Transformation and Projection of Images and Logos onto Distorted Surfaces

Lauren Jernigan, Lauren Mangibin, Marika Murphy, Roger Zhong
College of Natural Sciences: Computer Science
University of Texas at Austin, Austin, TX 78712
ljernigan447@gmail.com, lauren.mangibin@gmail.com, marikamurphy@utexas.edu, rzhong688@gmail.com

***Abstract*—Our project aims to transpose images onto 3D surfaces. The two surfaces in question are flat planes and cylinders. The method for both surfaces calculates the homography between the undistorted image and the surface that the image will be transposed onto. Finding points in the undistorted image is trivial, but we simplify the complicated process of finding points on the surface by placing a chessboard on the surface. With these two sets of points we calculate the homography and apply it to the image, then project it on top of the chess board surface. This method works well for the flat surface like the plane, but homographies only work for 2D to 2D transformations, so we have a modified method for the cylinder. The cylinder is broken up into the pieces of the chessboard, and we apply the homography method for each section of the cylinder. This method for transposing an image onto a cylinder gives an approximation of what the image would look like on the curved surface. There are still issues with scaling and matching the edges that need to be taken into account.**

I. Introduction

Our original inspiration for this project was to use the logo on the back of someones shirt as a way to identify and follow them. The first difficulty we had with this idea, was that shirts have folds and wrinkles which distort the image on them. This would make it considerably difficult to identify the image. To solve this problem, we decided to work towards undistorting an image. This led us to homographies, the relationship between a 2D image and a linear transformation of that image. Homographies work well on linear transformation, but they are unable to fully represent non-linear transformations such as curves. Determining a method to project an image onto a non-linear surface became the focus of our project.

Converting between distorted and undistorted images can be helpful for how people and computers perceive the images. Distorted images seen through a camera can be difficult for computers to process. The image is most likely presented from a straight on perspective. Images seen from a different angle or on a non-linear surface will look nothing like their base shape, which makes it difficult for computers to draw links between the two. On the other hand, perceiving and recognizing depth and distortion comes more easily to humans. For example, if someone were to take a picture of a Ferris wheel at an angle, a cylinder in 3D space, they would find that the Ferris wheel is an oval in their photo. As humans, we recognize that the oval in 2D is really a circle in 3D because 3D objects have depth as an additional layer of information. If an image that is supposed to be on an angled surface appears from a straight on perspective, we recognize that it looks wrong. Therefore, it is important to seamlessly transition between different distortions for both computers and humans.

The relationship between the distorted image and its original image can be described using a homography. In this project, we use homographies to map logos onto different surfaces, both linear and non-linear. The homography only works for linear transformations, so it does well when projecting onto another plane. However, we have to modify our approach for non-linear surfaces such as cylinders. For surfaces that are not deformed linearly, we break them up into smaller rectangle. For each of these rectangles, the homography between the rectangle to the original image is mapped. These homographies are applied to their corresponding section of the original image and transposed onto the surface.

II. Background

Image distortion involves two components: the original image and its distorted version. This means there is a relationship between the points of the images that can be captured. This relationship is called a homography. Homographies have been a prominent part of computer vision, specifically with distortion, because they describe projective transformations. In logo detection, we want to track how a logo was transformed and be able to calculate that relationship to undistort the image.

Capturing an image in itself distorts the image because of its set parameters. One caveat of analyzing data from a camera is that not all cameras are created equal. Every camera has varying amounts of lens distortion that distorts the location of points in the image plane. A fisheye lens, for example, captures a wide field of view around the camera and projects the image onto the plane with a large amount of lens distortion. Straight lines are not depicted as straight lines in the image. To correct for these inaccuracies, creating a matrix that describes these camera intrinsic parameters is necessary. For the purposes of this project, all of the images are taken with a typical smartphone camera with little lens distortion, so we can assume that the camera is close to an ideal camera, hence no camera intrinsic matrix is needed.

Although we did not need to take into account the camera intrinsic parameters for the purposes of this project, we still needed to compute the homography, which involves 2D homogeneous coordinates. Homogeneous coordinates are a system of coordinates used in projective geometry just as Cartesian

coordinates are used in Euclidean geometry. Homogeneous coordinates are easier to rotate and translate because they allow these transformations to be represented as a matrix by which a rigid transformation, or a matrix comprised of both rotations and translations, is multiplied [1].

We use homographies because they are the representation of the relationship between two 2D planes within a 3D matrix. They are projective in nature. In order to compute homographies, you need a 2D homogeneous plane which includes the X, Y, and W axis obtained through a 3D system. This would require a rank reducing projection, which involves the projective transformation from a higher to a lower dimension. In our case we reduce from 3D to a 2D plane. In our project we rotate a 3D matrix using a rigid transformation and project it down into 2D by converting the 3D matrix into a 2D homogenous matrix.

