Perspective Transformation

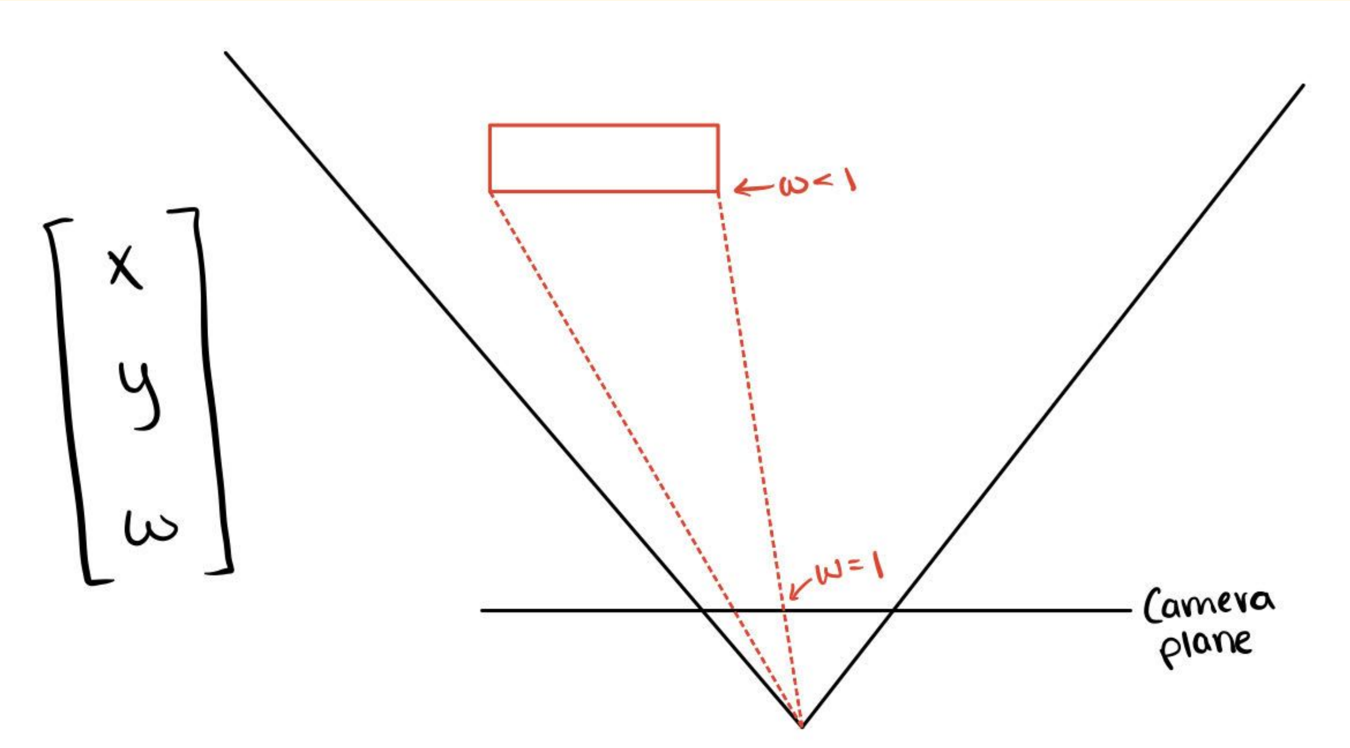
John Trager and Daniel Stefanescu

Overview

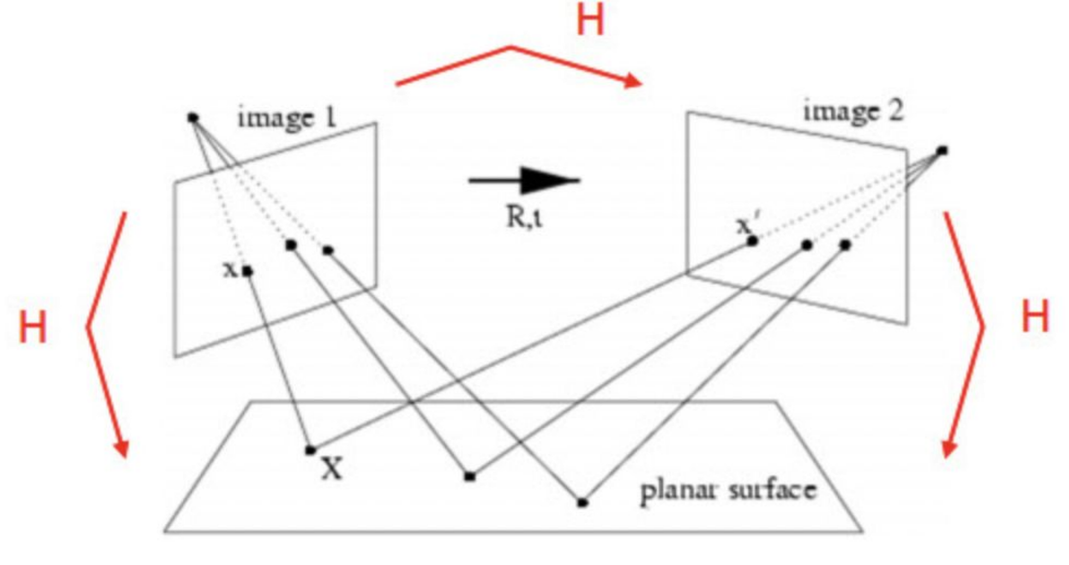
- Isolate Sub-image

- Translate to correct perspective

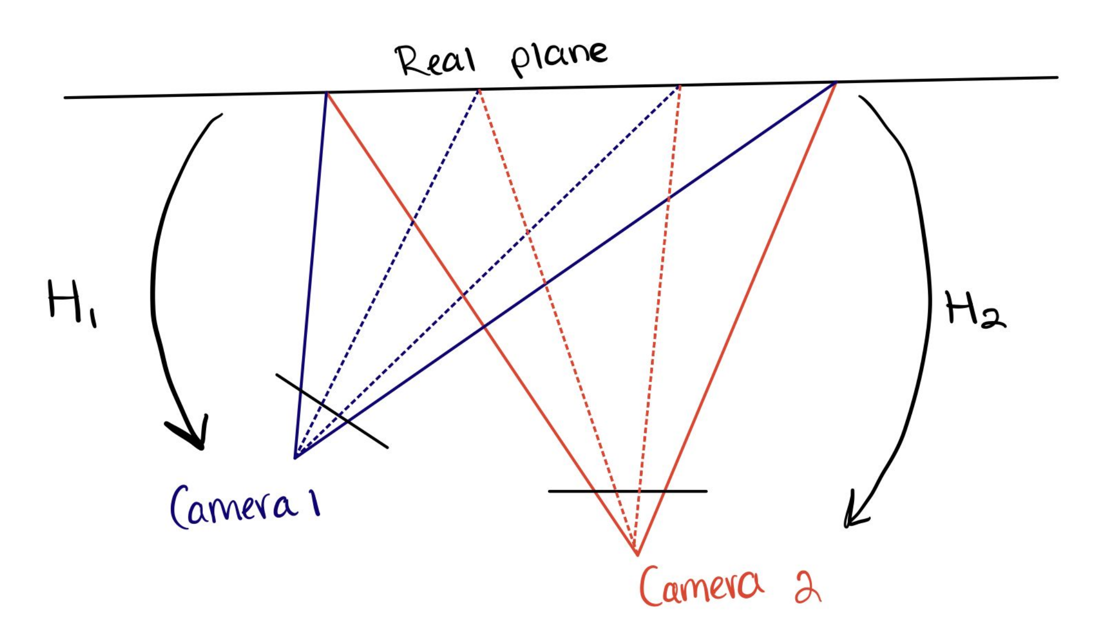
Homogeneous Coordinates



Homography Transformation



Homography Transformation



Camera → Camera 2

$$H_1^{-1} \cdot H_2 = H_{n \times n}$$

$$\begin{pmatrix} x_2 \\ y_2 \\ w_2 \end{pmatrix} = H \cdot \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x_2 \\ y_2 \\ w_2 \end{pmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$x' = \frac{x_2}{w_2} = \frac{H_{11} \cdot x_1 + H_{12} \cdot y_1 + H_{13}}{H_{31} \cdot x_1 + H_{32} \cdot y_1 + H_{33}}$$

