Development of 3D Feature Detection and on Board Mapping Algorithm from Video Camera for Navigation

Eng Zi Hao and Sutthiphong Srigrarom*

University of Glasgow in Partnership with Singapore Institute of Technology, 139660, Singapore

Abstract

In this paper, a different approach is introduced to produce comparable 3D reconstruction outcomes similar to that of working geometry method but not as computationally extensive as well as mathematically complex. An image pair, capturing the left and right view of the object or surrounding, is used as inputs. The analogy is very similar to how the human eye perceives the world. The 3D reconstruction program is broken down into two sections, with 3 MATLAB codes been written in total. First, to generate the image frames, followed by the second section, generating the 3D point cloud. In the first part of the program, 2 MATLAB codes have been written with the end result of estimated image frames between the two views which are not captured by the camera will be generated. In the second half of the program, the image pair is now processed to generate 3D point clouds containing 3D co-ordinates of the features. This techniques allows the partial reconstruction of a 3D environment by stitching together these image frames, thus creating a video of the environment as if the camera is moving from the left camera point to the right, giving the user the depth perception one would get when viewing it in real life. After which a 3D point cloud is generated, however to achieve this, the camera must first be calibrated to obtain the camera parameter with the aid of a checkerboard. The camera positions are also estimated and this is combined with the 3D co-ordinates of the features, producing the 3D point cloud. This will give the 3D co-ordinates of the features in an interactive 3D plot within MATLAB extracted from just a pair of input images.

Key Words: 3D Point Cloud, Features Detection, Vision-based Navigation, MATLAB

1. Introduction

In recent years, unmanned aerial vehicles (UAVs) have received an enormous amount of attention from educational institutes as well as the industry. For small scale UAVs, it is common to see light-weight and low-cost inertial measurement units (IMUs) being utilized for navigation as they are light and low cost. However, these low-cost IMUs are often affected by high measurement noises resulting in large measurement biases causing the UAVs to drifts rapidly. [1] To achieve a drift-free state, inertial navigation is commonly accompanied by the use of global positioning system (GPS). However, due to variety of reasons such as deliberate jamming or environment obstructions causes intermittent or a lack of GPS signal posing a major problem for safe navigation. Hence, there is need for much research to be done finding alternative navigating means to cope with GPS outages.

Computer vision systems has proven to be a compatible and successful application to a variety of UAV navigation task. Exteroceptive sensors, flawless estimation algorithms, proficient on-board computers, have been equipped to the UAV to achieve an accurate perception of the environment. In order to ensure the data received
by the external sensors are reliable, the measurements from the sensors must be processed keeping in mind the relationship between the motion of the UAV and the surrounding environment [2].

While images are characteristically in a two-dimensional (2D) array, the world is made out scenes where there are volume and depth, and entities are positioned blocking one another giving a third dimension to the viewer. Computer Vision in the past 50 years, has made substantial developments in the 3D reconstruction community. With the principal research being concentrated on 3D reconstruction in the past two decades, depth maps [3,4,8] or 3D point clouds [5–7,9] are now being produced with higher accuracy. The computation of a fairly accurate sparse 3D model as well as dense 3D surface models of an object from substantial amount of overlying views of the particular object at different angles and viewpoints has been made possible with techniques [5,6] such as feature detection and key point matching [3,4,7–10]. Furthermore, with the usage of external motion sensing input device such Microsoft Kinect, Primesense or Asus Xtion, reliable information such as a depth map can be computed conveniently and effectively out of box.

In this paper, a different approach is introduced to produce comparable 3D reconstruction outcomes similar to that of workive geometry method and is not as computationally extensive as well as mathematically complex. The feature detection is similar in terms of descriptors to those from SIFT (scale invariant feature transform) method. When compare to other 3D mapping scheme, this current detect smaller number of ‘features’ than those from direct method, hence, lesser memory use, with reasonable performance accuracy [29,30]. After which a 3D point cloud is generated, however to achieve this, the camera must first be calibrated to obtain the camera parameter with the aid of a checkerboard. The images used to generate the 3D point cloud must also have the checkerboard on surface in which the depth of the features can be calculated and therefore, producing the 3D point cloud.

The objective of this work is to develop an algorithm that can convert 2D planes into 3D geometry using input from a given source in the form of images, or a pre-recorded video. The image pair captures a perspective of the same scene but from slightly different angles, followed by using feature detection and matching in MATLAB to detect the unique features of the surrounding environment and object in the images. The incoherence in the input images, often indicates that the surrounding environment has discontinuities in depth, a change in property in the material, varying alignments in the surface or an inconsistent illumination of the surroundings. Depth information can then be extracted based on the horizontal translations of the matched features between the image pair, or with the aid of a checkerboard, in which the distance of the matched features to the camera can be determined, giving the 3D co-ordinates of the feature.

