotate around the object from a fixed distance from the object. For both methods, each sequential photo should be the same number of degrees apart.

The user is faced with another dilemma when deciding the distance between the object and the camera. Assuming the user has access the ideal set-up (i.e. white background, white floor, bright lighting that doesn't create a lot of harsh reflections), stationing the photo further away from the camera would be beneficial. The resulting point cloud would appear more dense, and rotation errors would be more forgivable. However, since most user would be taking the input photos at a convenient location, I would assume that the user's set-up is not ideal (i.e. lots of background noise, lighting that creates a lot of contrast). In this case, stationing the camera closer to the object would compensate for the set-up by showing less of it, but, as mentioned before, this would mean that faulty position fixtures would be more obvious in the resulting 3D model.

The algorithm is set up so that the quantity of input photos effects run-time and quality of the output. The more photos the user provides, the more accurate the 3D model will be and the longer the algorithm will run. From experimentation, I would recommend an input size of somewhere between 18-36 images. In the case of an 18 photo entry, rotate the object 20 degrees between each sequential photo, and in the case of a 36 photo entry, rotate the object 10 degrees between each sequential photo.

4.2.2 Photo-taking for Calibration

This algorithm also asks the user for a separate set of input photos for the purpose of calibrating the camera. The object in focus here should be a black and white, 6 x 9 checkerboard. This means that there should be 7 tiles or squares along the width and 10 squares along the length. The checkerboard must be on a plane, meaning the checkerboard should be a flat object. The user must also be careful not to have any bends in the checkerboard, as it would affect the algorithm's interpretation of the camera's distortion properties. There must be at least 3 calibration photos provided. Each photo should vary in either position or rotation of the checkerboard. I've noticed that the algorithm has a difficult time identifying the checkerboard in the face of significant background noise, therefore, the board must take up the majority of the camera space.

4.2.3 Camera Used

The camera to be used must have information that is provided by the manufacturer online. There are lines that must be altered within the code itself to adapt to a new set of input photos. This algorithm, specifically, requires manual inputs of the camera's sensor dimensions and focal length in order to calculate the ratio of millimeters to pixels for the specific photos.

4.3 Camera Calibration

In order to calibrate the cameras, I initially attempted to use Zhang's method of calibration since it is one of the most popular methods of camera calibration and is respected for its flexibility and robustness [5]. I also considered using Tsai's Algorithm, another popular method of calibration, since Zhang's method is known to be significantly more labor-intensive than Tsai's, based on the number of images alone. [11] What I landed on could be seen as more similar to Tsai's method of calibration. I incorporated the use of openCV chessboard functions. The functions provided by the library itself stay true to Zhang's method of camera calibration, however, in our particular model, inputs have known dimensions, such as the degrees between rotations and distance from the center of the camera to the object. This makes it possible to calculate the distance between cameras. This reduces unknown parameters, meaning the user can load these parameters into the algorithm to reduce the number of input calibration images needed. The structure of the input requirement ensures that all extrinsic parameters excluding focal length are known. The advantage of this is increased accuracy since camera calibration is the process of calculating estimated measurements. Therefore, we are able to use the openCV library because of the assumption that the user is using a simple phone camera as their choice of photo-capturing device. This drastically simplifies the photo-taking process since the only additional input needed for this step is three photos of a checkerboard in different locations and rotations on the plane.

4.4 Key Point Identification

For key points detection, I incorporated two methods into my algorithm. For the first method, I used the SIFT method that is provided by the openCV library. I thought it would be the best fit for my algorithm due to its robustness. However, I also wanted to account for the possibility that the user won't have access to a high-powered computer. In order to be more cost-effective, I've also incorporated an option to use the SURF method. This method would also be helpful when the input size is large.

4.5 Image Matching

For image matching, I incorporated two methods into my algorithm. I used the Brute-Force method that is provided by the openCV library, as well as the FLANN method. Brute-Force was used because it is a reliable algorithm, however, as mentioned before, it has a large time-complexity. Similar to the key point detection portion of my project, I also wanted to consider the user's availability.

