Display Virtual Cubes on Your Desk!
: 3D Reconstruction and AR via Stereo Vision on Videos

Jihwan Kim
2019-11028
Department of Electrical and Computer Engineering
kjh26720@snu.ac.kr

Hojune Kim
2019-19560
Department of Aerospace Engineering
hojjunekim@snu.ac.kr

Abstract

We built a simple AR technology not using deep learning approach, but only the traditional computer vision method. The ultimate goal of our project is displaying virtual cubes on the dominant plane of the arbitrary videos. In contrast to other works, it does not require any predefined spaces or objects. The only assumption is 10cm translation to right side for the first few seconds of the videos. We implemented camera calibration, 3D map initialization, dominant plane selection, 2D point tracking, camera tracking, cube projection, and several innovative error handling methods to develop significantly stable and precise AR model. It can display the cubes on the videos from not only the fast moving camera, but also the camera that are entirely out of the initial position and direction. Furthermore, it takes only 0.015 seconds per each frame to handle 1920×1080 *videos, which can be implemented in real-time operation. We release our project and public manual at* https://github.com/Jjihwan/2023S_SNU_CV_Project

1. Introduction

1.1. Augmented Reality(AR)

Augmented Reality(AR) is a technology that combines the real world and the virtual world so that they can interact between them. This differs from VR(Virtual Reality), which creates an independent virtual world, in that it displays virtual objects on the actual surrounding environment. AR can be defined as following three characteristics. [1]

1) The combination of real and virual worlds
2) Real-time interactions
3) Accurate 3D registration of virtual and real objects

All of these features should basically presuppose accurate 3D reconstruction of the surrounding environment. For example, to display a virtual object in a desired location on the real world, you need to know the 3D world coordinate in the desired location. In addition, in order to display the virtual object on the screen, it is necessary to be able to project the object into a plane by tracking the location and direction of the camera. However, the only way to get information about the surroundings are 2D images from videos. That is, it is necessary to solve the problem of reconstructing the 3D world from the 2D input images by the camera.

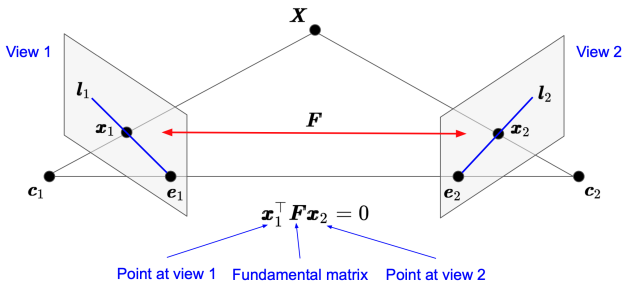

Figure 1. Epipolar geometry

1.2. objectives

Our project aims to implement AR by exploiting traditional computer vision technologies such as Stereo Vision, epipolar geometry, feature descriptor, RANSAC, and robust fitting. The ultimate goal is displaying virtual cubes on videos from naturally move and shoot over a space that is not predefined. Especially, since we pursues a model that is as stable and precise as possible, we made a lot of effort into handling various errors.

2. Literature View

2.1. Epipolar Geometry

Let the point where any 3D point X is projected from views 1 and 2 be x_1 and x_2, respectively. We cannot know the depth only from x_1 of view 1, but what we only know

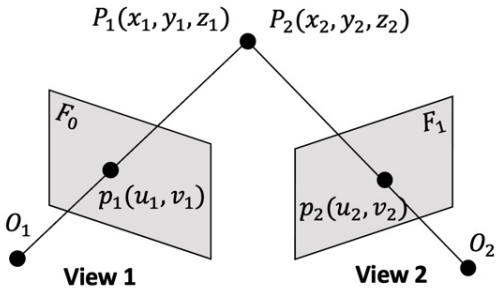

Figure 2. 3D Reconstruction

is that X is placed on the line connecting the camera center and x_1. Therefore, projecting a line into view 2 image where X can be placed is represented as one line. This is called epipolar line corresponds to l_2 in Fig.1. The point x_2 at which X is actually projected in view 2 is also placed on the epipolar line l_2.
Conversely, in view 2, It is able to draw an epipolar line in view 1. The points where the line connecting the camera centers of view 1 and view 2 intersect with each image are called epipoles, which is always placed on the epipolar line. The epipoles, epipolar lines, camera centers, and the actual 3D point X are all placed in one plane, which is called epipolar plane.

