Project_5

April 27, 2020

1 Programming Project #5: Video Stitching and Processing

1.1 CS445: Computational Photography - Spring 2020

1.1.1 Part I: Stitch two key frames

This involves:

1. compute homography H between two frames;
2. project each frame onto the same surface;
3. blend the surfaces.

Check that your homography is correct by plotting four points that form a square in frame 270 and their projections in each image, like this:

[1]:
```
import cv2
import numpy as np
from numpy.linalg import svd, inv
import utils
%matplotlib inline
from matplotlib import pyplot as plt
```

[2]:
```
# images location
im1 = './images/input/frames/f0270.jpg'
im2 = './images/input/frames/f0450.jpg'

# Load an color image in grayscale
im1 = cv2.imread(im1)
im2 = cv2.imread(im2)

im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2RGB)
```

[3]:
```
def auto_homography(Ia,Ib, homography_func=None,normalization_func=None):
    '''
    Computes a homography that maps points from Ia to Ib

    Input: Ia and Ib are images
    Output: H is the homography
```

```
    '''
    if Ia.dtype == 'float32' and Ib.dtype == 'float32':
        Ia = (Ia*255).astype(np.uint8)
        Ib = (Ib*255).astype(np.uint8)

    Ia_gray = cv2.cvtColor(Ia,cv2.COLOR_BGR2GRAY)
    Ib_gray = cv2.cvtColor(Ib,cv2.COLOR_BGR2GRAY)

    # Initiate SIFT detector
    sift = cv2.xfeatures2d.SIFT_create()

    # find the keypoints and descriptors with SIFT
    kp_a, des_a = sift.detectAndCompute(Ia_gray,None)
    kp_b, des_b = sift.detectAndCompute(Ib_gray,None)

    # BFMatcher with default params
    bf = cv2.BFMatcher()
    matches = bf.knnMatch(des_a,des_b, k=2)

    # Apply ratio test
    good = []
    for m,n in matches:
        if m.distance < 0.75*n.distance:
            good.append(m)

    numMatches = int(len(good))

    matches = good

    # Xa and Xb are 3xN matrices that contain homogeneous coordinates for the N
    # matching points for each image
    Xa = np.ones((3,numMatches))
    Xb = np.ones((3,numMatches))

    for idx, match_i in enumerate(matches):
        Xa[:,idx][0:2] = kp_a[match_i.queryIdx].pt
        Xb[:,idx][0:2] = kp_b[match_i.trainIdx].pt

    ## RANSAC
    niter = 1000
    best_score = 0

    for t in range(niter):
        # estimate homography
        subset = np.random.choice(numMatches, 4, replace=False)
        pts1 = Xa[:,subset]
        pts2 = Xb[:,subset]
```

```
        H_t = homography_func(pts1, pts2, normalization_func) # edit helper␣
↪code below (computeHomography)

        # score homography
        Xb_ = np.dot(H_t, Xa) # project points from first image to second using␣
↪H
        du = Xb_[0,:]/Xb_[2,:] - Xb[0,:]/Xb[2,:]
        dv = Xb_[1,:]/Xb_[2,:] - Xb[1,:]/Xb[2,:]

        ok_t = np.sqrt(du**2 + dv**2) < 1  # you may need to play with this␣
↪threshold
        score_t = sum(ok_t)

        if score_t > best_score:
            best_score = score_t
            H = H_t
            in_idx = ok_t

    print('best score: {:02f}'.format(best_score))

    # Optionally, you may want to re-estimate H based on inliers

    return H
```

[4]:
```
def computeHomography(pts1, pts2,normalization_func=None):
    '''
    Compute homography that maps from pts1 to pts2 using SVD

    Input: pts1 and pts2 are 3xN matrices for N points in homogeneous
    coordinates.

    Output: H is a 3x3 matrix, such that pts2~=H*pts1
    '''
    #raise Exception("TODO in computeHomography() not implemented")
    N = pts1.shape[1]
    A = np.zeros((2*N, 9))

    for i in range(N):
        x = pts1[:, i]
        y = pts2[:, i]
        u = x[0]
        v = x[1]
        u2 = y[0]
        v2 = y[1]
        A[2*i,:] = np.array([-u, -v, -1, 0, 0, 0, u*u2, v*u2, u2])
        A[2*i+1, :] = np.array([0, 0, 0, -u, -v, -1, u*v2, v*v2, v2])
```

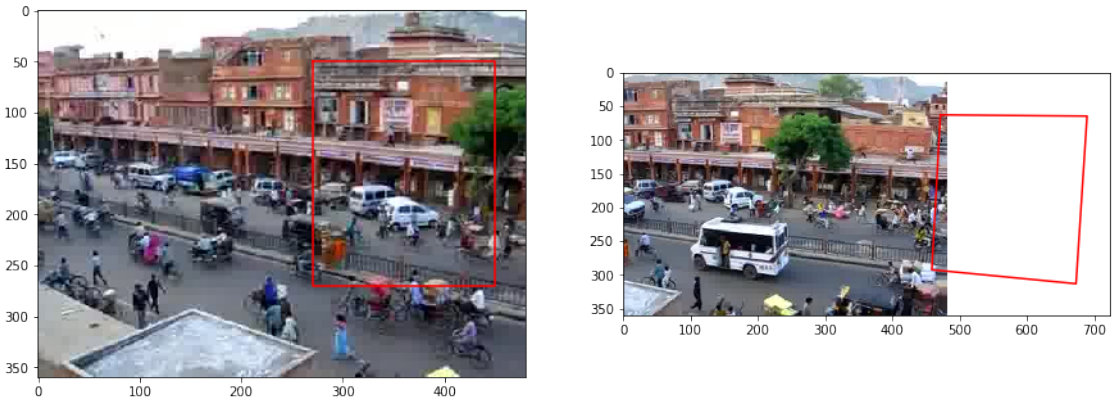
```
    a,b,vh = np.linalg.svd(A)
    h = vh[vh.shape[0]-1,:]
    h = np.reshape(h, (3,3))
    return h
```

