Robotics Class Spring 2016

Matching Images through Features and Determining Change in Camera Position

Final Report

Group 13:
Nathaniel PiSierra
njp286@nyu.edu

May 10, 2016

ABSTRACT

The ability to precisely identify the location of a vehicle and map the surrounding environment are key problems in robotics motion planning—especially when dealing with hardware limitations and computing power. Many techniques are used to deal with this problem, such as SLAM—which seeks to estimate the vehicle's location and map the environment by keeping track of landmarks to better understand its position; however, no perfect one solution works for every situation and implementation is usually based off of the available hardware. As such, the mapping of unstructured, changing, and large areas is still a major area of research in the field of robotics. This paper attempts to expand on the identification of region and changes in positioning, using only feature mapping to determine changes in location. We attempt to demonstrate through utilization of the ORB feature matching and the determination of translation and rotation found through the calculation of the essential matrix through the fundamental matrix of two images using a set of 8 matching point pairs.

BACKGROUND

The ability to precisely identify the location of a robot and map the surrounding environment are key problems in robotics motion planning. And the problem only increases due to hardware limitations and computational time constraints. There exist many techniques that aim to solve this problem, such as SLAM—which seeks to estimate the vehicle's location and map the environment by keeping track of landmarks to better understand its position. However, there is no perfect one solution that works for every situation and implementation of a solution is usually based off on the hardware available. As such, the mapping of unstructured, changing, and large areas is still a major area of research in the field of robotics. In the past few decades, the field of computer vision has exploded with research and findings — resulting in new techniques for robots equipped with visual sensors, like a camera, to emerge and is greatly influencing the field of robotic motion planning.

Many of the techniques involved with mapping the robot's environment involve some sort of feature detection— be it line detection or finding key points— and placing these features into a database that will act as a mapping of the environment and give insight into the robot's position. Knowledge about the robot's position is essential especially for robots moving in 3 dimensional space, i.e. drones and autonomous movement of these robots will be rooted in the capacity to know their position in space through mapping of the surrounding environment. In this paper, we research a method for finding the robot's position in three-dimensional space in comparison to a prior position utilizing imaging from the robot.

PROBLEM IDENTIFICATION

This paper aims to find changes in a robots position through comparison of current image to prior images and determining changes in camera position through calculating the fundamental matrix using a set of 8 matched pair points to determine this. We will use a database containing many images from multiple rooms to determine through utilization of the ORB technique to find the closest image—and therefore room of the current image. From there the best matching points between the images will be determined through RANSAC with a high threshold and a fundamental matrix will be determined. This fundamental matrix will then be deconstructed to determine the rotation, and translation from the database photo to the current photo.

We seek to determine how accurately we can determine matching sets of points using the ORB technique and from those points how well can we find an accurate fundamental matrix F and then calculate E and its decomposed values for

translation and rotation of the current camera position as compared to those of the best matched image.

Is there a way to optimize point matching from a current image taken from an obscure angle? And if so, how accurate will the fundamental matrix be when the matching points from a source image are calculated?

Other than distance, is there a way to measure the accuracy of the top ORB calculated matching points?

PROJECT AIM

We seek to find the position of the current camera in comparison to the position of the camera who took the image with the strongest environmental matching features to the current image in order to gain a better understanding of the current position in space and the overall environment.

RESEARCH QUESTIONS

We seek to find when it is possible to get the current camera's position in relationship to a prior position given a set of images that match the current position image. We do this by first finding the image that most closely matches our current image based off finding key points in each image and matching key points from both photos to find those with the closest matching points. We then calculate the fundamental matrix that relates the points in the current image to those in the database image given the set of the key points. From there we calculate the essential matrix, which is deconstructed to give the current camera position in relation to the position of the camera in the most closely matched database image. We hope that this information will give greater insight into how imaging can assist in a robots knowledge of its position in space.

SIGNIFICANCE OF QUESTIONS

The project aim— to understand our current position through it's relationship to a past camera's position, will be met only if a number of checkpoints are met. Firstly, we must successfully choose key points in the images that are accurate. Then these key points must be matched successfully. If they are not matched successfully, we are unable to continue with our goal. If matched correctly, We can then calculate a fundamental matrix, linking the matching points of each image together through a transformation. If this is done with limited error, we can then compute our essential matrix E, which can then be decomposed to determine the rotational matrix and translation vector, which give the difference from our best matched image camera position and our current image camera position. Each stage presents many possible

errors, and these errors are additive, in such that each stage is based on the successful and accurate findings from the previous. As errors increase throughout our program, we get a less accurate number. We strive to lower the errors in each stage as much as possible to get the most accurate results.