The usage and fluency of image analysis techniques will heavily influence the progression and success of this work. The various stages for the progression of the 3D reconstruction program has been mentioned above with the core component of the work being feature detection and matching of entities and environment in pair of images. The flow chart below shown in Figure 2 illustrates the process in which image analysis program is developed. 7 major phases form the back bone of this work and will be further elaborated in greater detail in the following section in the paper.

### 2. Required Hardware and Software

#### 2.1 Hardware

**2.1.1 Camera**

Before getting to the program, the image pair must

![Feature-Based Direct](image_url)

**Figure 1.** Comparison of feature-based and direct 3D reconstruction scheme [29].
first be acquired and either a digital camera or an analogue camera can be used for this purpose. Digital cameras are able to capture images of various resolutions and also capable of direct interaction and data transmission with computers. In contrast, analogue cameras require supplementary suitable data reading and conversion cards for data communication between the device and the computer. In addition, digital cameras produce higher quality images with low noise whereas analogue cameras are less sensitive, producing poorer quality images. Hence, a standard digital camera makes a better device to use as the hardware for this work. Any digital imaging devices can be used, including cellphones or even a webcam, making acquiring images for the program very easy. The main consideration would be the resolution of the image, better resolutions would give the program more pixels to process, in turn, improving the feature detection as well as feature matching. However, with a higher resolution photo, the program will need a longer processing time as it tediously analyses the larger amount of pixels. An experiment was conducted using a DSLR camera, however, the image files captured is very large and required a long processing time hence had to be reduced in size. This defeated the purpose of using a high quality camera hence was taken out of consideration.

The particular device chosen is a Samsung Galaxy Tab S. Its excellent rear camera with 8 megapixels produces images of resolution up to 3264 × 1836. This gives the program more than adequate amount of pixels to perform image processing on the acquired image.

2.1.2 Checkerboard

Checkerboard patterns are useful in numerous computer vision applications. The alternating black and white grids create distinct grid corners features which are very strong points to detect and extract. These intersections of every four alternating black and white regions, marked by circles in Figure 3, are therefore, main features used in countless applications involving checkerboards [18].

Given an image pair with checkerboard patterns surfacing in both images, the detection and extractions of these grid corner point, enables the mapping between those scenes depicted. The correlation between the two images consists of numerous homographies between the estimated planar quadrangle regions of the checkerboard patterns in the two images. When both camera views captures the image pair with the stationary checkerboard pattern on the object, it is then possible to reconstruct the scene, which in this work, using the 3D point cloud reconstruction technique. Furthermore, if one of the image has captured a view of the planar checkerboard pattern, this can be used along with the original checkerboard pattern image, for camera calibration, which also is employed later on in the work.

2.2 Software

MATLAB [19] is a high performing technical computing language that integrates an environment for visualization, numerical computation, and of course programming. Furthermore, it is capable of analyzing sophisticated structures of data, develop algorithms with in-built debugging tools, and allows object-oriented pro-
gramming such as creating 2D and 3D models and other applications. Compared to spreadsheets or conventional programming languages such as C, C++ and JAVA, MATLAB is able to solve technical problems, explore various approaches and compute the solution faster as there is no need to execute low-level organizational tasks for example, declaring variables, establishing data types, and designating memory. It has a simple programming language, in-built math functions powerful in-built tools enabling a wide spectrum of computational abilities. The programming language are the most “human-like” and simple graphics commands makes visualizing results immediate and straightforward. MATLAB software packages have been available commercially since 1984 and has now become a frequently used tool by over a million scientists and engineers in the industry and also in most universities worldwide. Support packages known as toolboxes are available for more specific applications. There are various toolboxes for control systems, signal processing, computational finance, symbolic computation, simulation, computational biology, image and video processing, just to name a few.

2.2.1 Image Processing Toolbox

Image Processing Toolbox offers a broad array of algorithms and functions for reference, as well as apps designed to process, analyze, and visualize images, and also algorithm development. With the algorithm and apps provided, techniques such as image registration, image analysis, image enhancement, image segmentation, geometric transformations, and noise reduction are made possible. Many functions in the toolboxes recognizes various graphics processing units (GPU), multicore processors, and C-code generation [20].