[5]:
```
H = auto_homography(im2,im1, computeHomography)
```

```
best score: 155.000000
```

[6]:
```
# Checking
plotA = np.array([270, 450, 450, 270, 270])
plotB = np.array([50, 50, 270, 270, 50])
ones = np.ones(5)
plotC = np.array([plotA, plotB, ones])
plotD = np.dot(H, plotC)
plotD = plotD / plotD[-1]

fig, axes = plt.subplots(1, 2)
fig.set_size_inches(15, 7)

axes[0].imshow(im1)
axes[0].plot(plotA, plotB, 'r') #use red instead of blue
axes[1].imshow(im2)
axes[1].plot(plotD[0], plotD[1], 'r')
```

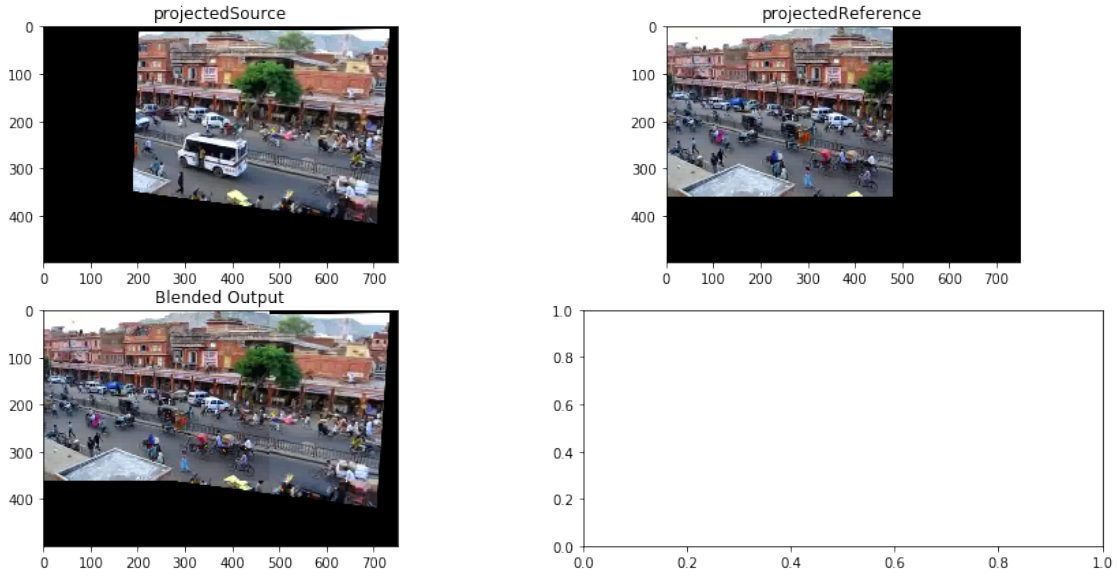
[6]:
```
[<matplotlib.lines.Line2D at 0x1192ec690>]
```



[7]:
```
# # Example usage of cv2.warpPerspective
# projectedWidth = 1600
# projectedHeight = 500

projectedWidth = 750
```

```
projectedHeight = 500

sourceHomography = H
refHomography = np.identity(3)

projectedSource = cv2.warpPerspective(im2, sourceHomography, (projectedWidth,␣
 ↪projectedHeight))
projectedReference = cv2.warpPerspective(im1, refHomography, (projectedWidth,␣
 ↪projectedHeight))

fig, axes = plt.subplots(2, 2)
fig.set_size_inches(15, 7)

axes[0, 0].imshow(projectedSource)
axes[0, 0].set_title("projectedSource")
axes[0, 1].imshow(projectedReference)
axes[0, 1].set_title("projectedReference")

# Example usage of utils.blendImages
blendedOutput = utils.blendImages(projectedSource, projectedReference)
# blendedOutput = cv2.cvtColor(blendedOutput, cv2.COLOR_BGR2RGB)
# plt.figure()
# plt.imshow(blendedOutput)
axes[1, 0].imshow(blendedOutput)
axes[1, 0].set_title("Blended Output")

# cv2.imwrite("part1_projectedSource.jpg", projectedSource)
# cv2.imwrite("part1_projectedReference.jpg", projectedReference)
# cv2.imwrite("part1_blended.jpg", blendedOutput)
```

[7]: `Text(0.5, 1.0, 'Blended Output')`



1.1.2 Part II: Panorama using five key frames

In this part you will produce a panorama using five key frames. Let's determine frames [90, 270, 450, 630, 810] as key frames. The goal is to map all the five frames onto the plane corresponding to frame 450 (that we also call the *reference frame*). For the frames 270 and 630 you can follow the instructions in part 1.