LITERATURE REVIEW

Throughout the course of this paper, we have utilized countless resources coming from great engineers and mathematicians as well as more general background knowledge resources in order to gain a strong grasp on the material needed to understand this project. Below we discuss the resources that we utilized the most and who's work presented us with the strongest grasp of the material.

1) Ramani Duraiswami's presentation on Epipolar Geometry and the Fundamental Matrix[6], presented us with a strong basis of the field of Epipolar geometry, in which he outlines the steps needed to calculate epipolar lines and the fundamental matrix given corresponding points between two images. His work also presented us with a problem that is the opposite of ours — how to compete the fundamental matrix given the current camera positions, which provided further insights and understanding into our problem. Furthermore, his work is based highly in describing the relationships between the different aspects of epipolar geometry, specifically how points on one image view relate to another through the fundamental matrix.
2) Probably our greatest resource was Zisserman and Hartman's book, Multiple View Geometry in Computer Vision [8], which after reading other works that gave us initial understanding and insights into epipolar geometry, gave us much deeper knowledge of the workings and relationships of these concepts, which the authors present in mathematical terms along with detailed explanations. This book was truly invaluable in the writing of this paper and grasping the dense material that is epipolar geometry.

Many other resources were used in the writing and researching of this paper, but none gave as great insight into the material as the two sources above.

THEORETICAL FRAMEWORK

Before running the program with data, the camera's intrinsic matrix must be calculated. The intrinsic matrix of a camera represents the 3D camera coordinates to 2D homogeneous image coordinates and holds properties about the camera such as its focal length, principal offset point, and skew, which are needed to understand any lens distortion that is caused by the camera, which creates discrepancies between real-world 3 dimensional points and the 2 dimensional image points in the photo. We calculated the intrinsic matrix of the camera by using Matlab's single-camera calibration feature which exported the camera parameters. We will not go into this in greater detail as it is outside the scope of our paper. This camera intrinsic matrix

only needs to be calculated once— after which its matrix is inserted into the program for future use.

In the running of the program, an image is imported, acting as the "current" position of the robot. This image is then smoothed using a median blur with a kernel of size 3 to reduce pixelation of the image and assist in finding more accurate key points in the image in the following steps. Key points are then found in the image using the ORB technique. ORB is a mapping technique based off the an alteration of the FAST Keypoint Detector and BRIEF features. ORB introduces orientation mapping of features to FAST, which efficiently finds corner keypoints—in a manner similar to the Harris keypoint detector. BRIEF—a method of describing feature points through the use of binary strings[3], is used and modified in the ORB technique of feature finding to save the orientation of keypoints as part of the descriptors. Ultimately, ORB is introduced as an efficient alternative to SIFT and SURF techniques of feature finding, which are slow and costly[1]. Additionally, we attempted to use these techniques and they did not result in significantly better key points and subsequent matches as compared to those found by ORB.

Next, saved images in the database are then looped through one by one. During this stage the ORB is applied to each image as it is looped through and its keypoints and descriptors are compared to those of the original image. The points are matched and then organized based on closest distance from one another— where a distance of 0 means there is an exact match between the points, distances that move away from zero are less likely to be matches the further away they are. The computation of distances is done using the Hamming distance between the two points' descriptors (binary strings that hold data about the point's orientation, size and position). The Hamming distance is how greatly the strings differ from each other, where, for example "0000" and "0006" would have a Hamming distance of 1 as the strings only differ by changing one number. After the best key points are matched through a function BFMatcher — which essentially loops through each points to find the pair with the lowest distance score[5]. Once the algorithm has found the points the pairs are ordered from lowest to greatest distance. From there, the standard deviation of the difference in each of the ten best matching pair's sizes and angles is computed and each multiplied to the distance score. This acts as the final score of the image in how closely it matches the "current image", where the closer to zero it is, the more likely it matches. After each photo in the database is looped through— there photo with the lowest score is the one that most closely resembles the current image. From previous trial-and-error if the score of the photo is less than 30, then it is most likely a match with the image, it is above 30 then we cannot say with confidence that the image is a match and the program will return stating such.