4.6 Triangulation

In the final portion of the project, we use triangulation to calculate feature position in 3D world space using image plane difference, creating a point cloud. There are two processes that go into this step of the algorithm: projective reconstruction and Euclidean reconstruction.

4.6.1 Projective Reconstruction

As of now, we have key points and how they relate to each other. Now we project them into the 3D space. How I decided to do that was by estimating the fundamental matrix between the images using feature points. From that, I created a pair of perspective projection matrices for the images and chose one to use in the creation of a linear equation. This linear equation maps the relationship between the 3D points to corresponding image points through the projection matrix.

4.6.2 Euclidean Reconstruction

Now, we have a point cloud, however, our coordinates are still in homogeneous form. As a result, our point cloud appears a bit different from the real-world object. To solve that issue, I estimated a 3D homography that would transform the projective point cloud into Euclidean space. Similar to the last step, I derived a linear equation from the relationship between target 3D coordinate points and the projective coordinate points. The resulting 3D homography can then be used to transform our projective point cloud into Euclidean space.

4.7 Output

The final 3D model is generated in the form of a point cloud. For this, I used the PyntCloud method that is provided by the pandas library. The resulting product is a .PLY file that can be imported into various 3D modeling software such as Blender or Maya.

5 Evaluation

There are two metrics to evaluate the performance of a photogrammetry algorithm: speed and accuracy. Inaccurate models usually occur because of mishaps in the image matching process or the input process [7]. Time-complexity is affected by the structure of the image matching process and the key point detection process. For the purposes of evaluation, I attempted to take three series of photos with the same setup. The object, camera used, and distance measurements stayed the same.

5.1 Evaluating Quality

To evaluate the algorithm's performance, the three inputs were tested using different image matching methods. Originally, I had implemented the Brute-Force approach since it is considered the most standard method of image matching. To evaluate its accuracy performance, I experimented with the FLANN (Fast Library for Approximate Nearest Neighbors) library [14]. In contrast to Brute-Force, the FLANN approach only searches for key point matches among its approximate nearest neighbor searches. This means that a key point in one image is not compared to every key point in the other image making this method faster but less accurate. As an additional indicator of accurate matches, I plotted physical lines between the matching key points as a visual aid.

5.2 Evaluating Speed

To evaluate the algorithm's speed, the three inputs were tested on various key point detection methods. I replaced the SIFT method with SURF, BRIEF, and ORB to gauge whether there is a clear run-time disparity among the 4 different approaches.

6 Results and Discussion

From visual representations alone, I could gather that key points were concentrated on one region of the object. The discrepancy between the object's shape and the point cloud results from the label existing only on one side of the object, leading to a condensing of key points in a certain area. Substituting the Brute-Force method with the FLANN approach

showed little to no difference in the final model. However, the final models of the different inputs looked drastically different. From this, I concluded that the integrity of the 3D structure was faulty, not due to the methods used in the algorithm, but the photo-taking set-up itself. Upon examining the visual lines that connect the key points, it is clear that the algorithm struggles to account for various background noise and transformations, largely due to the set-up. As for key point detection, SIFT proved to be marginally slower than SURF, BRIEF, and ORB, however, in the context of this particular project, the difference in run-time will not span past a few seconds, therefore the time difference doesn't pose a big problem. An additional notice was that SIFT models were noticeably more consistent than other models. Therefore, I have concluded that SIFT is the better key point detector for this project. As a side experiment, I also looked a little bit into image matching using epipolar lines to visually evaluate whether the matches are more accurate. For this experiment, converted the image into a gray-scaled one, meaning the value of each pixel's color value is taken away. This leaves the pixel with only information pertaining to its intensity value or light exposure. The purpose of doing this was to account for various noises such as reflection and background color variations. While there are still outlying key points being detected, fewer faulty matches are made with this method.