Mapping frame 90 to frame 450 is difficult because they share very little area. Therefore you need to perform a two stage mapping by using frame 270 as a guide. Compute one projection from 90 to 270 and one from 270 to 450 and multiply the two matrices. This produces a projection from 90 to 450 even though these frames have very little area in common

[8]:
```
import cv2
import numpy as np
```

[9]:
```
master_frames =[90, 270, 450, 630, 810]
reference_frame = 450
reference_idx = master_frames.index(reference_frame)

im1 = cv2.imread('./images/input/frames/f0090.jpg')
im2 = cv2.imread('./images/input/frames/f0270.jpg')
im3 = cv2.imread('./images/input/frames/f0450.jpg')
im4 = cv2.imread('./images/input/frames/f0630.jpg')
im5 = cv2.imread('./images/input/frames/f0810.jpg')

im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2RGB)
im3 = cv2.cvtColor(im3, cv2.COLOR_BGR2RGB)
im4 = cv2.cvtColor(im4, cv2.COLOR_BGR2RGB)
```

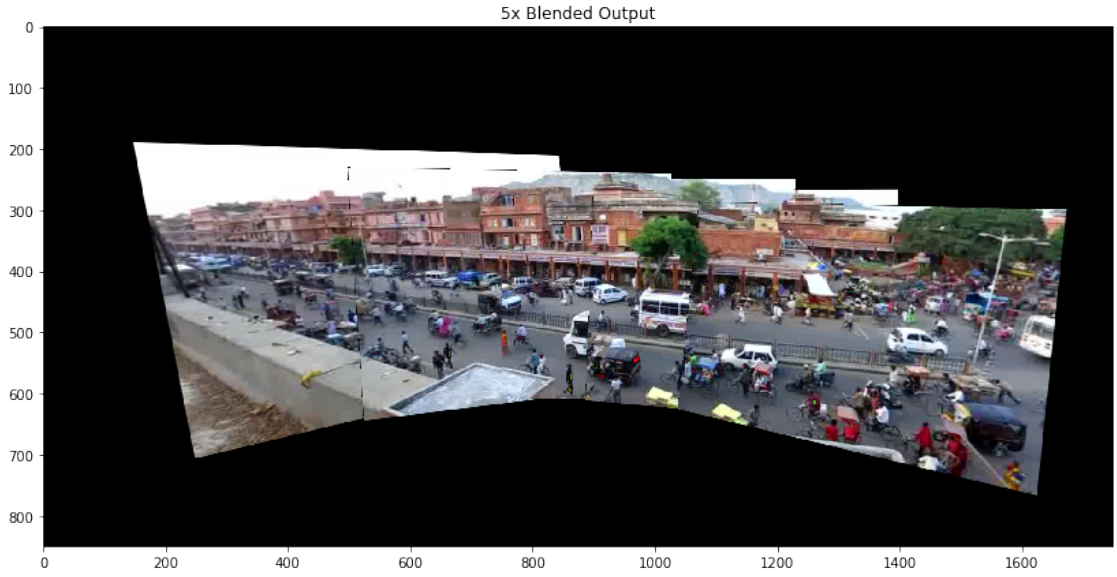
```
im5 = cv2.cvtColor(im5, cv2.COLOR_BGR2RGB)
```

[10]:
```
# projectedWidth = 750
# projectedHeight = 500

# sourceHomography = H
# refHomography = np.identity(3)

# projectedSource = cv2.warpPerspective(im2, sourceHomography, (projectedWidth,␣
 ↪projectedHeight))
# projectedReference = cv2.warpPerspective(im1, refHomography, (projectedWidth,␣
 ↪projectedHeight))
# blendedOutput = utils.blendImages(projectedSource, projectedReference)

fig, axes = plt.subplots(1, 1)
fig.set_size_inches(15, 7)

projectedWidth = 1750
projectedHeight = 850

H_12 = auto_homography(im1,im2, computeHomography)
H_23 = auto_homography(im2,im3, computeHomography)
H_34 = auto_homography(im4,im3, computeHomography)
H_45 = auto_homography(im5,im4, computeHomography)

H_t = np.array([[1, 0, 750], [0, 1, 250], [0, 0, 1]])
refHomography = np.identity(3)

H_13 = np.dot(H_23,H_12)
H_35 = np.dot(H_45,H_34)

warped_1 = cv2.warpPerspective(im1, H_t.dot(H_13), (projectedWidth,␣
 ↪projectedHeight))
warped_2 = cv2.warpPerspective(im2, H_t.dot(H_23), (projectedWidth,␣
 ↪projectedHeight))
warped_3 = cv2.warpPerspective(im3, H_t.dot(refHomography), (projectedWidth,␣
 ↪projectedHeight))
warped_4 = cv2.warpPerspective(im4, H_t.dot(H_34), (projectedWidth,␣
 ↪projectedHeight))
warped_5 = cv2.warpPerspective(im5, H_t.dot(H_35), (projectedWidth,␣
 ↪projectedHeight))

blend_1 = utils.blendImages(warped_1, warped_2)
blend_2 = utils.blendImages(blend_1, warped_3)
blend_3 = utils.blendImages(blend_2, warped_4)
blend_4 = utils.blendImages(blend_3, warped_5)
```

```
axes.imshow(blend_4)
axes.set_title("5x Blended Output")
cv2.imwrite("part2.jpg", blend_4)
```

```
best score: 190.000000
best score: 167.000000
best score: 128.000000
best score: 94.000000
```

[10]: True



1.1.3 Part 3: Map the video to the reference plane

[11]:
```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from math import floor

import utils
```

[12]:
```
dir_frames = 'images/input/frames'
filenames = []
filesinfo = os.scandir(dir_frames)
```

[13]:
```
filenames = [f.path for f in filesinfo if f.name.endswith(".jpg")]
filenames.sort(key=lambda f: int(''.join(filter(str.isdigit, f))))
```

[14]:
```
frameCount = len(filenames)
frameHeight, frameWidth, frameChannels = cv2.imread(filenames[0]).shape
frames = np.zeros((frameCount, frameHeight, frameWidth, frameChannels),dtype=np.
↪float32)
```