If the image's matching score is below 30, the program continues to fill two arrays with the 40 best matching points, one holding the current images points and the other holding the best matching images points. These points then used to calculate the fundamental matrix, using OpenCV's findFundamentalMatrix function. The fundamental matrix, F is used in epipolar geometry to describe the projection between two images. Both translation and rotation of the camera must occur in order to calculate F. If the difference between the images is solely based on

rotational differences between the camera pose, then a homography must be used to calculate the difference in position[A1]. F represents the relationship between two images that hold matching points that correspond to the same real-world points. For example, this point would be x in the best match image, and x_i in the current image, both of which map to a real-world point, X, and the relationship is written as:

$$x_i'^{T} F x = 0 \qquad (0)$$

which demonstrates that F is essentially a projection matrix from x to x_i. In order to explain how F is calculated— I must first explain some concepts in epipolar geometry.

To begin with, there are two images, I and I', which are essentially planes which respectfully contain points x and x_i that correspond to a real world point X. To better explain this imagine that I and I' are placed in a 3-dimensional space in positions that are similar to those of how each was taken by the camera. There runs a ray from each camera center that goes through their corresponding x image point and through the real-world point X, additionally there is an image in the Appendix that demonstrates this idea more clearly. These two rays connect at point x and if you imagine a line running from each camera center through their respective image plane toward the other camera center, then there is a triangle of sorts connecting these 3 points— this is called the epipolar plane. Additionally the spot where the line connecting the camera centers meets their image plane is the epipole— all intersections between the epipolar planes go through this point— creating an epipolar line in the image plane. Essentially, the epipolar line (l') in the current image is the projection of the ray from the point x through the best matched camera center. This creates a mapping of all points x onto l'. This is essential in satisfying (0), due to the fact that if points x and x′ correspond, then x′ lies on the epipolar line l′ = Fx corresponding to the point x. Furthermore, the set of all epipolar lines in each of the images forms a pencil, a family of geometric objects with a common property— in this case each line passes through the epipole[8].

This pencil is a one-dimensional projective space as it is only a point. Since the lines are related, there is a homography [A1] between the pencil in the first image plane and that in the second, which results in 3 degrees of freedom due to being in a 3 dimensional space. Additionally, each epipole has two degrees of freedom (they exist in a two-dimensional space)[6]. This results in F having 7 degrees of freedom and as such in its calculation needs at least 7 matching point pairs. We calculate F automatically by utilizing a RANSAC (Random Sample Consensus) scheme, which consists of finding high correlations between randomly selected sets of matched pairs . In each set of point pairs the distance is calculated between each point pair and the corresponding epipolar lines and find which pairs exist within the parameterized threshold. The points outside the threshold are discarded and F is calculated. This process is done once more, with the smaller set of input pair points in order to get a more accurate F[7].

Once F is calculated, we can get an error of how close the points are by multiplying the transpose of the points of the current image by matrix F by the

points in the database image. This should result in zero, however, errors in the calculation of the fundamental matrix and incorrect matches of key points can result in a fundamental matrix that is slightly off.

We cannot get any direct information from the fundamental matrix about the pose of the camera, as the fundamental matrix only describes transformations about the images and the images are not exact matches to real world points. To get the real world points we must calculate the essential matrix, E, using our camera internal parameters, K:

$$E = K^T F K$$

This describes the transformation that is occurring in real world points as opposed to image points. We are not finished yet though, as E must be singular value decomposed to get :

$$E = U\,\mathrm{diag}([1\ 1\ 0])V^T$$

Where U is the upper triangle matrix From here we seek get the skew symmetric matrix, S and rotational matrix, R such that:

$$E = SR$$

To do so we utilize two matrices W (rotation) and Z (skew-symmetric):

$$W = \begin{vmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad Z = \begin{vmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