Image Processing Toolbox is able to support an assorted array of image types, such as images in the high dynamic range, resolution of giga-pixels, tomographic, and embedded ICC profiles. The various functions and apps allows the visualization of images and videos, examination of a neighborhood of pixels, adjustments of color and contrast, contours or histograms generation, and manipulation of regions of interest (ROIs).

2.2.2 Computer Vision System Toolbox

Computer Vision System Toolbox offers a selection of algorithms, apps, and functions for designing and simulating computer vision as well as video processing systems. These algorithmic approaches allow the detection and tracking of an object, detection, extraction and matching of its feature, camera calibration, stereo vision, and motion detection. It also offers tools and apps for processing videos, including the input and output of video files video display, graphics drawing, object recognition, and compositing. These algorithms are available in the form of MATLAB functions, System objects, and also Simulink blocks. The Computer Vision System Toolbox system toolbox recognizes fixed-point arithmetic and generates C-code automatically, which accommodates embedded system designing and rapid prototyping [21].

3. Software Development

3.1 Software Overview

The software will read a pair of images captured digitally and subsequently convert the image to the necessary format in order to detect, extract and match features, to generate estimated image frames between the original image pairs after which, a video can be produced by stitching the images frames together, giving the simulated 3D reconstruction. A 3D point cloud will also be generated in which the camera will have to be calibrated with the aid of a checkerboard. The image pair must also contain the checkerboard on the surface, in which the software then be able to estimate the depth of features matched therefore, the 3D point cloud can be generated.

3.2 Software Procedures

The 3D reconstruction software is broken down into two sections. First, to generate the image frames, followed by the second section, generating the 3D point cloud. In the first part of the software, 2 MATLAB codes have been written with the end result of the estimated image frames being generated. The codes will be attached in the appendix. The codes were written with ease of application in mind, with the ability to use any two image with enough matching features. It does not require camera calibration or the use of checkerboard, making it very versatile in application. The 3 codes has five crucial procedures following the input of image pair with or without checkerboard pattern:
In the second half of the software, in order to extract the 3D co-ordinates of the features, the image pair is now processed to develop a 3D point cloud. It has become apparent that the use of checkerboards and camera calibration is crucial for generating the 3D point cloud and attaining the depth information of the features and distance between the features and the camera. Using the knowledge gained from the development of the first part of the software, a single code is written to achieve this through four crucial procedures following the input of images with checkerboard pattern only:

1. reconpc_commented.m
   - Camera calibration
   - Checkerboard detection and Extraction
   - Feature Detection, Extraction and Matching
   - 3D point cloud generation

3.2.1 Feature Detection and Extraction

Initially, the features were manually inputted by the user by selecting the feature on the left image first followed by selecting the same feature in the right image, but this method was not effective as the success of this software is dependent on the amount of input as well as the accuracy. Manually selecting the inputs is a tedious and straining task for the user and although it was capable of producing partial 3D reconstructions of the image pair, for higher accuracy, an automatic solution has to be developed.

The image pair is first converted to grayscale using the syntax, rgb2gray, after which it can be used as an input as an M by N, 2D grayscale image. The feature detection function chosen is the speeded-up robust features (SURF) detection function due to its object detection along with image registration capabilities regardless of scale and rotation variations. With the syntax, detectSURFFeatures, it detects blob features, returning SURFPoints, containing data regarding the detected SURF features from the grayscale 2D input image. Figure 4 illustrates feature detection in MATLAB.

After which, each single-point SURFpoints detected earlier will specify the epicenter location of the point of focus in a square neighborhood. To extract data about these points, the descriptors must be derived based on the pixels neighboring the point of interest. The method for extracting descriptors utilized varies, depending on the method using to detect the input features. The syntax, extractFeatures, is used and it returns data about the descriptors also known as, extracted feature vectors, and their associated positions, from the binary or grayscale images.

3.2.2 Feature Matching

Upon the extraction of the two input features set obtained from the image pair, the features are now matched and returned as indices using the syntax, indexPairs = matchFeatures(features1, features2). Data of corresponding features as indices, are stored as P-by-2 matrices with P indicating the quantity of indices. The inputs, features1 and features2, represents the matched features between the image pair and correlates to individual index pairs. For example, as shown in Figure 5, the third feature in features1 matched the second feature in feature2 and is paired as the first index pair, subsequently the first feature in features1 matches the fourth feature in feature2, forming the second index pair.

The syntax, maxratio, is used as a ratio threshold within the range of 0 to 1 with the default value of 0.6, increasing this value will allow more matched to be re-
turned. Figure 6 shows how feature matching is illustrated in MATLAB. It is fairly accurate with only a 1.48% error with 2 incorrect matching out of 135 as shown. There are experiments conducted with no errors in matching illustrated later on in the report.