$$y' = \frac{y_2}{w_2} = \frac{H_{21} \cdot x_1 + H_{22} \cdot y_1 + H_{23}}{H_{31} \cdot x_1 + H_{32} \cdot y_1 + H_{33}}$$

$$-H_{11}\cdot x_1 - H_{12}\cdot y_1 - H_{13} + H_{31}\cdot x_1' \cdot x_1 + H_{32}\cdot x_1' \cdot y_1 + H_{33}\cdot x_1' = 0$$

$$-H_{21}\cdot x_1 - H_{22} y_1 - H_{23} + H_{31}\cdot y_1' \cdot x_1 + H_{32}\cdot y_1' \cdot y_1 + H_{33}\cdot y_1' = 0$$

$$\begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1 x_1' & y_1 x_1' & x_1' \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1 y_1' & y_1 y_1' & y_1' \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2 x_2' & y_2 x_2' & x_2' \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2 y_2' & y_2 y_2' & y_2' \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3 x_3' & y_3 x_3' & x_3' \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3 y_3' & y_3 y_3' & y_3' \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4 x_4' & y_4 x_4' & x_4' \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4 y_4' & y_4 y_4' & y_4' \end{bmatrix} \begin{bmatrix} h1 \\ h2 \\ h3 \\ h4 \\ h5 \\ h6 \\ h7 \\ h8 \\ h9 \end{bmatrix} = 0$$

Final Transformation Matrix

$$\begin{bmatrix} x'/\lambda \\ y'/\lambda \\ \lambda \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Our Implementation

$$\begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x'_1 & y_1x'_1 & x'_1 \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1y'_1 & y_1y'_1 & y'_1 \\ -x_2 & -y_2 & -1 & 0 & 0 & 0 & x_2x'_2 & y_2x'_2 & x'_2 \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2y'_2 & y_2y'_2 & y'_2 \\ -x_3 & -y_3 & -1 & 0 & 0 & 0 & x_3x'_3 & y_3x'_3 & x'_3 \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3y'_3 & y_3y'_3 & y'_3 \\ -x_4 & -y_4 & -1 & 0 & 0 & 0 & x_4x'_4 & y_4x'_4 & x'_4 \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4y'_4 & y_4y'_4 & y'_4 \end{bmatrix}$$

Least squares (H mat):

$$\widehat{x} = (A^T A)^{-1} A^T b.$$

```
A = np.mat([[-a1.x,-a1.y,-1,0,0,0, a1.x*b1.x, a1.y*b1.x, b1.x],
            [0,0,0,-a1.x,-a1.y,-1, a1.x*b1.y, a1.y*b1.y, b1.y],
            [-a2.x,-a2.y,-1,0,0,0, a2.x*b2.x, a2.y*b2.x, b2.x],
            [0,0,0,-a2.x,-a2.y,-1, a2.x*b2.y, a2.y*b2.y, b2.y],
            [-a3.x,-a3.y,-1,0,0,0, a3.x*b3.x, a3.y*b3.x, b3.x],
            [0,0,0,-a3.x,-a3.y,-1, a3.x*b3.y, a3.y*b3.y, b3.y],
            [-a4.x,-a4.y,-1,0,0,0, a4.x*b4.x, a4.y*b4.x, b4.x],
            [0,0,0,-a4.x,-a4.y,-1, a4.x*b4.y, a4.y*b4.y, b4.y],
            [0,0,0,0,0,0,0,0,1]
           ])

# b vector
b = np.array([0,0,0,0,0,0,0,0,1])

# (A^T * A)^-1
C = np.linalg.inv(np.matmul(np.transpose(A),A))
# (A^T * A)^-1 * A^T
C = np.matmul(C,np.transpose(A))

# (A^T * A)^-1 * A^T * b
self.H = np.matmul(C,b)

# shape H matrix to 3x3
self.H = np.reshape(self.H,(3,3))
```