Furthermore, for these matrices we have:

$$ZW = \mathrm{diag}([1\ 1\ 0])$$
$$ZW^T = -\mathrm{diag}([1\ 1\ 0])$$

We can now find two solutions: $E = S_1R_1$, where $S_1 = -UZU^T$, $R_1 = UW^TV^T$ and $E = S_2R_2$, where $S_2 = UZU^T$, $R_2 = UWV^T$. From here, we compute the translation by finding a translation vector t from S, which exists in S's null space. The null space of both S's are the same and is calculated from the third column of U. It is important to note that since t is in the null space of S, then t multiplied by any constant, λ is also in the null space, where λ is non-zero. Different λ's correspond to different scalings of the solution thus we cannot calculate the true value of λ, but the sign of λ is important because it determines whether points are in front of both cameras. To find a solution where the points are in front of both the cameras we test $\lambda = \pm 1$. We use the third column of U (the translation vector) , u_3 to get four solutions for the four possible relative camera positions:

$$P = [UWV^T\ u_3] \text{ or } [UW^TV^T\ u_3] \text{ , for } \lambda = 1$$

and

$$P = [UWV^T - u_3] \text{ or } [UW^TV^T - u_3], \text{ for } \lambda = -1$$

Only one of these solutions represents the correct camera positioning, and to determine which, we must use triangulation for each choice of P to select the one where the points are in front of both cameras[9]. Essentially there are two opposite directions where are possible, given two different rotations that are compatible with the essential matrix and two translations, as well. In total this gives 4 possible solutions to the camera's relative position. Only one of these solutions will be correct as the other 3 will create a 3d point which lies behind a camera pose.

We do this by first normalizing each set of points by applying the camera's intrinsic parameters matrix to each point, which removes some of the point discrepancies due to the camera's hardware. We then check for which of the four camera poses is accurate. This is done by plotting our 2d image points to their 3d counter parts through use of the rotational matrix being tested and the translation vector being tested. For every matched point we test if they are in front of both cameras. We start by using the following equation to determine the z coordinate of the 3d point:

$$z = ((r_1 - y_1 * r_3) * t) / ((r_1 - y_1 * r_3) * d)$$

In this equation, r_1 and r_3 represent the first and third rows of the Rotation matrix being tested, y_1 represents the current point's y value, t is the translation vector being tested and d is the database point [10]. This is done also from the perspective of the current point, where the equation is altered to reflect the other point. If a z point is less than zero, that means that it must be behind one of the cameras and therefore this rotation-translation pair does not represent the correct and the rest of the possible solutions are checked until the correct solution is found.

Finally, the rotational matrix is decomposed, where R is the rotation matrix:

$$R = \begin{vmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{vmatrix}$$

$$\text{Rotation}_X = \text{atan2}(r_{32}, r_{33})$$
$$\text{Rotation}_Y = -\text{asin}(r_{32})$$
$$\text{Rotation}_Z = \text{atan2}(r_{21}, r_{11})$$ [4]

These results are then presented to the user along with the translations, as well as the current image and database image, each with the epipolar lines drawn on.

RESEARCH METHODS/ METHODOLOGY

Results from the ORB and matching algorithm can be verified by one's eyes and can clearly be seen if they are accurately matched. The weight of the standard deviation of the matrix containing the difference of the top matching points' rotations and

sizes can be increased or decreased to adjust the score of the photo and increase accuracy in finding the most likely matching photo.

Verifying the results of the Fundamental Matrix, F are done by using the equation:

$$x_i'^T F x = 0$$

where we use all the current image points used in calculating F as x_i and the best matched image points as x. The further away from 0, the greater the error in either our F matrix or our key points.

It is more difficult to verify the results of the decomposition of the essential matrix as the construction of the essential matrix is due to the camera intrinsic parameters measured through Matlab and any error in this result leads to errors in the essential matrix and its resulting decomposition — i.e. the translation and rotation of the current camera position as opposed to the best matched image camera position. However there can be verification through general looking at the results and comparing the current camera image to the database image.

RESULTS

We have mixed results from our program— some were successful, while others were unsuccessful. The outcome of a successful result appears to occur when there are significant camera translations in the x and y directions, as well as a slight rotation of the camera. When the translations are small between the current image and the best matched image— the program is unable to accurately give the difference in rotation between the current position of the camera and the best matched camera position.

Introducing a cutoff point for matched images of 30.0, has resulted in more successful results, that had previously been unsuccessful; however, it does not allow for the full scope of the project aim to be reached.

