# Assignment 05: Panorama

Brock Davis

# Input Images

image1.png

image2.png

image3.png

image4.png

image5.png

Dimensions: 1920x1080 px

# Program Loop

```
while len(imgs)>1:
    new_imgs=[]
    length=len(imgs)
    print length
    for i in range(0,length,2):
      if i==length-1:
          new_imgs.append(imgs[i])
      else:
          new_img=combineImgs(imgs[i],imgs[i+1],matricies[i],count)
          count=count+1
          new_imgs.append(new_img)


    imgs=list(new_imgs)
    matricies=getMatricies(imgs,blur)

funcs.save(imgs[0],"output/pic1.png")
```

# getMatricies function

```
def getMatricies(imgs,blur):

    matricies=[]

    for i in range(len(imgs)):
     if i>0:
         M=getMatrix(imgs[i-1],imgs[i],blur)
          matricies.append(M)
    return matricies
```

# combineImgs function

```
def combineImgs(img1,img2,M,count):
    h,w=img1.shape[:2]
    corners=np.float32([[[0,0],[w,0],[0,h],[w,h]]])
    corners=cv2.perspectiveTransform(corners,M)
    corners=corners[0]
    minX=min(corners[:,0])
    maxX=max(corners[:,0])
    minY=min(corners[:,1])
    maxY=max(corners[:,1])
    M2=np.float32([[1, 0,-minX if minX<0 else 0],
                   [0, 1,-minY if minY<0 else 0],
                   [0, 0, 1]])
    warp_dim_x=int(abs(minX) + (abs(maxX) if maxX > w else w))
    warp_dim_y=int(abs(minY) + (abs(maxY) if maxY > h else h))
    img1_warp=cv2.warpPerspective(img1,M2.dot(M),(warp_dim_x, warp_dim_y))
    img2_warp=cv2.warpPerspective(img2,M2,(warp_dim_x, warp_dim_y))
    mask=np.float32(img1_warp>0)
    kernel=np.float32([[1,1,1]])
    mask=cv2.erode(mask,kernel)
    out=mask*img1_warp+(1-mask)*img2_warp
    out=funcs.normalize_pic(out)
    funcs.save(out,('output/prog_%d.png'%count))
    out=autoCrop(out)
    return out
```

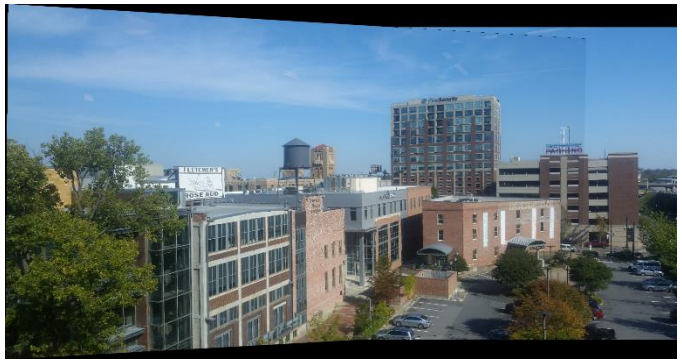
# autoCrop function

```
def autoCrop(img):
    img=cutSide(img)
    h,w=img.shape[:2]
    cutoff=w*3*.9    while True:
     if np.count_nonzero(img[0])<cutoff:
         img=img[1:]
     else:
         while True:
          if np.count_nonzero(img[-1])<cutoff:
              img=img[:-1]
          else:
              return img
```

# cutSide function

```
def cutSide(img):
    img=img.copy()
    while True:
     if np.count_nonzero(img[:,-1])==0:
         img=img[:,:-1]
     else:
         while True:
          if np.count_nonzero(img[:,0])==0:
              img=img[:,1:]
          else:
              return img
```

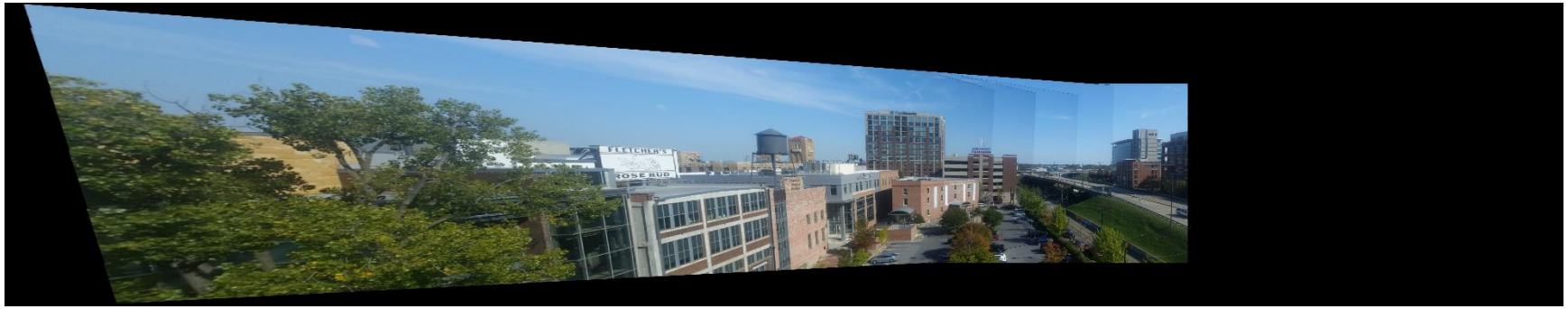
# Progress Pictures



prog_0.png

prog_1.png

prog_2.png



prog_3.png

# Final Panorama


Dimensions: 7058x1081 px

The built in autoCrop function could not handle the side of the final picture, so it was manually cropped to have no black area.


Dimensions: 6561x1081 px