2.2. 3D Reconstruction

In this project, it should be possible to restore 3D coordinates using images taken from two different views. Therefore, we defined 3D reconstruction as "a method of restoring 3D coordinate of a specific point using images taken from two views that know the location relationship", and Epipolar geometry was used to induce the following.

If a point (u, v) on the 2d image is expressed as homogeneous coordinate, it becomes $(u, v, 1)$. It means that all (cu, cv, c) coordinates in the space are projected as (u, v) on the image. For any point P in the space, as shown in Fig. 2, let's say the coordinate from the view 1 is $P_1(x_1, y_1, z_1)$ and the coordinate of the point projected on the image plane is $p_1(u_1, v_1)$. This is equally represented by $P_2(x_2, y_2, z_2)$ and $p_2(u_2, v_2)$ in the view 2 coordinate system. If the conversion between view 1 and view 2 is defined as a rotation matrix R and a translation vector t, the following equation holds between P_1 and P_2.

$$RP_1^T + t = P_2^T \tag{1}$$

If the corresponding point pair is known from the two images, three-dimensional coordinates can be obtained using triangulation and fitting. We want to get 3D coordinate $P_1(x_1, y_1, z_1)$ as seen in the first view.

When we rearrange 1 into $p_1(u_1, v_1)$ and $p_2(u_2, v_2)$, the following two equations can be obtained. Two expressions can be obtained from the above expression.

$$[R_2|t_3]u_1 - [R_1|t_1]\begin{pmatrix} P_1 \\ 1 \end{pmatrix} = 0 \tag{2}$$

$$[R_2|t_3]u_2 - [R_2|t_2]\begin{pmatrix} P_1 \\ 1 \end{pmatrix} = 0 \tag{3}$$

Also, $P_1(x_1, y_1, z_1)$ is projected as $p_1(x_1/z_1, y_1/z_1)$ on view 1 image, so the following expression is established.

$$[1, 0, -u_1, 0]\begin{pmatrix} P_1 \\ 1 \end{pmatrix} = 0 \tag{4}$$

$$[0, 1, -v_1, 0]\begin{pmatrix} P_1 \\ 1 \end{pmatrix} = 0 \tag{5}$$

Thus, $A\binom{P_1}{1} = 0$ is established for the 4*4 matrix A, and the problem of obtaining P_1 is replaced by the problem of solving the objective function below.

$$\min ||Ax|| \tag{6}$$

If A is eigen-decomposed as $A = VDV^T$ and $y = V^T x$, it is rearranged as follows.

$$||Ax|| = x^T A^T Ax = x^T VD^2 V^T x = y^T D^2 y \tag{7}$$

Consequently, $\binom{P_1}{1}$ is the eigenvector corresponding to the smallest eigenvalue of $A^T A$

2.3. Feature Descriptor

2.3.1 SIFT [2]

SIFT is a 128-dimensional descriptor that approximates Laplacian of Gaussian as a Difference of Gaussian to increase computation speed, and uses an image pyramid that converts image sizes in various ways to detect feature points with different scales. [1] The oval blob is detected through affine transformation.

2.3.2 SURF [3]

SURF is a 64-dimensional descriptor that uses integral image to improve computation speed very quickly and maintain competitive performance. Since the amount of computation is same regardless of the filter size using integral image, filter pyramid is used by converting the size of the filter without image conversion. It uses the round blob without affine transformation.

2.3.3 ORB(FAST+BRIEF) [4]

ORB is a detector created by combining FAST detector and BRIEF detector. In the case of FAST detector, unlike SIFT or SURF, one key point has only one feature. It obtains corner by Harris corner detection, distinguishing the difference between the center pixel and the surrounding pixel. However it cannot consider any direction of the corner. BRIEF descriptor is a method of increasing computational speed by reducing memory through binarization.

2.4. PTAM [5]

PTAM is one of the visual SLAM algorithms using key frames, and is a technology that optimizes conversion between key frames to maintain the accuracy of maps containing features while performing localization and mapping. The characteristic of PTAM is that tracking the location of the camera and the mapping process of extracting and storing feature points are divided into separate and parallel threads. It tracks the location of the camera in a short period of time, and can be implemented in real-time by processing only key frames and containing high-accuracy features even if it takes a long time.