[15]:
```
for idx, file_i in enumerate(filenames):
    frames[idx] = cv2.cvtColor(cv2.imread(file_i), cv2.COLOR_BGR2RGB) / 255.0
print("Number of Pictures to Generate", len(frames)) #User line check
```

```
Number of Pictures to Generate 900
```

[16]:
```
# Variables from Part 2 Still Being Used
# im1 = cv2.imread('./images/input/frames/f0090.jpg')
# im2 = cv2.imread('./images/input/frames/f0270.jpg')
# im3 = cv2.imread('./images/input/frames/f0450.jpg')
# im4 = cv2.imread('./images/input/frames/f0630.jpg')
# im5 = cv2.imread('./images/input/frames/f0810.jpg')

# im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
# im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2RGB)
# im3 = cv2.cvtColor(im3, cv2.COLOR_BGR2RGB)
# im4 = cv2.cvtColor(im4, cv2.COLOR_BGR2RGB)
# im5 = cv2.cvtColor(im5, cv2.COLOR_BGR2RGB)

# projectedWidth = 1750
# projectedHeight = 850

# H_12 = auto_homography(im90,im270, computeHomography)
# H_23 = auto_homography(im270,im450, computeHomography)
# H_34 = auto_homography(im630,im450, computeHomography)
# H_45 = auto_homography(im810,im630, computeHomography)

# H_13 = np.dot(H_12,H_23)
# H_35 = np.dot(H_45,H_34)

# H_t = np.array([[1, 0, 750], [0, 1, 250], [0, 0, 1]]) #Used from Above

os.mkdir("aligned_frames")

pastHomographies = np.zeros((len(filenames),len(filenames), 3, 3),dtype=np.
↪float32)

# Run through each file
for idx, file_i in enumerate(filenames):
    img = cv2.imread(file_i)
    if idx < 90:
        H = auto_homography(img,im1, computeHomography)
```

```
        H = np.dot(H,H_13)
        pastHomographies[idx] = H
        warped_1 = cv2.warpPerspective(img, np.dot(H_t,H), (projectedWidth,␣
↪projectedHeight))
        cv2.imwrite("aligned_frames/a{:04d}.jpg".format(idx), warped_1)
    elif (idx < 270) and (idx >= 90):
        H = auto_homography(img,im2, computeHomography)
        H = np.dot(H,H_23)
        pastHomographies[idx] = H
        warped_2 = cv2.warpPerspective(img, np.dot(H_t,H), (projectedWidth,␣
↪projectedHeight))
        cv2.imwrite("aligned_frames/a{:04d}.jpg".format(idx), warped_2)
    elif (idx >= 270) and (idx <= 630):
        H = auto_homography(img,im3, computeHomography)
        pastHomographies[idx] = H
        warped_3 = cv2.warpPerspective(img, np.dot(H_t,H), (projectedWidth,␣
↪projectedHeight))
        cv2.imwrite("aligned_frames/a{:04d}.jpg".format(idx), warped_3)
    elif (idx > 630) and (idx <= 810):
        H = auto_homography(img,im4, computeHomography)
        H = np.dot(H,H_34)
        pastHomographies[idx] = H
        warped_4 = cv2.warpPerspective(img, np.dot(H_t,H), (projectedWidth,␣
↪projectedHeight))
        cv2.imwrite("aligned_frames/a{:04d}.jpg".format(idx), warped_4)
    elif idx > 810:
        H = auto_homography(img,im5, computeHomography)
        H = np.dot(H,H_45)
        pastHomographies[idx] = H
        warped_5 = cv2.warpPerspective(img, np.dot(H_t,H), (projectedWidth,␣
↪projectedHeight))
        cv2.imwrite("aligned_frames/a{:04d}.jpg".format(idx), warped_5)
```

```
best score: 363.000000
best score: 356.000000
best score: 365.000000
best score: 353.000000
best score: 384.000000
best score: 383.000000
best score: 397.000000
best score: 404.000000
best score: 407.000000
best score: 417.000000
best score: 415.000000
best score: 409.000000
best score: 413.000000
best score: 401.000000
```

```
best score: 422.000000
best score: 429.000000
best score: 446.000000
best score: 429.000000
best score: 419.000000
best score: 418.000000
best score: 401.000000
best score: 423.000000
best score: 418.000000
best score: 407.000000
best score: 431.000000
best score: 412.000000
best score: 394.000000
best score: 428.000000
best score: 443.000000
best score: 428.000000
best score: 422.000000
best score: 454.000000
best score: 437.000000
best score: 427.000000
best score: 444.000000
best score: 456.000000
best score: 431.000000
best score: 443.000000
best score: 435.000000
best score: 459.000000
best score: 458.000000
best score: 450.000000
best score: 458.000000
best score: 458.000000
best score: 446.000000
best score: 419.000000
best score: 461.000000
best score: 458.000000
best score: 450.000000
best score: 468.000000
best score: 451.000000
best score: 473.000000
best score: 464.000000
best score: 475.000000
best score: 482.000000
best score: 463.000000
best score: 477.000000
best score: 497.000000
best score: 476.000000
best score: 491.000000
best score: 484.000000
best score: 475.000000
```

```
best score: 502.000000
best score: 491.000000
best score: 532.000000
best score: 501.000000
best score: 513.000000
best score: 555.000000
best score: 518.000000
best score: 534.000000
best score: 510.000000
best score: 524.000000
best score: 558.000000
best score: 521.000000
best score: 550.000000
best score: 550.000000
best score: 542.000000
best score: 537.000000
best score: 602.000000
best score: 647.000000
best score: 575.000000
best score: 537.000000
best score: 607.000000
best score: 609.000000
best score: 683.000000
best score: 718.000000
best score: 780.000000
best score: 819.000000
best score: 987.000000
best score: 2525.000000
best score: 190.000000
best score: 184.000000
best score: 170.000000
best score: 186.000000
best score: 180.000000
best score: 165.000000
best score: 194.000000
best score: 186.000000
best score: 201.000000
best score: 180.000000
best score: 194.000000
best score: 169.000000
best score: 193.000000
best score: 208.000000
best score: 183.000000
best score: 173.000000
best score: 190.000000
best score: 152.000000
best score: 166.000000
best score: 166.000000
```

```
best score: 177.000000
best score: 163.000000
best score: 179.000000
best score: 171.000000
best score: 171.000000
best score: 193.000000
best score: 172.000000
best score: 152.000000
best score: 180.000000
best score: 163.000000
best score: 164.000000
best score: 177.000000
best score: 183.000000
best score: 160.000000
best score: 183.000000
best score: 170.000000
best score: 172.000000