3.2.3 Estimation of Depth

The estimation of depth is established on the premise that the camera translates along the X axis, entities nearer to the camera are observed to have a larger difference in position between the 2 images compared to the entities further away. Figure 7 graphically represents this with the living room image as this effect is distinct with the luggage in the foreground displacing a large amount as compared to the white pillow on the couch, and finally the window panels in the background with the least displacement. This phenomenon can be observed in all images with the camera translating in a certain direction capturing images where the entities will be observed to have an offset in that same direction. The actual depth is estimated by calculating the distance of the horizontal offset of the pair of matched up features of the object between the original left and right image. A gain is then multiplied to this distance to gauge the appropriate depth with respect to the horizontal and vertical axis. Upon estimating the depth at the selected features, the depth of the neighboring pixels will follow the selected features nearest to it resulting in a depth approximation for all features in the input image.

3.2.4 Frame Generation

Once the depth points are estimated, a new surface can be generated in MATLAB with the depth data. After which, a path is simulated in a 3D environment layering a texture over the surface created earlier using the original image. The camera position can be simulated to move along this path created and estimate various views that were not captured in between the two original images. This technique allows the partial reconstruction of a 3D environment by stitching together these image frames, thus creating a video of the environment, giving the user the depth perception one would get when viewing it in real life.

3.2.5 Camera Calibration

The camera calibrator app is utilized to approximate the lens distortion, camera intrinsics and extrinsics parameters. These camera parameters are crucial in the application of various computer vision systems such as measuring planar entities, reduction of lens distortion effects from images, and of course in relation to this work, the reconstruction of 3-D scenes from numerous views. A checkerboard is used as the calibration target for the rea-
sons mentioned in section 3.1.2. The checkerboard pattern used for this work was obtained from MATLAB and printed. In order for the app to determine the checkerboard’s orientation, the pattern must comprise of an even number of blocks and an odd number of blocks on the other side. This creates a pattern containing two black blocks on the corners of one side and two white blocks on the corners of other side and the x-direction is assigned to the longer side.

With the checkerboard ready, the camera to be calibrated must now capture at least 10 to 20 different images of the checkerboard pattern. In these work, 39 images were used for calibration taken from the Samsung Galaxy Tab S. 4 of the images were rejected leaving 35 images to be used for calibration as shown in the figure below.

For the best results, it is important to note the following [22]:

- The checkerboard pattern should minimally fill 20% of the image frame.
- The checkerboard pattern should be captured by the camera at various orientations.
- The checkerboard should be at no more than 45 degrees angle comparative to the camera’s plane.
- For calculating lens distortion, the checkerboard must be captured near the edge of the frame. Lens distortion begins growing from the image center radially and is often not increasing uniformly across the frame of the image as shown in Figure 9.

The distorted features are represented as \((x_{\text{distorted}}, y_{\text{distorted}})\) [23]:

\[
\begin{align*}
    x_{\text{distorted}} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\
    y_{\text{distorted}} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)
\end{align*}
\]

where

- \(x, y\) represent the locations of undistorted pixels
- \(k_1, k_2\) and \(k_3\) are the coefficients of lens radial distortion
- \(r^2 = x^2 + y^2\)

Figure 7. Horizontal offsets of objects at various distance to the camera.

Figure 8. 35 images used for calibration.
Usually, two coefficients will suffice for camera calibration however, for severe distortion, like that obtained when using wide-angle lenses, 3 coefficients should be used which includes \( k_3 \). Once uploaded into the app, the calibration process starts by analyzing the various images, skipping duplicate images and rejecting images that cannot be processed. The camera calibration algorithm adopts the pinhole camera model:

\[
[w \ x \ y \ 1] = [X \ Y \ Z] \begin{bmatrix} R & t \end{bmatrix} K
\]

where

- \((X, Y, Z)\): World coordinates of a feature
- \((x, y)\): Image coordinates of the co-relating image feature in pixels
- \(w\): Arbitrary homogeneous coordinates scale factor
- \(R\): Rotation of the camera in 3D in Matrix form
- \(t\): Camera translation on relation to the world coordinates
- \(K\): Camera intrinsic matrix

The camera intrinsic matrix is defined as [24],

\[
\begin{bmatrix}
  f_x & 0 & 0 \\
  s & f_y & 0 \\
  c_x & c_y & 1
\end{bmatrix}
\]

where

\[ f_x = F \times S_x \]
\[ f_y = F \times S_y \]
both of which are expressed in pixels

The coordinates, \([c_x, c_y]\), represents the optical principal point in pixels. When the vertical and horizontal axis align perpendicularly, the skew parameter, \(S\), is 0. \(F\) represents the focal length in millimeters and \([S_x, S_y]\) are the number of pixels per millimeter in the vertical and horizontal direction respectively. Reworkion errors can be calculated as shown in Figure 10. It shows four images with adversely high errors for a more accurate calibration, the four images were removed lowering the overall mean error from 2.11 pixels to 1.45 pixels.