2.5. Lie Theory: SE(3) Group [6]

SE(3) Group of Lie theory refers to rotation and translation in 3D space. Since 3D motion can be represented as 4×4 matrix and the degree of freedom of the motion is 6, there must be a mapping between 6 dimensional vector and the 4×4 matrix. Let express the 6-dimensional vector as μ and the extrinsic matrix as P.

$$\mu = [t_1, t_2, t_3, w_1, w_2, w_3]^T = (\mathbf{t}, \mathbf{w}) \tag{8}$$

Then A can be represented as following:

$$P = \exp(\mu^\wedge) = \begin{bmatrix} R & Rc \\ 0^T & 1 \end{bmatrix} \tag{9}$$

The exponential mapping of 9 is defined by Lie theory with complex mathematical expressions.

2.6. Gauss-Newton Method [7]

The Gauss-Newton method is an iterative optimization algorithm used to solve nonlinear least squares problems. These problems involve finding the best-fitting parameters for a nonlinear model that minimizes the sum of squared differences between the observed data and the model predictions.

Given m functions of n variables $\beta = (\beta_1, ..., \beta_n)$, the Gauss-Newton algorithm iteratively finds the β that minimize the following sum of squares

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{m} r_i(\boldsymbol{\beta})^2 \tag{10}$$

It tried to find the solutions by the iterations

$$\boldsymbol{\beta}^{(s+1)} = \boldsymbol{\beta}^{(s)} - \left(\mathbf{J_r}^\mathsf{T}\mathbf{J_r}\right)^{-1}\mathbf{J_r}^\mathsf{T}\mathbf{r}\left(\boldsymbol{\beta}^{(s)}\right) \tag{11}$$

3. Method

3.1. Map Initialization

To make a map that contains the feature points, we need to define the coordinate. So choose the first frame to world reference frame. We can extract the feature by choosing any two frames, but to reconstruct the 3D coordinates of the featurs, the extrinsic pose between the two frames have to be known. So assume the start of the video have to be moved right side translation without any rotation. If we assume the initial pose between first frame and after 4 seconds frame has only translation of 10cm, the initial extrinsic pose will be as follows:

$$P_1 = [R|t] = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{12}$$

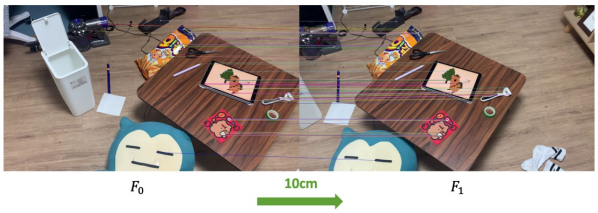

Figure 3. feature matching between 10cm translated frames

Then applying ORB descriptor for each frame, we can get a feature points of it. By using K-nearest neighbor method, matched feature between two frames can be obtained. It means we know the 2D coordinates $p_1(u_1, v_1), p_2(u_2, v_2)$. The pose P_1 is also known, so we can obtain the 3D coordinate of the feature point $P_1(x_1, y_1, z_1)$ by using the result of subsection 2.2. Then the initial map that contains 3D coordinates of features are prepared.

3.2. Plane Detection

To plot the cube in the video, the desk have to be defined first. So we assumed that the desk has the most feature points that composes the plane. In previous step, we obtained map that contains 3D feature points. So we can detect the dominant plane by using Random Sample Consensus(RANSAC). First, randomly choose 3 points in map. Then generate the plane containing it, and count the number of inlier points that the distance from the plane within threshold. Recursively perform it and find the best plane that represents the desk. We conducted 100 iterations in RANSAC.

3.3. Create 3D Cube

Now we have to select the 2D points where we want to place the cube on the desk. If we select 2D points in image, we have to make 3D projection coordinate that crossed with the plane we previously detected. Let's assume the situation as shown in Figure 4.