Our Implementation

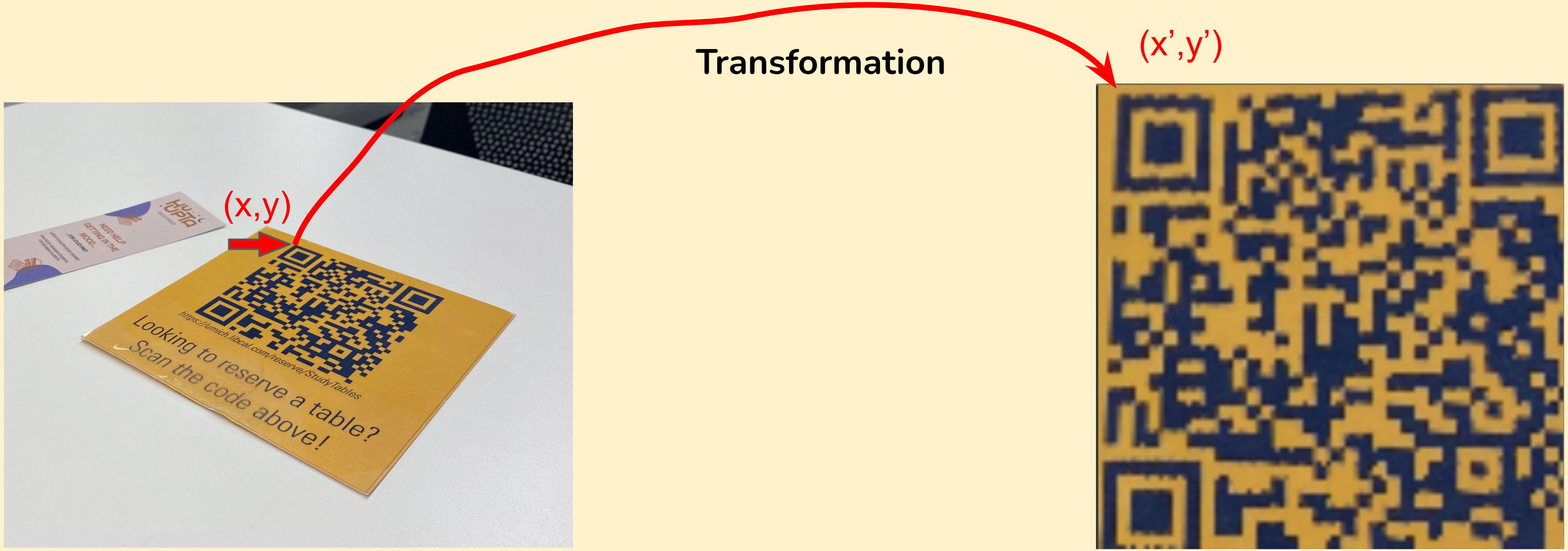
$$\begin{bmatrix} x'/\lambda \\ y'/\lambda \\ \lambda \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

```python
def transform_point(self,p: Point):
        if (self.H is not None):
            x = np.mat([p.x,p.y,1]).reshape((3,1))
            # H matrix * (x,y,1) vector
            x = np.matmul(self.H,x)
            # divide output by z value (to scale)
            ret = x/x[2]
            return ret # np mat size (3,1)

        else:
            print("Error: H matrix not generated")
            exit(1)
```

Our Implementation

```
# pseudo code
def transform_image():

        # make new image (blank image of same shape as img)
        img_transform = np.zeros(img.shape)

        # do transform (loop through each pixel in img)
        for i from 0 to img.height:
            for j from 0 to img.width:
                # transform point
                tp = transform_point(Point(j,i))

                if (tp is inside img_transform shape):
                    img_transform[tp.x][tp.y] = self.img[i][j]

       # other post-processing later (to crop image and remove image noise)
```