best score: 183.000000
best score: 168.000000
best score: 162.000000
best score: 181.000000
best score: 147.000000
best score: 172.000000
best score: 172.000000
best score: 147.000000
best score: 150.000000
best score: 184.000000
best score: 164.000000
best score: 148.000000
best score: 178.000000
best score: 165.000000
best score: 172.000000
best score: 164.000000
best score: 168.000000
best score: 169.000000
best score: 181.000000
best score: 168.000000
best score: 177.000000
best score: 179.000000
best score: 189.000000
best score: 170.000000
best score: 179.000000
best score: 174.000000
best score: 170.000000
best score: 183.000000
best score: 164.000000
best score: 169.000000
best score: 177.000000
```

```
best score: 162.000000
best score: 174.000000
best score: 177.000000
best score: 187.000000
best score: 189.000000
best score: 169.000000
best score: 181.000000
best score: 172.000000
best score: 187.000000
best score: 221.000000
best score: 177.000000
best score: 152.000000
best score: 178.000000
best score: 189.000000
best score: 185.000000
best score: 205.000000
best score: 171.000000
best score: 174.000000
best score: 191.000000
best score: 182.000000
best score: 216.000000
best score: 170.000000
best score: 184.000000
best score: 207.000000
best score: 171.000000
best score: 199.000000
best score: 192.000000
best score: 217.000000
best score: 201.000000
best score: 187.000000
best score: 199.000000
best score: 180.000000
best score: 232.000000
best score: 203.000000
best score: 193.000000
best score: 218.000000
best score: 213.000000
best score: 228.000000
best score: 187.000000
best score: 217.000000
best score: 208.000000
best score: 207.000000
best score: 211.000000
best score: 210.000000
best score: 211.000000
best score: 206.000000
best score: 220.000000
best score: 209.000000
```

```
best score: 218.000000
best score: 228.000000
best score: 207.000000
best score: 212.000000
best score: 230.000000
best score: 197.000000
best score: 215.000000
best score: 204.000000
best score: 229.000000
best score: 220.000000
best score: 216.000000
best score: 195.000000
best score: 220.000000
best score: 233.000000
best score: 267.000000
best score: 243.000000
best score: 266.000000
best score: 239.000000
best score: 240.000000
best score: 215.000000
best score: 216.000000
best score: 226.000000
best score: 227.000000
best score: 239.000000
best score: 224.000000
best score: 239.000000
best score: 204.000000
best score: 225.000000
best score: 217.000000
best score: 211.000000
best score: 211.000000
best score: 210.000000
best score: 225.000000
best score: 250.000000
best score: 216.000000
best score: 251.000000
best score: 277.000000
best score: 249.000000
best score: 253.000000
best score: 236.000000
best score: 254.000000
best score: 263.000000
best score: 262.000000
best score: 285.000000
best score: 300.000000
best score: 295.000000
best score: 318.000000
best score: 298.000000
```

```
best score: 267.000000
best score: 300.000000
best score: 344.000000
best score: 337.000000
best score: 318.000000
best score: 290.000000
best score: 292.000000
best score: 332.000000
best score: 376.000000
best score: 328.000000
best score: 349.000000
best score: 375.000000
best score: 369.000000
best score: 402.000000
best score: 446.000000
best score: 2554.000000
best score: 144.000000
best score: 140.000000
best score: 121.000000
best score: 149.000000
best score: 154.000000
best score: 150.000000
best score: 156.000000
best score: 154.000000
best score: 138.000000
best score: 192.000000
best score: 168.000000
best score: 150.000000
best score: 158.000000
best score: 168.000000
best score: 162.000000
best score: 155.000000
best score: 177.000000
best score: 172.000000
best score: 178.000000
best score: 175.000000
best score: 173.000000
best score: 161.000000
best score: 167.000000
best score: 162.000000
best score: 180.000000
best score: 157.000000
best score: 189.000000
best score: 167.000000
best score: 157.000000
best score: 156.000000
best score: 151.000000
best score: 163.000000
```

```
best score: 172.000000
best score: 181.000000
best score: 175.000000
best score: 134.000000
best score: 178.000000
best score: 169.000000
best score: 182.000000
best score: 177.000000
best score: 153.000000
best score: 182.000000
best score: 192.000000
best score: 188.000000
best score: 178.000000
best score: 174.000000
best score: 195.000000
best score: 188.000000
best score: 177.000000
best score: 167.000000
best score: 169.000000
best score: 183.000000
best score: 179.000000
best score: 177.000000
best score: 193.000000
best score: 183.000000
best score: 186.000000
best score: 186.000000
best score: 174.000000
best score: 196.000000
best score: 189.000000
best score: 186.000000
best score: 199.000000
best score: 202.000000
best score: 218.000000
best score: 190.000000
best score: 205.000000
best score: 178.000000
best score: 181.000000
best score: 193.000000
best score: 196.000000
best score: 203.000000
best score: 218.000000
best score: 214.000000
best score: 196.000000
best score: 196.000000
best score: 199.000000
best score: 181.000000
best score: 216.000000
best score: 221.000000
```

```
best score: 210.000000
best score: 216.000000
best score: 204.000000
best score: 216.000000
best score: 229.000000
best score: 198.000000
best score: 198.000000
best score: 214.000000
best score: 203.000000
best score: 211.000000
best score: 233.000000
best score: 219.000000
best score: 233.000000
best score: 261.000000
best score: 217.000000
best score: 207.000000
best score: 198.000000
best score: 209.000000
best score: 227.000000
best score: 216.000000
best score: 214.000000
best score: 276.000000
best score: 230.000000
best score: 242.000000
best score: 246.000000
best score: 232.000000
best score: 200.000000
best score: 255.000000
best score: 229.000000
best score: 199.000000
best score: 230.000000
best score: 226.000000
best score: 240.000000
best score: 228.000000
best score: 254.000000