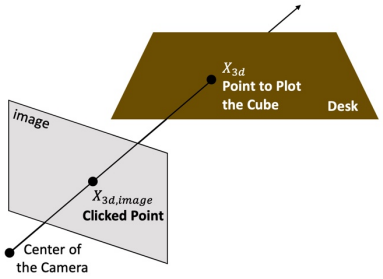

Figure 4. 3D projection

First, we have to transform the 2D point to 3D point in image plane. The relation between two points can be expressed with intrinsic parameters from calibration as below:

$$X_{2d} = (u, v)^T \tag{13}$$

$$X_{3d,image} = \left(\frac{u - c_x}{f_x}, \frac{v - c_y}{f_y}, 1\right)^T \tag{14}$$

The ray that projected $X_{3d,image}$ from center of the camera have to cross the 3D point on the desk plane X_{3d}. So we can obtain the X_{3d} by solving two equations as follows:

$$X_{3d} = kX_{3d,image} \tag{15}$$

$$nX_{3d} + d = 0 \tag{16}$$

Then, we can generate the cubes coordinates center of chosen points. It is defined in world coordinate, so it is constant value and projected to ith frame by using pose.

3.4. Optical Flow

The main task is to find a pose of ith pose. To estimate a pose, we have to find a matched 2D points first. We used optical flow by Lucas-Kanade [8] to find a matched points pairs. We use the brightness constancy assumption.

$$I(x_\Delta x, y + \Delta y, t) = I(x, y, t - 1) \tag{17}$$

If we use Linearizing by Taylor expansion, we can get a equation about motion vector $v = (\Delta x, \Delta y)^T$

$$I(x+\Delta x, y+\Delta y, t) \approx I(x, y, t)+\nabla_x I\Delta x+\nabla_y I\Delta y \tag{18}$$

$$\begin{pmatrix}\nabla_x I(p) & \nabla_y I(p)\end{pmatrix}\begin{pmatrix}\Delta x \\ \Delta y\end{pmatrix} = -\nabla_t I(p) \tag{19}$$

The unknown variable is motion vector which has 2 dimension. One pair of matched points gives one equation, so it makes aperture problem. We need more assumption that the points in local window has same motion vector. We used 50x50 window size.

$$\begin{pmatrix}\nabla_x I(p_1) & \nabla_y I(p_1) \\ \vdots & \vdots \\ \nabla_x I(p_n) & \nabla_y I(p_n)\end{pmatrix}\begin{pmatrix}\Delta x \\ \Delta y\end{pmatrix} = \begin{pmatrix}-\nabla_t I(p_1) \\ \vdots \\ -\nabla_t I(p_n)\end{pmatrix} \tag{20}$$

Let's express the equation 20 as $Av = b$. Then the least square solution of the motion vector can be derived.

$$v = (A^T A)^{-1} A^T b \tag{21}$$

We applied it for one frame unit after the map initialization. Image pyramid is used in 2-level for match various size of feature.

3.5. Camera Tracking

In this stage, we assume that we already have the 3D coordinates of 0th frame and ith frame, and 2D coordiantes of ith and $(i+1)$th frame, which are expressed by $X^0_{3D}, X^i_{3D}, X^i_{2D}$, and X^{i+1}_{2D}. We want to know the motion vector μ_{i+1} from the given coordinates. First, we first set the objective function as the error between the observed 2D coordinate of $(i+1)$th frame(X^{i+1}_{2D}) and estimated coordinates of the frame from the motion vector.

$$\mu_{i+1} = \arg\min_{\mu} e^2 \tag{22}$$

$$e^2 = \sum_{i=1}^{m} {e_j}^2 \tag{23}$$

$${e_j}^2 = \left\| \begin{bmatrix} u_j^{i+1} \\ \\ v_j^{i+1} \end{bmatrix} - \text{proj}(\exp\left(\mu_{i+1}\begin{bmatrix} x_j^i \\ y_j^i \\ z_j^i \\ 1 \end{bmatrix}\right)\right\|^2 \tag{24}$$

To apply Gauss-Newton method on 22, we should compute Jacobian matrix of e_j with respect to the motion vector μ. Since the Jacobian matrices are computed from the each matched point, there are total n Jacobian matrices. The computation is performed as following:

$$\boldsymbol{J}_j = \begin{bmatrix}\frac{\partial e_j}{\partial \mu_1} & \cdots & \frac{\partial e_j}{\partial \mu_6}\end{bmatrix} \in \mathbb{R}^{2\times 6} \tag{25}$$