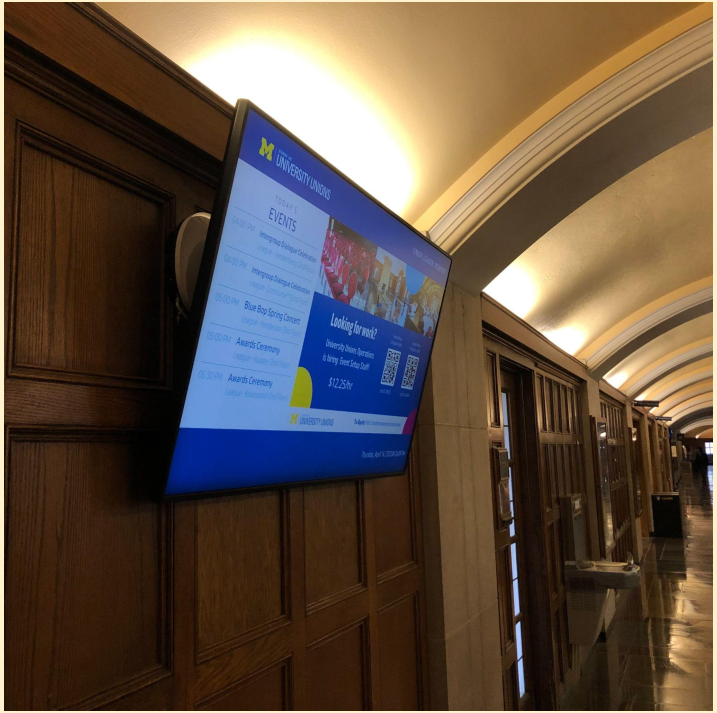
Example

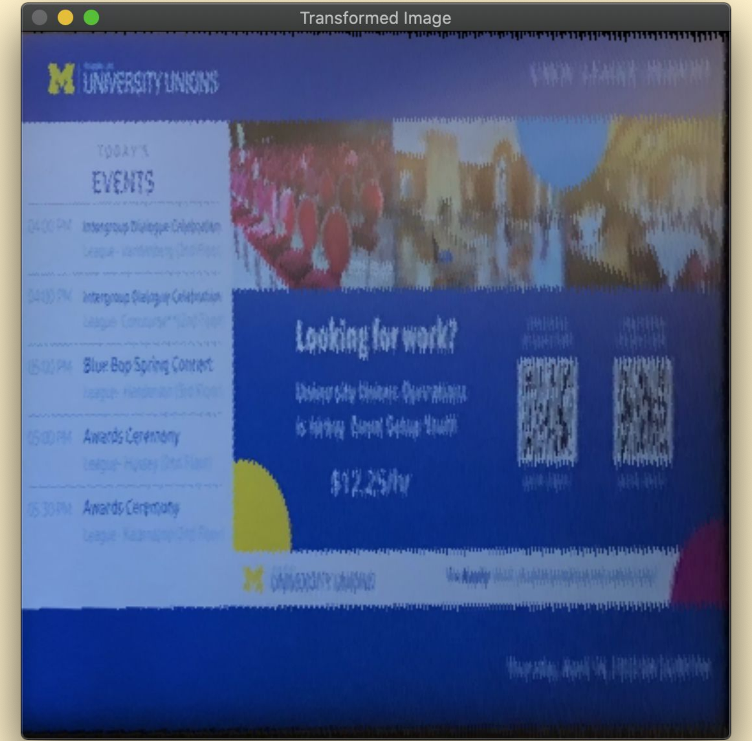

(x,y)
Transformation
(x',y')
https://umich.libcal.com/reserve/StudyTables
Looking to reserve a table?
Scan the code above!

Results

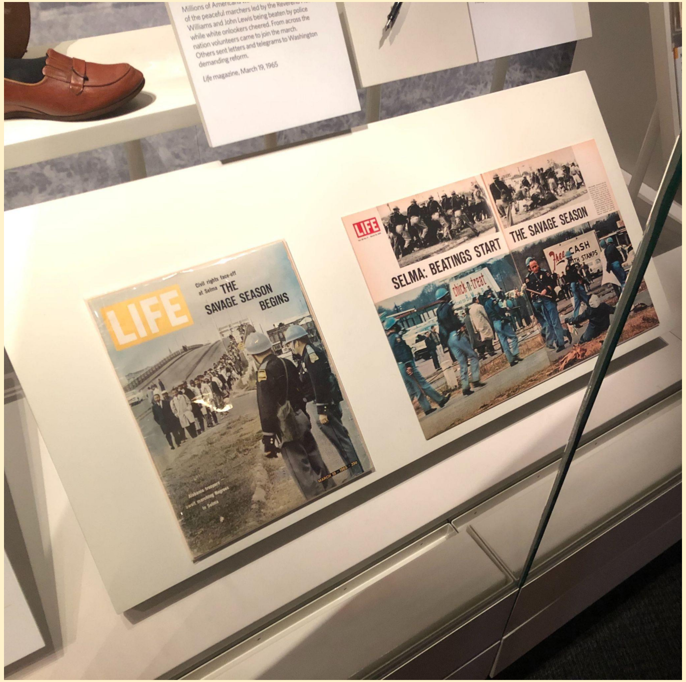

https://umich.libcal.com/reserve/StudyTables
Looking to reserve a table?
Scan the code above!
NEED HELP
GETTING IN THE
MOOD...