best score: 238.000000
best score: 240.000000
best score: 263.000000
best score: 225.000000
best score: 247.000000
best score: 225.000000
best score: 236.000000
best score: 220.000000
best score: 232.000000
best score: 270.000000
best score: 272.000000
best score: 245.000000
best score: 266.000000
```

```
best score: 219.000000
best score: 249.000000
best score: 243.000000
best score: 248.000000
best score: 254.000000
best score: 273.000000
best score: 251.000000
best score: 244.000000
best score: 228.000000
best score: 242.000000
best score: 306.000000
best score: 298.000000
best score: 292.000000
best score: 276.000000
best score: 236.000000
best score: 290.000000
best score: 292.000000
best score: 324.000000
best score: 307.000000
best score: 314.000000
best score: 319.000000
best score: 274.000000
best score: 277.000000
best score: 250.000000
best score: 277.000000
best score: 245.000000
best score: 262.000000
best score: 262.000000
best score: 333.000000
best score: 313.000000
best score: 286.000000
best score: 264.000000
best score: 272.000000
best score: 293.000000
best score: 283.000000
best score: 318.000000
best score: 339.000000
best score: 374.000000
best score: 350.000000
best score: 346.000000
best score: 336.000000
best score: 361.000000
best score: 368.000000
best score: 348.000000
best score: 384.000000
best score: 398.000000
best score: 390.000000
best score: 397.000000
```

```
best score: 466.000000
best score: 428.000000
best score: 647.000000
best score: 2530.000000
best score: 430.000000
best score: 372.000000
best score: 365.000000
best score: 370.000000
best score: 305.000000
best score: 304.000000
best score: 274.000000
best score: 289.000000
best score: 269.000000
best score: 317.000000
best score: 329.000000
best score: 301.000000
best score: 331.000000
best score: 337.000000
best score: 311.000000
best score: 305.000000
best score: 311.000000
best score: 285.000000
best score: 315.000000
best score: 269.000000
best score: 276.000000
best score: 278.000000
best score: 294.000000
best score: 290.000000
best score: 290.000000
best score: 263.000000
best score: 254.000000
best score: 246.000000
best score: 259.000000
best score: 281.000000
best score: 293.000000
best score: 284.000000
best score: 277.000000
best score: 264.000000
best score: 271.000000
best score: 269.000000
best score: 295.000000
best score: 277.000000
best score: 254.000000
best score: 264.000000
best score: 271.000000
best score: 265.000000
best score: 270.000000
best score: 258.000000
```

```
best score: 263.000000
best score: 233.000000
best score: 262.000000
best score: 235.000000
best score: 230.000000
best score: 230.000000
best score: 228.000000
best score: 233.000000
best score: 224.000000
best score: 245.000000
best score: 231.000000
best score: 228.000000
best score: 246.000000
best score: 227.000000
best score: 250.000000
best score: 221.000000
best score: 226.000000
best score: 233.000000
best score: 216.000000
best score: 233.000000
best score: 228.000000
best score: 215.000000
best score: 237.000000
best score: 198.000000
best score: 194.000000
best score: 226.000000
best score: 217.000000
best score: 231.000000
best score: 244.000000
best score: 229.000000
best score: 217.000000
best score: 220.000000
best score: 201.000000
best score: 207.000000
best score: 191.000000
best score: 204.000000
best score: 232.000000
best score: 212.000000
best score: 218.000000
best score: 230.000000
best score: 243.000000
best score: 268.000000
best score: 211.000000
best score: 197.000000
best score: 209.000000
best score: 210.000000
best score: 211.000000
best score: 229.000000
```

```
best score: 208.000000
best score: 204.000000
best score: 178.000000
best score: 209.000000
best score: 195.000000
best score: 206.000000
best score: 220.000000
best score: 183.000000
best score: 163.000000
best score: 167.000000
best score: 172.000000
best score: 182.000000
best score: 181.000000
best score: 186.000000
best score: 189.000000
best score: 201.000000
best score: 189.000000
best score: 174.000000
best score: 167.000000
best score: 198.000000
best score: 203.000000
best score: 184.000000
best score: 168.000000
best score: 152.000000
best score: 173.000000
best score: 168.000000
best score: 190.000000
best score: 161.000000
best score: 186.000000
best score: 177.000000
best score: 163.000000
best score: 165.000000
best score: 172.000000
best score: 179.000000
best score: 165.000000
best score: 181.000000
best score: 179.000000
best score: 184.000000
best score: 180.000000
best score: 169.000000
best score: 167.000000
best score: 146.000000
best score: 153.000000
best score: 175.000000
best score: 165.000000
best score: 176.000000
best score: 162.000000
best score: 172.000000
```

```
best score: 180.000000
best score: 151.000000
best score: 145.000000
best score: 151.000000
best score: 137.000000
best score: 149.000000
best score: 171.000000
best score: 156.000000
best score: 169.000000
best score: 133.000000
best score: 143.000000
best score: 163.000000
best score: 161.000000
best score: 161.000000
best score: 175.000000
best score: 161.000000
best score: 158.000000
best score: 180.000000
best score: 174.000000
best score: 181.000000
best score: 152.000000
best score: 142.000000
best score: 164.000000
best score: 141.000000
best score: 163.000000
best score: 154.000000
best score: 138.000000
best score: 136.000000
best score: 149.000000
best score: 167.000000
best score: 158.000000
best score: 137.000000
best score: 147.000000
best score: 136.000000
best score: 133.000000
best score: 147.000000
best score: 130.000000
best score: 132.000000
best score: 119.000000
best score: 149.000000
best score: 122.000000
best score: 327.000000
best score: 312.000000
best score: 304.000000
best score: 299.000000
best score: 262.000000
best score: 274.000000
best score: 254.000000
```

```
best score: 266.000000
best score: 241.000000
best score: 257.000000
best score: 397.000000