$$= \begin{bmatrix} f_x & 0 \\ \\ 0 & f_y \end{bmatrix}\begin{bmatrix} \frac{\partial\begin{bmatrix}\frac{x^{i+1}}{z^{i+1}} & \frac{y^{i+1}}{z^{i+1}}\end{bmatrix}^T}{\partial\mu_1} & \cdots & \frac{\partial\begin{bmatrix}\frac{x^{i+1}}{z^{i+1}} & \frac{y^{i+1}}{z^{i+1}}\end{bmatrix}^T}{\partial\mu_6}\end{bmatrix} \tag{26}$$

$$\boldsymbol{J} = \begin{bmatrix}\boldsymbol{J}_1 \\ \vdots \\ \boldsymbol{J}_6\end{bmatrix} \in \mathbb{R}^{2n\times 6} \tag{27}$$

In 26, we exploited SE(3) differentiation method from Lie theory. If the Jacobian matrix is successfully obtained, we apply Gauss-Newton method 10 times to get the optimal solution of 22. The number of iterations 10 is empirically obtained. After the iterations, we can obtain the μ_{i+1} which means the pose between ith frame and $i+1$th frame. So the pose of the $i+1$th frame from world frame is easily derived by multiply μ_{i+1} to the previous pose P_i.

$$P_{i+1} = \exp(\mu_{i+1})P_i \tag{28}$$

3.6. 2D Cube Projection

This step is to project the cubes in 3D world coordinate $X_{3d,w}$ to 2D image coordinate X_{2d}. We obtained the pose of the ith frame P_i, so the projected 2D points can be easily derived as follows:

$$X_{3d,i} = P_i X_{3d,w} = (x_i, y_i, z_i)^T \tag{29}$$

$$X_{2d} = K \begin{pmatrix} x_i/z_i \\ y_i/z_i \\ 1 \end{pmatrix} \tag{30}$$

3.7. Optical Flow Outlier Rejection

But the optical flow is not that accurate if there are flat or edge features. So we have to reject the outliers to handle the accumulative error. We suggested to estimate the position of feature in $i+1$th frame $\hat{X}_{i+1}$ by using the pose between $i-1$th frame and ith frame μ_i. Then compute the error and its mean, standard deviation between matched point determined by optical flow X_{i+1}. If the error is over than $\mu + k\sigma$, we considered to outlier and reject it. This steps is conducted under the assumption that the pose between $i-1$th frame and ith frame μ_i and pose between ith frame and $i+1$th frame μ_{i+1} will approximately same.

3.8. Map Reconstruction

The previous steps ensures to maintain the inliers features. But there are only rejection of outlier, so tracking features in map will be continuously reduced. To preserve the number of features, we need to reconstruct the map and add new features. In map initialize step in subsection 3.1, we had to assume the pose between the initial two frames. But we know the pose between ith frame and $i+1$th frame now, so we can construct the 3D coordinate of new matched features without any additional condition. However enough number of the features doesn't ensure the stability of tracking. If there are too much features, pose can bounce due to many outlier rejection in short period. So it is important to reconstruct the map in proper time. We conducted the map reconstruction when the number of the current feature points in map is less than half of the number of total feature points in initial map. Due to this step, it can preserve the pose tracking even if the camera view goes out of the desk.

4. Results

To test this algorithm, it can easily performed to put any videos that moves right translation for first 4 seconds. There has to be some features in desk to detect. The result of the reconstructed 3D feature points in world frame and the dominant plane chosen by RANSAC is shown in Figure 5. There will be matched features not on the desk between initial two frames, and in bad case, outliers can also generate the dominant plane if there are less features in desk or the features on the other plane is extracted too much. So to make sure the desk plane is well detected, we have to check the features on the desk in 2D image is well chosen to inlier and project the plane to 2D image. We can see that there

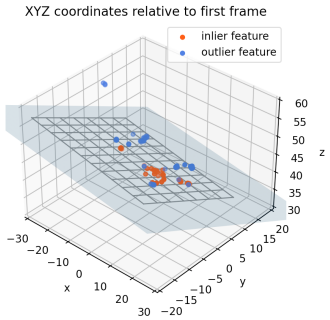

Figure 5. plane detection via RANSAC

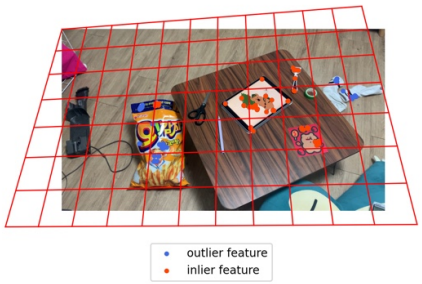

Figure 6. plane projection to image

are various features not only in the desk, and a desk is well detected. Now we can choose the points to place the cubes and the optical flow is conducted. If we don't use the outlier rejection in optical flow, it fails to track the pose due to accumulative error. In the figure 7, the motion vector of the features are not accuracy, so the tracking feature goes out of the corner to flat or edge. If we apply the outlier rejection, figure 8 shows that the optical flow has strong robustness even in fast motions.