6.1 Ways to Improve the Algorithm

In order to improve my program, I have come to the conclusion that I must make it more robust in the face of various transformations. There are a few ways that I would go about this. First, I would work on improving the matching precision of scale-invariant feature transform (SIFT) by modifying it. is evaluated and improved in this paper. There are many variants of SIFT that are available for use, currently. Among those variants are GSIFT, CSIFT, ASIFT, PCA-SIFT, and our old friend SURF [21]. SURF is known to perform the worst out of all of these variants, however, it is generally the fastest in all circumstances. What I'm looking for is improved precision in the face of all transformations. When evaluated against different sets of images, each algorithm proves to have its advantage [21]. Some perform the best under transformations such as scale and rotation changes and others in the face of blur and illumination changes. SURF performs the worst in different situations, but runs the fastest. PCA-SIFT seems to be the most consistent variant, performing second in different situations [21]. Another possible way to improve my algorithm is by incorporating parts of what I learned while experimenting with epipolar lines. The gray-scaling of an image can account for noise, reducing the number of key points that don't lie on the central object. In addition, color can appear differently in certain lighting exposures, it would also eliminate the possibility of mismatches by color. I would also incorporate the epipolar lines themselves. As stated before, the reimaging of the photos by using these lines would shorten run-time as well as increase accuracy. An additional advantage to this method is that we can skip the camera calibration portion of the program. When the camera is calibrated, a 3D structure can be recovered from two images by choosing an arbitrary coordinate system as a world coordinate system. The projective reconstruction, in an uncalibrated case, is very similar to a camera calibration 3D reconstruction. The difference is that we need to compute the camera projection matrices from the fundamental matrix F concerning a projective space. One way of doing so is by selecting four pairs of points and mapping them to their corresponding points in the next image. As long as these points do not lie on the same plane, they are eligible to be used to compute the camera projection matrices [22].

7 Ethical Considerations

The dangers posed by advanced photogrammetry follow suit of the threats posed by most technology advances and are mainly focused on privacy and security. Photogrammetry algorithms are very similar to remote sensing algorithms due to the nature of information gathering from images. Because of this similarity, aerial photogrammetry is often considered a subset of remote sensing, and the concerns surrounding remote sensing often extend to photogrammetry. As photogrammetry is used to create 3D models of 2D images, there are concerns about what models it can be used to recreate. As it is legal to photograph anything in public view, it is certainly possible to take photographs of federal buildings, businesses, or residential property. Photogrammetry could be used to convert these photos into a 3D map that can be inspected and analyzed for weaknesses and security shortcomings. This may lead to the safety of government buildings, banks, businesses, and residents being compromised. Due to its efficacy and intrusiveness, privacy concerns surrounding remote sensing technologies have been a topic of debate for years, however, the issue has been glossed over due to technological limitations [20]. However, the field of computer vision and 3D construction is expanding and advancing in such a way that may change this nonchalant sentiment surrounding these technologies. Until recently, good applications of photogrammetry and remote sensing, such as map making and protecting human health and natural resources seemed like a reasonable trade-off for possible privacy intrusion [20].

Another possible ethical concern would be its inaccessibility. Computers, laptops, and other hardware that can run a program or game can be prohibitively expensive. According to the United States Census Bureau, 48 percent of all

households have "high connectivity." [18] Meaning nearly half of the households accounted for by the Census had a laptop or desktop computer, a smartphone, a tablet, and a broadband Internet connection. "High connectivity ranged from 80 percent of households with an income of $150,000 or more, to 21 percent of households with an income under $25,000." [18] More specifically, 81 percent of households had a broadband Internet subscription, and 77 percent of households had a desktop or laptop computer. [18] This inversely means that 23 percent of the United States population does not have access to the Internet through a computer, and there is a strong correlation between internet/technology access and income/wealth. This makes it, at the least, extremely inconvenient for lower-income households to access computer education games designed for desktops. My particular program requires the user to have a bit of prior knowledge of programming and a computing program powerful enough to run code. Although a sub-mission of this project was to make an accessible photogrammetry tool, a computer-run program cannot be fully accessible to the general public.