Results


UNIVERSITY UNIONS
EVENTS
Blue Bop Spring Concert
Awards Ceremony
Looking for work?
$12.25/hr


Transformed Image
UNIVERSITY UNIONS
TODAY'S
EVENTS
Intergroup Dialogue Celebration
Blue Bop Spring Concert
Awards Ceremony
Looking for work?
$12.25/hr

Results

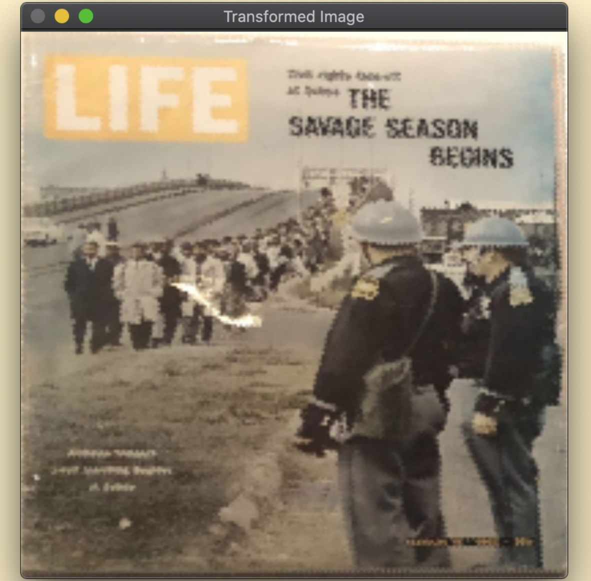

LIFE
Civil rights face-off at Selma THE SAVAGE SEASON BEGINS
SELMA: BEATINGS START
THE SAVAGE SEASON


Transformed Image
LIFE
THE SAVAGE SEASON BEGINS

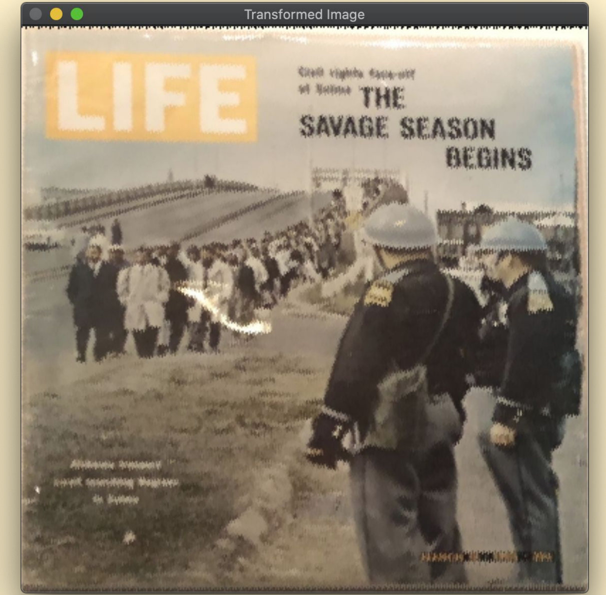
Results - Hyperparameter Tuning

Compression: 4:8

Compression: 4:4



Compression: 4:2



Pros

- Working algorithm
- Does not use outside functions for anything other than image loading/displaying and matrix calculations

Cons

- Slow compared to pre existing methods
  - Multi-threading or gpu programming (beyond scope)
- Information is lost in transform (better methods such as pixel unwrapping)

Thank You!

John Trager and Daniel Stefanescu

Sources

https://www.cs.toronto.edu/~jepson/csc420/notes/imageProjection.pdf

https://medium.com/analytics-vidhya/opencv-perspective-transformation-9edffefb2143

https://www.educba.com/opencv-perspectivetransform/

Basic concepts of the homography explained with code

Homography Estimation∗

Lecture 16: Planar Homographies

Lecture 20: The Eight-Point Algorithm

http://www.cs.cornell.edu/courses/cs6670/2011sp/lectures/lec07_panoramas.pdf