best score: 237.000000
best score: 224.000000
best score: 216.000000
best score: 229.000000
best score: 291.000000
best score: 211.000000
best score: 217.000000
best score: 205.000000
best score: 217.000000
best score: 223.000000
best score: 211.000000
best score: 216.000000
best score: 182.000000
best score: 214.000000
best score: 190.000000
best score: 195.000000
best score: 187.000000
best score: 210.000000
best score: 233.000000
best score: 213.000000
best score: 209.000000
best score: 203.000000
best score: 180.000000
best score: 157.000000
best score: 189.000000
best score: 182.000000
best score: 194.000000
best score: 203.000000
best score: 190.000000
best score: 184.000000
best score: 174.000000
best score: 192.000000
best score: 166.000000
best score: 194.000000
best score: 171.000000
best score: 188.000000
best score: 192.000000
best score: 166.000000
best score: 185.000000
best score: 169.000000
best score: 200.000000
best score: 174.000000
best score: 157.000000
best score: 170.000000
```

```
best score: 172.000000
best score: 195.000000
best score: 153.000000
best score: 156.000000
best score: 162.000000
best score: 155.000000
best score: 166.000000
best score: 131.000000
best score: 164.000000
best score: 142.000000
best score: 175.000000
best score: 154.000000
best score: 166.000000
best score: 159.000000
best score: 161.000000
best score: 169.000000
best score: 150.000000
best score: 154.000000
best score: 166.000000
best score: 163.000000
best score: 147.000000
best score: 168.000000
best score: 143.000000
best score: 157.000000
best score: 161.000000
best score: 167.000000
best score: 159.000000
best score: 140.000000
best score: 165.000000
best score: 176.000000
best score: 159.000000
best score: 150.000000
best score: 152.000000
best score: 135.000000
best score: 139.000000
best score: 152.000000
best score: 150.000000
best score: 160.000000
best score: 157.000000
best score: 142.000000
best score: 144.000000
best score: 126.000000
best score: 140.000000
best score: 127.000000
best score: 134.000000
best score: 131.000000
best score: 122.000000
best score: 122.000000
```

```
best score: 148.000000
best score: 127.000000
best score: 128.000000
best score: 135.000000
best score: 131.000000
best score: 128.000000
best score: 135.000000
best score: 150.000000
best score: 119.000000
best score: 144.000000
best score: 133.000000
best score: 130.000000
best score: 126.000000
best score: 118.000000
best score: 119.000000
best score: 118.000000
best score: 113.000000
best score: 118.000000
best score: 107.000000
best score: 132.000000
best score: 110.000000
best score: 101.000000
best score: 108.000000
best score: 120.000000
best score: 116.000000
best score: 110.000000
best score: 113.000000
best score: 102.000000
best score: 125.000000
best score: 107.000000
best score: 125.000000
best score: 120.000000
best score: 101.000000
best score: 96.000000
best score: 97.000000
best score: 104.000000
best score: 102.000000
best score: 101.000000
best score: 90.000000
best score: 89.000000
best score: 97.000000
best score: 108.000000
best score: 108.000000
best score: 92.000000
best score: 106.000000
best score: 91.000000
best score: 102.000000
best score: 108.000000
```

```
best score: 105.000000
best score: 104.000000
best score: 94.000000
best score: 132.000000
best score: 105.000000
best score: 127.000000
best score: 119.000000
best score: 93.000000
best score: 116.000000
best score: 101.000000
best score: 115.000000
best score: 112.000000
best score: 115.000000
best score: 111.000000
best score: 101.000000
best score: 112.000000
best score: 115.000000
best score: 111.000000
best score: 111.000000
best score: 83.000000
best score: 103.000000
best score: 128.000000
best score: 100.000000
best score: 110.000000
best score: 90.000000
best score: 102.000000
best score: 105.000000
best score: 101.000000
best score: 109.000000
best score: 315.000000
best score: 263.000000
best score: 274.000000
best score: 272.000000
best score: 229.000000
best score: 234.000000
best score: 236.000000
best score: 255.000000
best score: 209.000000
best score: 201.000000
best score: 208.000000
best score: 210.000000
best score: 212.000000
best score: 214.000000
best score: 219.000000
best score: 241.000000
best score: 224.000000
best score: 220.000000
best score: 194.000000
```

```
best score: 202.000000
best score: 175.000000
best score: 193.000000
best score: 181.000000
best score: 209.000000
best score: 200.000000
best score: 190.000000
best score: 174.000000
best score: 194.000000
best score: 182.000000
best score: 193.000000
best score: 181.000000
best score: 174.000000
best score: 174.000000
best score: 170.000000
best score: 195.000000
best score: 180.000000
best score: 181.000000
best score: 169.000000
best score: 164.000000
best score: 179.000000
best score: 175.000000
best score: 164.000000
best score: 167.000000
best score: 166.000000
best score: 150.000000
best score: 157.000000
best score: 184.000000
best score: 166.000000
best score: 159.000000
best score: 150.000000
best score: 163.000000
best score: 155.000000
best score: 159.000000
best score: 175.000000
best score: 167.000000
best score: 175.000000
best score: 186.000000
best score: 186.000000
best score: 173.000000
best score: 172.000000
best score: 157.000000
best score: 170.000000
best score: 176.000000
best score: 168.000000
best score: 173.000000
best score: 193.000000
best score: 163.000000
```

```
best score: 177.000000
best score: 169.000000
best score: 171.000000
best score: 177.000000
best score: 166.000000
best score: 152.000000
best score: 157.000000
best score: 150.000000
best score: 152.000000
best score: 149.000000
best score: 163.000000
best score: 146.000000
best score: 132.000000
best score: 144.000000
best score: 155.000000
best score: 146.000000
best score: 155.000000
best score: 164.000000
best score: 153.000000
best score: 155.000000
best score: 129.000000
best score: 142.000000
```