This is partially due to the limitations imposed by epipolar geometry and the fundamental matrix, as it only is a viable solution to this problem when there is translation as well as rotation of the camera, as discussed in the theoretical framework section. In the case of only rotation, it would be better to use a homographical mapping between the images to determine the rotations between the cameras.

Additionally, due to the setup and nature of our program, there are many steps in calculating the difference in camera pose, and at each of these steps there exists the possibility of errors. These errors become additive, and as the program reaches the end, the errors, be it from the initial camera calibration for the camera's intrinsic matrix, differences between the matched points, or from choosing the best points for the calculation of the Fundamental matrix, are lumped together and the greater the sum of the parts, the worse the result. We have attempted to remove as many errors as in our capacity, but some still can sometimes persist to give poor results.

Below, we have given a couple examples of successful trials, as well as our unsuccessful trials.

TRIAL 1

Left: Current image after blurring and gray scaling as well as epilines are drawn on and the matched points are plotted
Right: Best Matched image from the database under the same conditions as the left image

You are most likely in room: 3
Distance: 2.01372270499
Error in fundamental matrix: [[8.82627305e-14]]
XRotation: 0.190966, YRotation: -0.179864, ZRotation: -3.065229
Translation vector: [3.95087714e-04 2.10102809e-02 9.99779182e-01]

From this output we can see there there is a clear rotation from the best matched image camera position to the current camera position in the Z direction. Additionally the scale of the translation appears to be correct as there is mostly movement forward in the Z direction and significantly less movement in the x and y directions. Additionally, the correct room is selected.

TRIAL 2

This represents a somewhat successful trial. The left image shows the current photo and the right is the best matched photo.

Output:
You are most likely in room: 2
Distance: 18.8128269858 (the closer to zero the higher the probability you are in given room)
Error in fundamental matrix: [[-1.53243280e-05]]
XRotation: 2.190350, YRotation: -0.929665, ZRotation: 0.005245
Translation vector: [-0.08484345 0.28693603 -0.95418515]

For this result, the translations are pretty accurate— there is significantly more movement in the Z direction and some movement in the Y, and almost none in the X. The rotations, are clearly incorrect though. There is no rotation in the X direction and a lot of rotation in the Y.

TRIAL 3

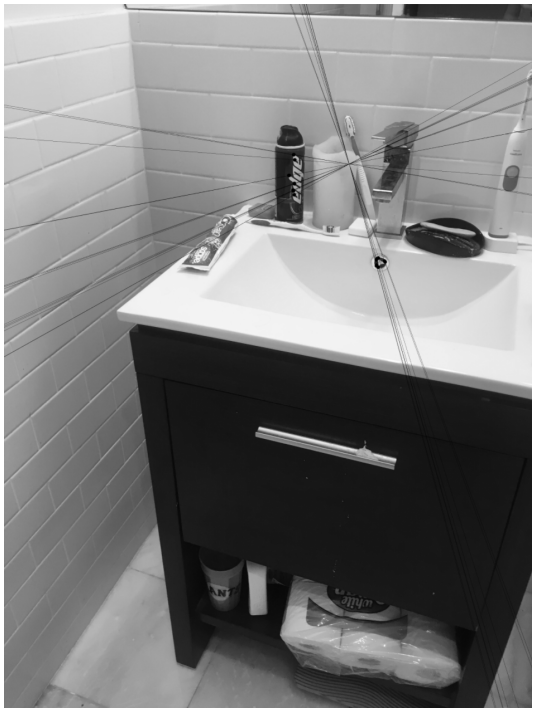
This represents an unsuccessful trial. The left image shows the current photo and the right is the best matched photo.

You are most likely in room: 3
Distance: 2.4360595042
Error in fundamental matrix: [[-4.70683211e-05]]
XRotation: 2.581300, YRotation: -0.548327, ZRotation: 0.004643
Translation vector: [0.12343071 -0.27567838 0.95329234]

This is clearly an unsuccessful trial as the rotations are off as well as the translations. There should be a lot of rotation in the Z direction, but the output is the least of the 3 rotations. The x directions rotation might be correct. Furthermore, the translation vector shows the greatest change, however it seems from the photos that the camera moved the most in the x direction. This incorrect finding could be due to the fact that all the matched key points fall on a single line, which doesn't allow the fundamental matrix to be calculated correctly— thus affecting the results.