The camera extrinsics can be visualized through MATLAB, in two forms, the pattern centric view as shown in Figure 11 on the left and camera centric view on the right.

Finally, “export camera parameters” will generate `cameraParameters` in MATLAB’s workspace containing intrinsic, extrinsic parameters, and distortion coefficients of the camera.

### 3.2.6 Checkerboard Detection and Extraction

The checkerboard not only assists in the calibration of the camera, it is also vital on the process of partial 3D reconstruction. The images used to generate the 3D point cloud must also have the checkerboard on surface in which the depth of the features can be calculated using the checkerboard as the origins. When the checkerboard is within the image frame captured, the syntax, \([\text{rotationMatrix}, \text{translationVector}] = \text{extrinsics}\)
\((imagePoints, worldPoints, cameraParams)\) can be used to compute the location of the camera.

\(imagePoints\) is the input of Image coordinates of features, in an M-by-2 array containing M amount of \([x,y]\) coordinates whereas \(worldPoints\) inputs world coordinates co-relating to the image coordinates, and can be represented by either M-by-2 or M-by-3 matrix. \(cameraParams\) inputs the camera parameter obtained earlier through the calibration process.

The outputs of \(rotationMatrix\) and \(translationVector\), are represented by a 3-by-3 matrix and a 1-by-3 matrix respectively [25]. The equation is as such

\[xyz = [XYZ]R + t\]  \hspace{1cm} (4)

where

- \([x y z]\) are the camera coordinates
- \([XYZ]\) are the checkerboard points coordinates
- \(R\) is the rotation matrix
- \(t\) is the translation vector.

Finally using the syntax [26],

\[camMatrix = cameraMatrix(cameraParams, rotationMatrix, translationVector)\], it returns a 4-by-3 camera workion matrix in homogenous coordinates containing 3-D world points which works onto the image. The \(cameraMatrix\) function computes with the following formulae,

\[camMatrix = [rotationMatrix; translationVector] \times K\]  \hspace{1cm} (5)

where, \(K\) is the intrinsic matrix.

After which, with the camera matrix obtained, a world point can be worked onto the image:

\[xyz = [XYZ] \times camMatrix = [XY 1] \times s\]  \hspace{1cm} (6)

where, \(s\) is an arbitrary scale factor.

With the camera’s workion matrix calculated, the camera’s 2D space co-ordinates can be worked into 3D, provided that the features are detected in both images. Therefore, the location of the camera and the data of the matched points in relation to the camera and checkerboard is made known, the reconstruction in 3D can begin.

3.2.7 3D Point Cloud Generation

In order to approximate the 3-D co-ordinates of the matched features, the \(triangulate\) function is implemented and it is based on an algorithm called direct linear transformation (DLT).

3.2.7.1 Direct Linear Transformation (DLT) Algorithm

This algorithm fundamentally calculates the relation of \(x_i\), 2D image space co-ordinates and \(X_i\), 3D object space co-ordinates. Provided with an adequate amount data of the matching features between the two images, the algorithm can relate the linear mappings of any two sets of feature’s data [27]. The simplest form of DLT algorithm is described where a set of four feature correspondences between \(x_i\) and \(x'_i\), corresponding co-ordinates
in the other image, is required and described by the equation $x'_i = Hx_i$.

$$Hx_i = \begin{pmatrix} h^T_{1x} & x_i \\ h^T_{2x} & x_i \\ h^T_{3x} & x_i \end{pmatrix}$$ (7)

where, $h^nT$ is the $n$th row of $H$.

In order to simply calculate a linear solution to $H$, the above equation is to be expressed as a vector cross product, $x'_i \times Hx_i = 0$,

$$x'_i \times Hx_i = \begin{pmatrix} y'_i h^T_{1x} - w'_i h^T_{3x} \\ w'_i h^T_{1x} - x'_i h^T_{3x} \\ x'_i h^T_{2x} - y'_i h^T_{1x} \end{pmatrix}$$ (8)

where $x'_i$ is $(x'_1, y'_1, w'_1)^T$.