References

[1] Herbert Bay et al. "Speeded-up robust features (SURF)". In: *Computer vision and image understanding* 110.3 (2008), pp. 346–359.

[2] Halim Benhabiles et al. "A comparative study of existing metrics for 3D-mesh segmentation evaluation". In: *The Visual Computer* 26.12 (2010), pp. 1451–1466.

[3] Jules Bloomenthal and Jon Rokne. "Homogeneous coordinates". In: *The Visual Computer* 11.1 (1994), pp. 15–26.

[4] Michael Calonder et al. "Brief: Binary robust independent elementary features". In: *European conference on computer vision*. Springer. 2010, pp. 778–792.

[5] Xiuxia Feng et al. "The comparison of camera calibration methods based on structured-light measurement". In: *2008 Congress on Image and Signal Processing*. Vol. 2. IEEE. 2008, pp. 155–160.

[6] Amila Jakubović and Jasmin Velagić. "Image feature matching and object detection using brute-force matchers". In: *2018 International Symposium ELMAR*. IEEE. 2018, pp. 83–86.

[7] N Jayanthi and S Indu. "Comparison of image matching techniques". In: *International journal of latest trends in engineering and technology* 7.3 (2016), pp. 396–401.

[8] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. "Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images". In: *arXiv preprint arXiv:1710.02726* (2017).

[9] Guillaume Lavoué, Florent Dupont, and Atilla Baskurt. "A new CAD mesh segmentation method, based on curvature tensor analysis". In: *Computer-Aided Design* 37.10 (2005), pp. 975–987.

[10] Franz Leberl et al. "Point clouds". In: *Photogrammetric Engineering & Remote Sensing* 76.10 (2010), pp. 1123–1134.

[11] Wei Li et al. "A practical comparison between Zhang's and Tsai's calibration approaches". In: *Proceedings of the 29th International Conference on Image and Vision Computing New Zealand*. 2014, pp. 166–171.

[12] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[13] Fraser Morgan and David O'Sullivan. "Using binary space partitioning to generate urban spatial patterns". In: *4th International Conference on Computers in Urban Planning and Urban Management*. 2009, pp. 1–16.

[14] Marius Muja and David Lowe. "Flann-fast library for approximate nearest neighbors user manual". In: *Computer Science Department, University of British Columbia, Vancouver, BC, Canada* 5 (2009).

[15] Demetrios V Papadimitriou and Tim J Dennis. "Epipolar line estimation and rectification for stereo image pairs". In: *IEEE transactions on image processing* 5.4 (1996), pp. 672–676.

[16] Fabio Remondino and Clive Fraser. "Digital camera calibration methods: considerations and comparisons". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 36.5 (2006), pp. 266–272.

[17] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International conference on computer vision*. Ieee. 2011, pp. 2564–2571.

[18] Camille L Ryan and Jamie M Lewis. *Computer and internet use in the United States: 2016*. US Department of Commerce, Economics and Statistics Administration, US . . ., 2017.

[19] Toni Schenk. "Introduction to photogrammetry". In: *The Ohio State University, Columbus* 106 (2005).

[20] E Terrence Slonecker, Denice M Shaw, and Thomas M Lillesand. “Emerging legal and ethical issues in advanced remote sensing technology”. In: *Photogrammetric engineering and remote sensing* 64.6 (1998), pp. 589–595.

[21] Jian Wu et al. “A Comparative Study of SIFT and its Variants.” In: *Measurement science review* 13.3 (2013).

[22] Zhengyou Zhang. “Determining the epipolar geometry and its uncertainty: A review”. In: *International journal of computer vision* 27.2 (1998), pp. 161–195.