TRIAL 4

This represents a successful trial. The left image shows the current photo and the right is the best matched photo.



You are most likely in room: 2
Distance: 11.01564591
Error in fundamental matrix: [[8.10777406e-05]]
XRotation: -3.132764, YRotation: 0.008614, ZRotation: 3.011175
Translation vector: [0.12594586 0.29034351 -0.94859806]

This trial is successful in both its translation and its rotation, both of which appear to be accurate as there is clearly rotation in the z and x directions, but not really in the y. Additionally there is movement in the Z direction, and significantly less in the x and y directions.

These 4 results are a good representation of the average results of the program. We would like to note that our matching of key points is extremely accurate throughout the program, such that no wrong photos were matched that fell beneath the cutoff point. It seems that our program is lacking in its ability to always accurately determine the translation and rotations between camera positions.

CONCLUSIONS

From our results, it is clear that the program is not completely accurate in its findings of camera position. In some cases, the program performs well and it is clear that the matches between photos are accurately made and the given rotations of the current camera position to the camera position in the matching photo seem to be accurate. Additionally it appears that when a current image is being compared to an image where there is little translation of the camera position between photos and only rotation occurring, that our program cannot accurately describe the rotation that is occurring. Finally, we see that our program falls short in its ability to determine the actual translation of between the camera positions. This is due to the parameters of our study— as it is not possible to get the real translations, due to not knowing either camera's actual position in space, thus it can only be given as a scaled translation, where the actual scale is unknown.

If we continue this research in the future, we will look into implementing different methods for finding the Fundamental matrix and see if they produce more accurate results.

RESOURCES

Python with OpenCV libraries was used to preform analysis and calculate functions.

Matlab was also used to get the Intrinsic parameters of the camera.

REFERENCES

1. Rublee, Ethan; Rabaud, Vincent; Konolige, Kurt; Bradski, Gary (2011). "ORB: an efficient alternative to SIFT or SURF" (PDF). *IEEE International Conference on Computer Vision (ICCV)*.
2. Vincent, Etienne, and Robert Laganie`re. "Detecting Planar Homographies in an Image Pair." University of Ottawa, n.d. Web.
3. Calonder, Michael, Vincent Lepetit, Christoph Strecha, and Pascal Fua. "BRIEF: Binary Robust Independent Elementary Features ⋆." (n.d.): n. pag. CVLab, EPFL, Lausanne, Switzerland. Web. <https://www.robots.ox.ac.uk/~vgg/rg/papers/CalonderLSF10.pdf>.
4. Ho, Nghia. "DECOMPOSING AND COMPOSING A 3×3 ROTATION MATRIX." *Nghia Ho*. N.p., n.d. Web. <http://nghiaho.com/?page_id=846>.
5. "Hamming Distance." Wikipedia. Wikimedia Foundation, n.d. Web. 09 May 2016. <https://en.wikipedia.org/wiki/Hamming_distance>.

6. Duraiswami, Ramani. Epipolar Geometry and the Fundamental MatrixReview about Camera Matrix P (n.d.): n. pag. Ramona Duraiswami. University Of Maryland. Web. <http://www.umiacs.umd.edu/~ramani/cmsc828d/lecture27.pdf>.
7. "Geometrisk Bildanalys (Geometrical Image Analysis), VT-05." Reconstruct 3d Image. N.p., n.d. Web. 09 May 2016. <https://www8.cs.umu.se/kurser/TDBD19/VT05/reconstruct-4.pdf>.
8. Hartley, Richard, and Andrew Zisserman. "8." Multiple View Geometry in Computer Vision. Cambridge, UK: Cambridge UP, 2003. <http://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf>.
9. Olsson, Carl. "Lecture 6: Camera Computation and the Essential Matrix." Lund Institute of Technology. Web. <http://www.maths.lth.se/matematiklth/personal/calle/datorseende13/notes/forelas6.pdf>.
10. "Essential Matrix." Wikipedia. Wikimedia Foundation, n.d. Web. 10 May 2016. <https://en.wikipedia.org/wiki/Essential_matrix>.
11. "Epipolar Geometry." Sirs Lab, n.d. Web. <http://sirslab.dii.unisi.it/vision/research/VS_nonh_4.jpg>.