A set of equations for $H$ when $n = 1, 2, 3$, can be written with the unknown, $h$, linear in the following 3 equations, where, $h^nT = X^n^T h$.

$$\begin{bmatrix} 0^T & -w'_i x'_i \\ w'_i x'_i & 0^T & -x'_i x'_i \\ -y'_i x'_i & x'_i x'_i & 0^T \end{bmatrix} \begin{bmatrix} h' \\ h' \\ h' \end{bmatrix} = 0$$ (9)

When taking into consideration all four feature correspondences and expressing it in the same form, the results will be another set of equations described as, $Ah = 0$, with $A_i$, being a $3 \times 9$ matrix and $h$ being 9 vectors consisting of the matrix $H$ entries.

While implementing DLT algorithm and solving for $H$, it is common to disregard the third equation as only two of equations are linearly independent although each set of feature correspondences leads to three equations. Thus, the equations are then reduced to:

$$\begin{bmatrix} 0^T & -w'_1 x'_1 \\ w'_1 x'_1 & 0^T & -x'_1 x'_1 \end{bmatrix} \begin{bmatrix} h' \\ h' \end{bmatrix} = 0$$ (10)

The equation $A_i h = 0$, where $A_i$ has become a $2 \times 9$ matrix, remains unchanged and applies for all homogeneous co-ordinate representation of feature correspondences involved. Each feature correspondence brings about two independent equations for $H$ and with four feature correspondences in total, this gives the set of equations of $Ah = 0$ with $A$ being formed with equation coefficients based from the $A_i$ matrix.

Solving for $h$ requires a singular value decomposition (SVD) of $A$. This calculates the linear transformation between $x_i$ and $x'_i$, by taking the lowest singular value as the solution. In the event that there are five or more feature correspondences involved and there is noise degrading the data, which in computer vision processes is a common scenario, rather than expecting an exact solution, calculating an approximate solution should be attempted. This approximation is called an over-determined solution, and it is calculated using the same equation, $Ah = 0$. By simply stacking the extra, $n$, amount of $A_i 2 \times 9$ matrices forming a single $2n \times 9$ matrix and likewise, using SVD, the solution, $h$, can be found. $h$ is a singular unit vector with correspondence to the lowest singular value of $A$ and subsequently the over-determined solution is obtained.

### 3.2.7.2 Triangulation

The reconstruction is executed simply with a linear triangulation technique. Every feature detected from the input images can be described using, $x = PX$, $x' = P'X$, with $x$ being the camera’s 2D space co-ordinates of the feature, and $x'$ being the same feature worked onto the second camera’s space co-ordinates. $X$ denotes the third dimensional space co-ordinate that is in question. Combining the two equations gives, $AX = 0$, with $X$ being linear in this form. A cross product eliminates all homogeneous scale factors, resulting in three equations for every feature detected in both images. To illustrate, we take an example of the equation, $x \times (PX) = 0$ being derived for a feature in the left image. Working the equation out will result in the following three equations [28]:

$$x(p^{1T}X) - (p^{1T}X) = 0$$
$$y(p^{1T}X) - (p^{1T}X) = 0$$
$$x(p^{2T}X) - y(p^{2T}X) = 0$$ (11)

As mentioned earlier, to form $AX = 0$, the equations from both cameras are combined to produce the equation:

$$A = \begin{bmatrix} xp^{1T} - p^{1T} \\ yp^{1T} - p^{2T} \\ x' p^{1T} - p^{1T} \\ y' p^{1T} - p^{2T} \end{bmatrix}$$ (12)
Similarly, $A$ is calculated with the SVD method, subsequently, the value of $X$ can be estimated and finally, obtain the co-ordinates of all the feature matched in 3D with respect to the camera’s space co-ordinates calculated earlier with the workion matrix of the two cameras. A representation of the locations of these features can be visualized by plotting the 3D co-ordinate of the features, with the function “scatter3”, producing 3D point clouds representing the feature’s 3D location and even the corresponding color.

4. Results and Discussion

4.1 Image Frame Generation

The MATLAB codes `feature3.m` and `part3.m` are capable of producing partial 3D reconstruction using the static input image pairs. This part of the program can function with or without the aid of the checkerboard as shown in Figure 13 below. The success of this program relies heavily on the feature detection and matching and the MATLAB Computer Vision System Toolbox is very effective and accurate in doing so as illustrated in Figures 12 and 20. The yellow lines indicate the matching features between the 2 images as shown, allowing the users to check the matching to determine its accuracy and as seen in Figure 13, the matching of the features are perfect, with no errors. MATLAB can output the matched features in both grayscale as well as colored images depending on the user specification.