If there are less features in current map, the map reconstruction conducted. It occurs when the motion is too fast or the camera view goes out of the initial view. In figure 9 and 10, the view goes outside of the desk, so the features

Figure 7. no outlier rejection

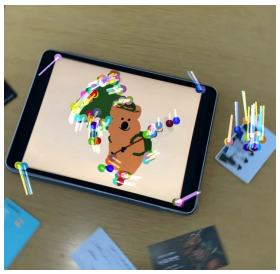

Figure 8. with outlier rejection

Figure 9. before map reconstruction

Figure 10. after map reconstruction

in map are disappearing. But when the map reconstruction conducted, we can see that the new matched feature points are added and conserve the tracking. It makes the tracking robust to the time, fast motion and view of the camera. Finally, we can get the video contains cubes. Cubes are defined in 3D coordinates in world frame, so if the tracking is successfully proceeded, cubes can be placed in any view of camera shown in figure 11 and 12. Figure 14 is the frame that came back after moving the camera at the view outside the desk from figure 13. It means the map reconstruction is well performed and tracking conserved. The time for feature matching and pose tracking between two frames takes up to 0.0153 seconds and the average time was 0.0112 seconds. You can see the example videos or test your own videos in github link in abstract.

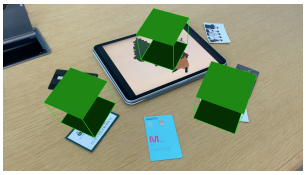

Figure 11. video1 view1

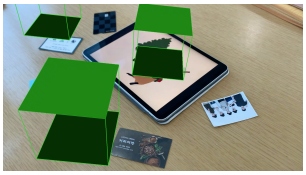

Figure 12. video1 view2

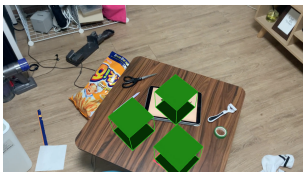

Figure 13. video2 view1

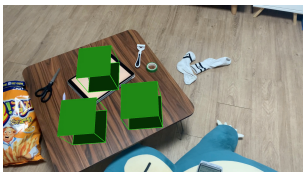

Figure 14. video2 view2

5. Conclusion

To sum up, we have successfully developed a simple augmented reality (AR) technology using traditional computer vision methods, without relying on deep learning approaches. Through the implementation of camera calibration, 3D map initialization, dominant plane selection, 2D point tracking, camera tracking, cube projection, and innovative error handling methods, we have achieved a significantly stable and precise AR model. Our system can accurately display cubes in videos captured by fast-moving cameras or cameras completely out of the initial position and direction. Notably, our approach exhibits impressive efficiency, as it takes only 0.015 seconds per frame to handle high-resolution (1920x1080) videos, enabling real-time operation until 67 fps camera. Overall, our project shows the potential of traditional computer vision techniques in the field of augmented reality.

References

[1] Hsin-Kai Wu, Silvia Wen-Yu Lee, Hsin-Yi Chang, and Jyh-Chong Liang. Current status, opportunities and challenges of augmented reality in education. *Computers and Education*, 62:41–49, 2013. 1, 2

[2] Pauline C Ng and Steven Henikoff. Sift: Predicting amino acid changes that affect protein function. *Nucleic acids research*, 31(13):3812–3814, 2003. 2

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346–359, 2008. Similarity Matching in Computer Vision and Multimedia. 2

[4] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011. 3

[5] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007. 3

[6] Nathan Jacobson. *Lie algebras*. Number 10. Courier Corporation, 1979. 3

[7] Carl Friedrich Gauss. *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*, volume 7. FA Perthes, 1877. 3

[8] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, 1981. 4