APPENDICES

A1.

Homography is the projection of one image plane onto another from using matched feature points. Homographies are represented by a nonsingular, 3 x 3 matrix H, which relates images x and x^i (having matching keypoints) onto a plane:

$$x' = Hx$$

For H to be calculated there must be several pairs of matching points in each image. It is of the utmost importance that these pairs match in order to get the correct homography or translation matrix. For this reason many of the best techniques for mapping homographies utilize a RANSAC (Random Sample Consensus) scheme, which consists of finding high correlations between matched pairs. This prevents outliers from inaccurately identifying homographies. It is important to note that the homography equation can be rearranged to:

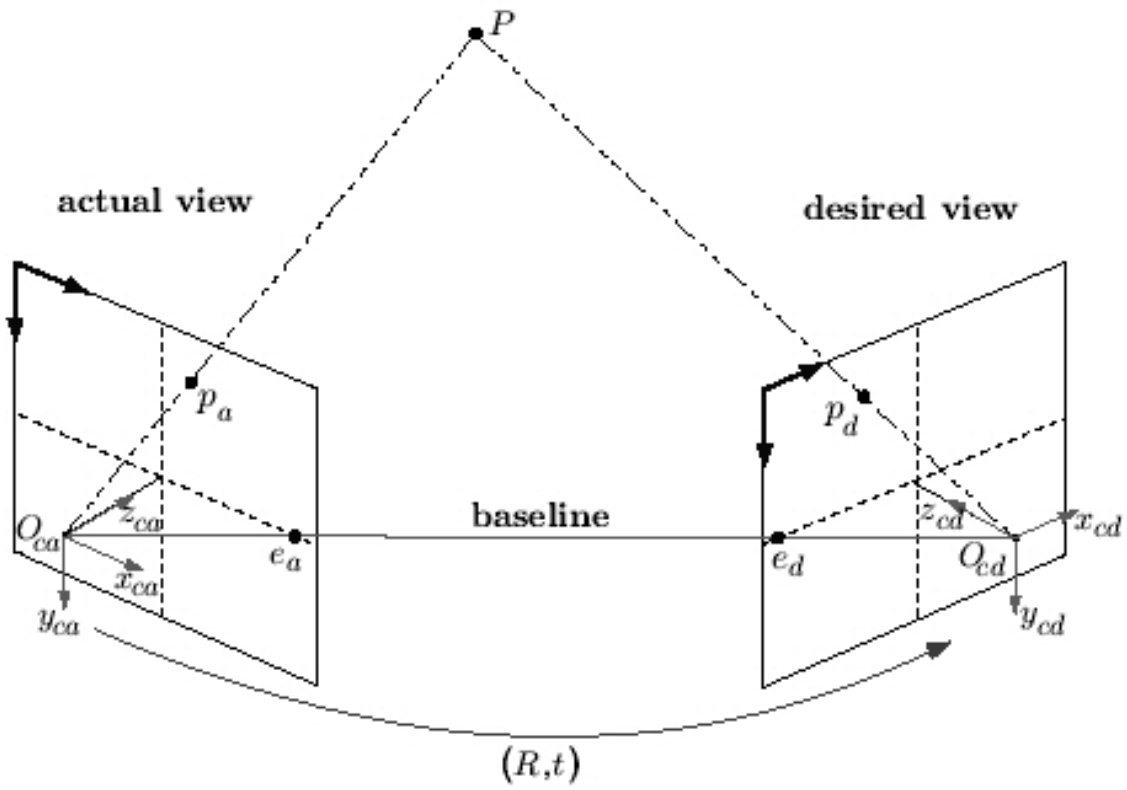
$$x' \times Hx = 0$$

RANSAC heavily relies on this equation and only chooses feature pairs that agree with this homography given for a threshold ε :

$$\text{dist}(Hx, x') < \varepsilon$$

The alteration of the threshold results in different pairs of points being considered for the homography—with a more relaxed threshold allowing for a faster, albeit more crude and innaccurate, homography to be found[2]. Ultimately a homography is nothing more than a matrix that represents the transformation of one camera position to the current camera position.

A2.



[11]

A3. Examples of images in the database: