Lab Description:

Image rectification is the process of taking a perspective image and altering by ways of a perspective matrix to a rectangle or square. This lab is meant to reinforce the basic data structures, variables, and methods that you have learned thus far. You must have the image processing toolbox from Matlab installed.

The image shown above has the corner points labeled with the pixel coordinates shown. We can use this information to associate the vanishing points to the points at infinity. As a rule, the cross product of points give the equations of the lines that connect them! (remember that points are actually vectors in homogenous coordinates) Also the cross product of lines give the points of intersection.
**Step 1.** Create a function called “imrectification” that returns a matrix \( M \) corresponding to the projective transformation matrix and takes as input a variable “imname” that corresponds to the string filename of the connect 4 image. Read in the image using imread and call it “im”.

\[ \text{im = imread(imname);} \]

**Step 2.** Create variables p1 through p4 that store the pixel coordinates. Ensure that these are homogenous coordinates.

\[ p1 = [153; 84; 1]; \]

\[ \text{do } p2, p3, \text{ etc.} \]

**Step 3.** Compute the equation of a line that crosses through p1 and p2 and store that in a variable named l1. Do the same for all 4 lines. (hint: You may use either the “cross” command in Matlab or the equation of the cross product.)

**Step 4.** Compute the intersection of lines l1 and l2 and store that in a variable int1. Remember that the intersection of these lines is the vanishing point. Do the same for lines l3 and l4 and store it in a variable int2.

**Step 5.** Setup a system of linear equations that can solve the projection matrix, \( M \). The key here is to realize what the matrix is doing. Think about the vanishing point of a perfect square, where is the point of intersection of the lines? At infinity, because there is no intersection of parallel lines. Now think about our perspective image. There are vanishing points of l1 and l2 and of l3 and l4, (denoted by int1 and int2 above); however, we want to make these lines parallel. Thus, we are going to create a projective matrix that bends parallel lines to match the perspective of our current image...it will bend parallel lines to intersect at int1 and int2. Once we have this matrix, we can invert it! And create a projective matrix that bends our image back to parallel. \( M \) is equal to the 3x3 projection matrix.

\[ M \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \text{int1(1)} \\ \text{int1(2)} \\ \text{int1(3)} \end{bmatrix} \]

\[ M \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \text{int2(1)} \\ \text{int2(2)} \\ \text{int2(3)} \end{bmatrix} \]

We can now define a concrete point in cartesian space, \( [0 \ 0 \ 1] \), as simply one of our points. Let’s choose p1.

\[ M \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 153 \\ 84 \\ 1 \end{bmatrix} \]

Finally, to help us solve the matrix, we will define one more concrete point in cartesian space, \( [1 \ 1 \]
As we know, in a matrix-vector multiplication, we are doing a linear combination of the rows. For example,

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{bmatrix}
\begin{bmatrix}
\alpha \\
\beta \\
\gamma
\end{bmatrix}
= 
\begin{bmatrix}
\alpha + \beta + \gamma \\
\alpha + \beta + \gamma \\
\alpha + \beta + \gamma
\end{bmatrix}
\]

Verify that this is true on your own.

**Step 6.** We need to solve for the coefficients \(\alpha, \beta, \gamma\) from above. Our matrix is a combination of intersection1, intersection2, and point 1. Our above matrix can be represented as the following,

\[
P = \begin{bmatrix} int1 & int2 & p1 \end{bmatrix} \cdot 
\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = 
\begin{bmatrix} 443 \\ 296 \\ 1 \end{bmatrix}
\]

Solve for \(\alpha, \beta, \gamma\) using the inverse of \(P\),

\[
\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = inv(P) \cdot 
\begin{bmatrix} 443 \\ 296 \\ 1 \end{bmatrix}
\]

Your final M matrix should be your \(P\) matrix multiplied by your coefficients.

\[
M = \begin{bmatrix} \alpha_{int1} & \beta_{int2} & \gamma_{p1} \end{bmatrix}
\]

Finally, remember that this projective matrix bends parallel lines towards the intersections. You actually want to bend lines back to parallel. Thus you want to make a projective transformation that is the inverse of \(M\). Use the following to create your final M matrix. \(M = inv(M)'\); Note that Matlab wants the transpose of this matrix, hence the \\

Show the transformed image by creating a transformation object in Matlab,

\[
>> T = maketform('projective',M)
\]

finally transform the image named “im”.

1] as p4.
\[ \text{>> im2 = imtransform(im, T, 'size', size(im))} \]
\[ \text{>> imshow(im2)} \]

**Deliverables:** Submit on Blackboard.

(a) Original image.  
(b) Rectified.

Figure 2: Example of image rectification on a license plate.