As demonstrated in Figure 1, the 3D matrix has the X, Y, Z, and W axis, which can then be transformed into a homogeneous 2D matrix by dividing X, Y, and Z by Z, leaving Z's row to be all ones and thus redundant. This gives us the homogeneous 2D matrix on the right of Figure 1.

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x/z \\ y/z \\ w \end{bmatrix}$$

Fig. 1. Converting a 3D matrix into a homogeneous 2D matrix

Once there are two 2D planes, one can find the homography between the two through the Direct Linear Transformation (DLT) algorithm where there at least four point correspondences between plane one and plane two [2].

The basis for DLT rests on the cross product of the set of transformed points and the set of original points multiplied the homography matrix.

For each point and its correspondent, the DLT matrix in Figure 2 is filled and is concatenated to the matrix of the previous point. This matrix is then multiplied by a 1x9 matrix of values to solve for (i.e $H_1, H_2, ..., H_9$), which are the individual elements of the homography matrix as seen in Figure 3 when transformed into a 3x3 matrix. If there were four points, there would be eight lines, where each line is an equation set equal to zero.

$$\begin{bmatrix} 0 & 0 & 0 & -w*x_0 & -w*y_0 & -w*w_0 & y*x_0 & y*y_0 & y*w_0 \\ -w*x_0 & -w*y_0 & -w*w_0 & 0 & 0 & 0 & x*x_0 & x*y_0 & x*w_0 \end{bmatrix}$$

Fig. 2. Direct Linear Transformation coefficient matrix

Eight equations are produced from the DLT algorithm that we can put into a linear solver to solve the homogeneous system $Ah = 0$, where A is the coefficient matrix and h is the elements of the homography matrix. This returns a 3x3 matrix that conveys the relationship between each set of four points such that when you apply this matrix to the points in set one, we get the transformed points in set two.

$$H = \begin{bmatrix} H_1 & H_2 & H_3 \\ H_4 & H_5 & H_6 \\ H_7 & H_8 & H_9 \end{bmatrix}$$

Fig. 3. Homography matrix

We can find homographies between sets of more than 4 points. However doing this will create a coefficient matrix A of size $2n$x9, where n is the number of points. The matrix will therefore have more rows than columns which means that this is an overdetermined system. A good method of finding a least squares solution to the overdetermined system is taking the singular value decomposition of A. The solution for the system will be captured in the V matrix as the smallest eigenvector of $A^T A$ that is non-zero.

Homographies, distortions, and projective geometry are well associated with chess boards because they have a recognizable pattern that makes tracking points easier. OpenCV has a program, findChessboardCorners, that finds the corners of a chessboard automatically.

Chessboards are useful in that they have a set of easily found points at regular intervals. This makes it easy to sample points along an undistorted image, as you can choose points along a grid that models the chessboard. It also simplifies the process of finding the corresponding points on the surface. This allows us to calculate the distortion of our surface and the corresponding homography.

III. Methodology

A. Creating the Homography Method

To understand homographies, we created our own homography method. This required seven steps: creating a 3D board and converting it into a 2D homogeneous matrix; creating a random rotation matrix that would be applied to the 2D homogeneous matrix; applying the rotation matrix to the 2D homogeneous matrix; finding the homography of the 2D transformed matrix and the original board; applying the homography to the original board; and lastly displaying the transformed matrix.

We first created a 3D board then converted it into a homogenous 2D board. We then created a random transform and applied it to the original board through the dot product.

Now that we had both the original and transformed board, we were able to calculate the homography using the Direct Linear Transformation Algorithm. Then, we put the concatenated matrix through a linear solver to find the 3x3 homography matrix detailing the relationship between the original board and the transformed board. This allowed us to test our implementation. Since the randomly generated distorted board matched the one generated by multiplying the original board by the homography, we knew our implementation worked.

After creating the homography, we then applied it to an actual image of a logo by using CV2's warpPerspective method. This allowed us to distort the image such that it matched our boards distortion. We also attempted to use the

function remap, however, due to the distortion, the function overwrote pixels and didn't display the full image.

B. Applying the Homography to Chessboard Images

We then applied this method to chessboard images, calculating the homography between points on a distorted chessboard image and a regular chessboard image. To get the points on the chessboard and create our matrix, we used CV2's findChessboardCorners. After finding the points on a chessboard, we calculated and applied the homography to the full image to get the distorted version and overlaid it on the chessboard itself. In order to maintain the proportions of the image, we padded the image with enough black pixels to make it square. To overlay the image, we found the outline and contours of the logo using CV2's findContours and found the bounding rectangle of the logo. We then removed the extra pixels and pasted the image at the appropriate coordinates on the chessboard.