Image detection works best in environments that are...
well lit and free of shadows. Areas that are in the shadows often do not return any detected features. Objects with distinct features and patterns return a higher rate of feature detected as well as higher accuracy when matched as compared to images with repetitive patterns or similar color throughout. This is illustrated in Figure 12 where the deodorant can with distinct marking and labels produces 60 matched features detected as compared to Figure 13, the stuffed toy without distinctive features with only 28 points. However it is worthy to note that Figure 13 shows no errors out of the 60 matched points, however, Figure 12 shows at least 6 errors in matching out of just 28 points. This goes to show how heavily the accuracy of feature matching is influenced by having distinct features and good lighting.

Another factor that affects the feature matching is the camera translation. In Figure 13 the cameras translation is kept mainly along the x axis as compared to Figure 12 where the camera underwent a slight rotation as well, giving an extra degree of difficulty when matching the features.

Several scenes were experimented with different objects and environments for partial 3D reconstruction and there were instances where distortion was observed between the reconstructed frames. This is most noticeable when looking at the video or by scrolling through the image frames quickly. This is due several factors such as, some features being occluded in either one of the original image pairs, the mismatch of features or the lack of features present to create depth field in that area. The same scenarios used above will be discussed. Figure 14 shows some 10 of the image frames extracted from the 20 3D reconstruction image frames of the experiment held with the stuffed toy. These 20 frames were extracted using the function defined in videoframe.m and is reliable to execute as shown in the figures below. As expected, there exists distortion within the image frame generated as there were fewer features matched and also some errors during the matching process due to the indistinct patterns on the toy as mentioned above. Despite the distortion, the viewer is still able to get the sense of depth from the video with the points nearer to the camera such as the checkerboard or the nose translating more in the x direction as compared to the pig stool at a medium distance and of

![Figure 14. Image frames produced for the image of the stuffed toy.](image-url)
course the tail, being the furthest feature from the camera.

When compared to Figure 15, it is easy to see that there is less distortion on the deodorant can as compared to the stuff dog. This is as predicted due to the large amount of features matched and the high accuracy of matching the distinct features as shown in Figure 13. Figure 15 shows the movement of the camera from the left to right illustrating that the program is able to function in both direction of translation along the x axis. The frames start with a view slightly to the right of the deodorant with a larger gap between the deodorant and the cologne, and ends with a more central view of the deodorant with a smaller gap between the 2 objects. This give the viewer the sense of depth as mentioned before with the deodorant can translating more than the cologne as it is nearer to the camera. Noting the minimal to almost non-existing distortion on the deodorant can on all image frames. This shows the effects of having high quantity and quality features matching has on the 3D reconstruction. The only notable distortion is the tile separations which were too indistinctive to return a feature detection.

4.2 3D Point Cloud Generation

The MATLAB code, reconpc_commented.m, produced interactive 3D point clouds representing the locations of the features detected and matched from the image pair allowing the viewer to rotate and obtain a 360 degree view at any angle of the reconstructed scene. As mentioned in the previous sections, the images must contain the checkerboard for the calculation of the 3D coordinates. Starting with the same stuff toy example used above, Figure 16 shows the generated interactive 3D point cloud with the green point indicating the origin of the checkerboard. The other points are colored, based on the corresponding features, with the pink dots representing the stool and the brown dots representing the stuffed toy which can be cross referenced to Figure 12. The checkerboard is aligned with the z direction pointing into the checkerboard and the y direction is pointing towards the camera hence, the features matched which are above the checkerboard and behind the checkerboard will be negative as shown in Figure 18. The 3D point cloud is therefore displayed upside down due to the orientation of

Figure 15. Image frame produced for the image of the deodorant and cologne.
the z axis of the checkerboard, thus the left and right cameras being worked on the opposite sides. The camera position is also displayed and from Figure 17, it is shown that the left camera is approximately 20 cm to the left of the checkerboard and the right camera is approximately 15 cm to the right. From Figure 18, it can be deduced that the distance of the camera is approximately 52 to 55 cm in front of the checkerboard and the distance of the features is around 30 cm above the checkerboard which is accurate compared to the actual distance of the camera measured to be 55 cm away from the toy, giving an error of 5%.

Another experiment is performed using a helmet with the feature matching shown in Figure 19. The feature matching is once again seen to be very accurate on areas with distinct markings as shown. Likewise, the green dot represents the origin of the checkerboard and the colored points of the corresponding features. Figure 19 shows the maroon, black and grey points in the 3D point cloud, as shown in Figures 20 and 21, accurately displaying the 3D co-ordinates of the features. The camera is seen to be approximately 56 to 58 cm away from the checkerboard which is very accurate given the actual measured distance is 55 cm giving an error of less than 5%. The majority of the features at 5 cm behind and a maximum of 30 cm above the checkerboard proving the accuracy and reliance of the code.

5. Conclusions

This work presents a different approach based on feature detections is introduced to produce comparable 3D reconstruction outcomes similar to that of workive geometry method (direct detection). The feature detection is similar in terms of descriptors to those from SIFT (scale invariant feature transform) method. A few examples are presented to show the effective of this approach.
3 MATLAB codes are written for this work. First, the feature detection and extraction code, feature3.m, achieved the purpose of detecting numerous features at an accelerated speed. The image should be taken in a well-lit environment, minimizing shadows as much as possible as feature detection within the shadowed areas are low. Feature detection is most efficient for objects with distinct features as compared to objects with plain patterns or a single solid color throughout. Distinct features also make for more accurate feature matching as compared to repetitive patterns. The user can check for errors in feature matching from showing the matched features within MATLAB. The program has returned matches with no errors showing how effective it is in feature detection and matching given an object with distinct features and good lighting.

Second, with the code, part3.m, the matched features are analyzed for the horizontal displacement of the feature between the 2 images and used to approximate depth for all the features of the image. The success of this portion of the work is heavily influenced by appropriate matching of the features. Having a higher amount of matched features along with the accurate matching of the feature produces a higher quality video in which there are less distortions. The image frames are subsequently produced for views in which was not captured by the camera and are effective in giving the viewer a sense of depth and creates a partial 3D reconstruction of the scene. Along with the use of the function defined in videoframe.m, the image frames can be extracted concluding the first part of the program.

The advantage of the code is that it is capable of producing a partial 3D reconstruction without much computational, camera calibration is not necessary and there is no need to determine camera’s intrinsic or extrinsic parameters. Without the reliance on checkerboards, any 2 images containing enough features to be detected and matched will be able to be used for a partial 3D reconstruction.

Last, reconpc_commented.m is written to obtain the 3D co-ordinates of the features, generating the 3D point cloud after camera calibration and with the aid of the checkerboard. It has become apparent that the use of checkerboards and camera calibration is crucial for attaining the exact 3D co-ordinates of the features and using the knowledge gained from the development of the first part of the program, the code is written. The results are conclusive with the 3D point clouds accurately plotted with errors of only 5%. Camera locations and feature locations in the x, y and z direction are plotted in the interactive 3D point cloud allowing the viewer to rotate and obtain a 360 degree view at any angle of the reconstructed scene.

These images can be captured with only the use of one camera, for example mobile devices or digital cameras, taking the left and right views individually, making the

![Figure 20. 3D point cloud of experiment using a helmet.](image1)

![Figure 21. Y and Z axis of 3D point cloud of experiment using a helmet.](image2)
application of the program very versatile and convenient, without relying on special stereo hardware to capture the images. Both the image frames and 3D point cloud are able to produce a partial 3D reconstruction that gives the viewer a sense of depth whereas the exact 3D coordinates are obtained from the 3D point cloud.

References


Call for Papers

Tamkang University was founded in 1950 and Tamkang Journal was published since 1962. Starting from 1998, Tamkang Journal was divided into two, which are Tamkang Journal of Science and Engineering and Tamkang Journal of Humanity and Social Science. Tamkang Journal of Science and Engineering became an international journal since 2000 and four issues are published each year. Tamkang Journal of Science and Engineering has been published under the new title Journal of Applied Science and Engineering (JASE) since 2012. This peer-reviewed journal with a new-look cover will continue to be published quarterly in print and online (http://jase.tku.edu.tw).

Journal of Applied Science and Engineering (JASE) devotes to addressing the following broad research topics: Aerospace Engineering, Applied and Engineering Mathematics, Applied Physics, Architecture, Biochemistry and Biomedical Engineering, Civil Engineering, Chemical and Materials Engineering, Computer Science and Information Engineering, Electrical Engineering, Mechanical and Electromechanical Engineering, and Environmental Engineering. Each issue will focus on a particular topic as possible as we can to persistently pursue advanced research papers from all areas. Each article published in this journal is included in EI Compendex/Scopus so far. We moreover hope JASE can be recognized by and included in other well-known international databases, such as Science Citation Index Expanded and JCR in the near future. As such, we would like to sincerely invite the prestige and devoted researchers to submit their significant research results to Journal of Applied Science and Engineering (JASE) through “Scholar One” Online Submission & Peer Review Service (website: https://mc04.manuscriptcentral.com/jase). Your submission can not only significantly promote your research effort but will also superb our journal.