C. Overlaying an Image Onto a Chessboard

Taking it a step further, we then tackled non-linear transformations and projective geometry using a curved surface. When a 2D image is mapped to a non-flat plane, transformation cant be described by a single homography. Our solution was to break the transformation into multiple homographies with each chessboard square representing its own plane and generating a homography for its four points. This is akin to calculating a Riemann sum for a curve by splitting it up into sections. Since we were using a 7x7 board, we calculated 36 different homographies since findChessboardCorners does not find the points of the outer row. Next we divided the image up into 36 subsections and multiplied it by its corresponding calculation. After each calculation, we pasted the section onto the distorted chess board image using our previous method. One issue that we encountered was that the dimensions were skewed and the subsections didn't properly fit together as seen in Figure 5. Our solution was to resize each subsection to fit the dimensions of the chessboard square that it covered before projecting it onto the board.

IV. Experimental Setup

To test if the image overlaid on the chess board properly, we took pictures of a chess board at different angles and applied our program on them to:

- Track the homography between the original chess board and the angled chess board
- Apply the homography to the image we wanted to paste onto the chess board
- Paste the image onto the chess board

By applying our program to different distortions of the chess board, we were able to verify the accuracy and consistency of our program. We checked if the homography between the chessboard and the angled chessboard and that of the distorted image is the same. We also checked how accurately the image was displayed on the chessboard by seeing firstly that the angle and rotation was the same and secondly if the image fully covered the chessboard.

V. Results

We were able to successfully track the points from the distorted chess board photos as seen in the middle right of Figure 4. However, this only worked when the chessboard was at a certain angle. CV.findChessboardCorners was not able to find corners at extremes, limiting our ability to track certain distortions and homographies.

From the chess boards that CV.findChessboardCorners was able to track, we calculated the homography between a regular chessboard and the distorted chessboard and applied it to the image as seen in the middle left of Figure 4.

Then we were able to overlay the image over the chessboard as seen in the bottom of Figure 4.

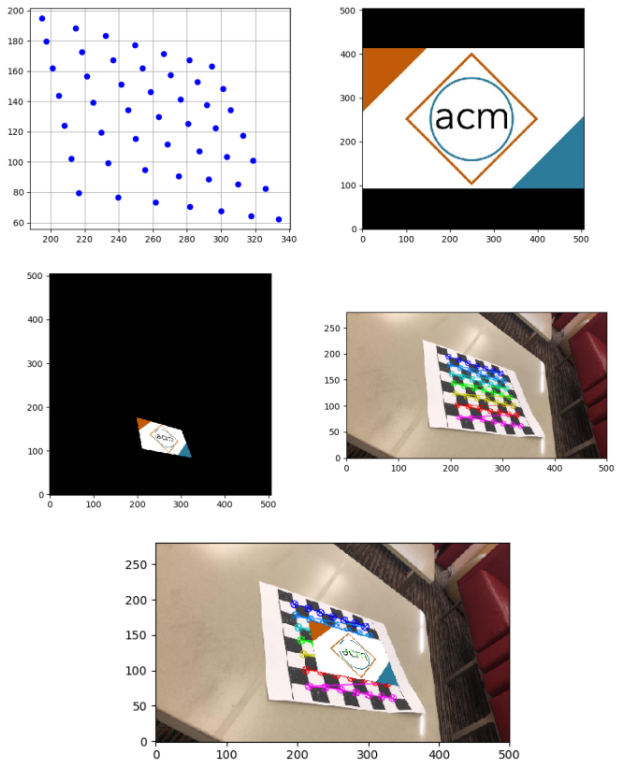

Fig. 4. Top Left: Homography applied to the original board. Top Right: Original image. Middle Left: Homography applied to image. Middle Right: OpenCV's findChessboardCorners. Bottom: Image overlaid on chessboard.

Unfortunately, we were not able to overlay the image completely onto the chessboard because of a difference in aspect ratio. The chessboard is a square, and our image is a rectangle, so to fix the aspect ratio, we added a black border to make the image square. From there we applied the homography, then cropped out the black, making the image smaller than the board. Furthermore, findChessboardCorners is not able to track the outer border of the chessboard, limiting how much of the board can be filled. We hope to fix this in the future, but despite the size difference, the image has the same linear distortion as the chessboard.

We were also able to overlay an image onto a non-linear plane. After breaking up the image and applying a homogra-

phy to each individual section, we combined those sections together to get a ragged wrapped image onto the chessboard. Figure 6 shows this ragged board with both a curved board and a folded board.

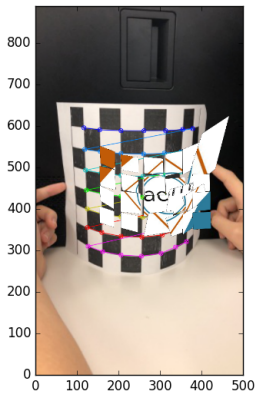

Fig. 5. Non-linear projection before resizing.

While the individual homographies alone yielded a recognizable result, there are gaps in the figure due to the disco ball effect where we attempt to overlay a linear image onto a non-linear image. There was also an anomaly in the left corner of the original application with the top left rectangle because its size did not fit the chessboard. This left some sections of the chessboard unfilled and others overlapping as seen in Figure 5.

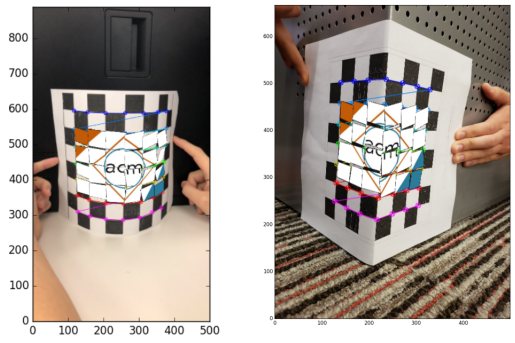

Fig. 6. Left: Homographies applied to a curved chessboard. Right: Homographies applied to a folded chessboard.

As seen in Figures 6 and 7, each piece was reduced in size so that it fit inside its corresponding rectangle on the chessboard. The scaling made the pieces align better to create a more cohesive image.

Although the connection between each section was a bit choppy, we still managed to project the image onto a non-linearly distorted chessboard.

VI. Discussion

Our next step in the project would be to project onto non-linear surfaces more smoothly by either using a point cloud or

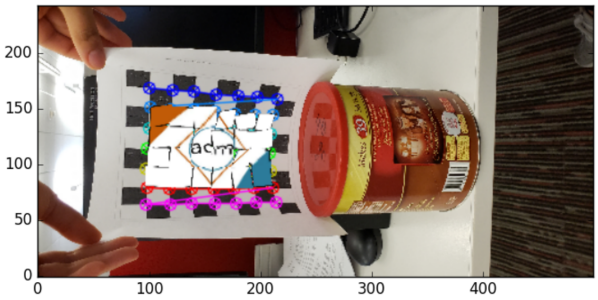

Fig. 7. Left: Homographies applied to a concave chessboard.

a spline. If we were to use a point cloud and more individual homographies, we would be able to increase the accuracy of the image. A point cloud gives us a better understanding of the surface our image is being projected on. This would give us more points, and therefore more groups of four, minimizing the amount of unfilled space between each section, still using the original linear method.

If we were to use a spline to define curves, we would be able to apply the function from each section of the spline to the image, which would curve the image based on the function. Using a spline would allow us to more thoroughly define non-linear distortions and create smoother images because they are aligned to a curve rather than linear approximations to a curve.

There are many applications of image distortions, especially in regards to image wrapping and undistortion. For example, wrapping images around non-linear objects would allow you to cover a bottle with a company logo in a photo or even change logos on the bottle by covering it with a different logo. This could help with copyright infringement on live images by blurring out or replacing copyrighted works more automatically. On a larger scale, this would be helpful for recognizing logos at RoboCup at Home. For example, when the robot is asked to fetch a Sprite, instead of searching its data base for Sprite cans which may not be in the database, it could recognize that the Sprite can is the same as a Coke can, which is in the data base, and just look for the Sprite logo distorted to the shape of a Coke can.

Our ultimate goal, however, is to use logos as a means for person following by the Building Wide Intelligence (BWI) robot. If used by the BWI robot, logo detection is a potential tool for creating a unique ID for a target person to help identify the person if the robot loses track of the person. Therefore, undistortion would be important to recognize and keep track of the logo if it was distorted. Of course, these are applications in the future, but through our project, we have gained a basic understanding of image distortion and projective geometry that has potential for these future projects.

VII. Conclusion

This project has explored distortions of images and their relationship between their undistorted counterpart. Through projective geometry, we were able to calculate the homography between two chess boards and apply it to an image to

create the same distortion. From there we overlaid it onto the chessboard and wrapped the image around a non-linearly distorted chessboard even though the resulting image was a bit choppy. Despite focusing primarily on linear distortions, we explored the relationship between points of both non-linear and linear transformations and applied this relationship to image distortion.

In the future, we hope to apply our program to non-linear distortions to help with other robotic tasks in RoboCup at Home or with the BWI robot. While this project was a general observation on image distortions, applying it to logo detection would allow for potential progress on person following as well as object recognition.

REFERENCES

[1] J. J. McConnell, *Computer Graphics: Theory into Practice*, 1st ed. Jones & Bartlett Learning, ISBN 0-7637-2250-2, 2006.

[2] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.