[17]:
```
utils.imageFolder2mpeg('aligned_frames', fps=30)
```

1.1.4 Part 4: Create background panorama

In this part you will remove moving objects from the video and create a background panorama that should incorporate pixels from all the frames.

In the video you produced in **part 3** each pixel appears in several frames. You need to estimate which of the many colors correspond to the background. We take advantage of the fact that the background color is fixed while the foreground color changes frequently (because foreground moves).

For each pixel in the sequence of **part 3**, determine all valid colors (colors that come from all frames that overlap that pixel). You can experiment with different methods for determining the background color of each pixel, as discussed in class. Perform the same procedure for all pixels and generate output. The output should be a completed panorama showing only pixels of background or non-moving objects.

[18]:
```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

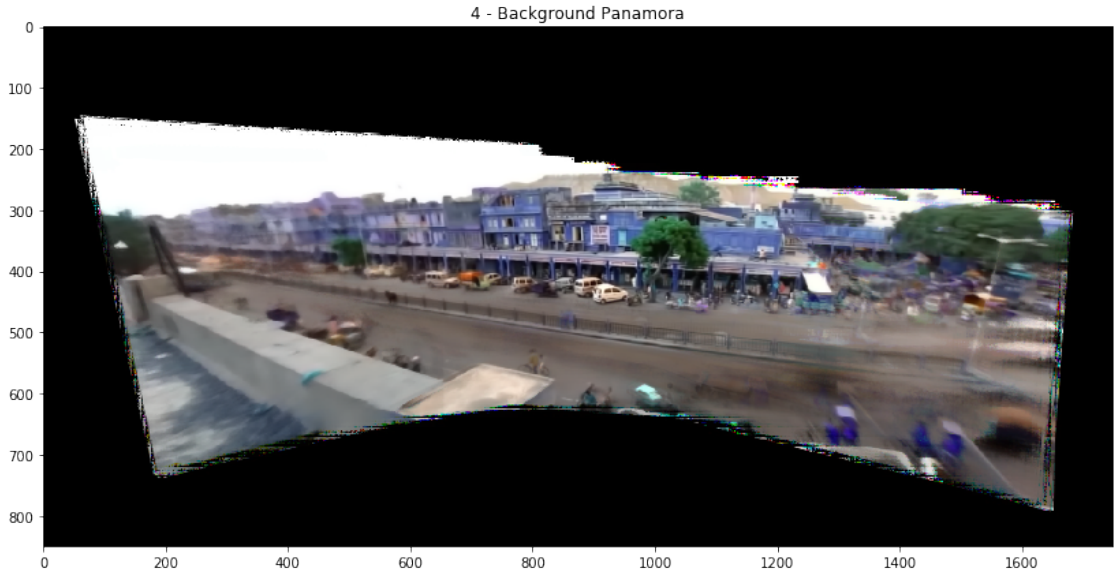
[19]:
```
#Variables Copied from Above. "Reset" frames variable to use panoramaFrames
dir_frames = 'aligned_frames'
filenames = []
```

```
filesinfo = os.scandir(dir_frames)

filenames = [f.path for f in filesinfo if f.name.endswith(".jpg")]
filenames.sort(key=lambda f: int(''.join(filter(str.isdigit, f))))

frameCount = len(filenames)
# frameHeight, frameWidth, frameChannels = cv2.imread(filenames[0]).shape
panoramaFrames = np.zeros((frameCount, projectedHeight, projectedWidth,␣
 ↪frameChannels),dtype=np.float32)
print("Scanning")
for idx, file_i in enumerate(filenames):
    panoramaFrames[idx] = cv2.imread(file_i) / 255.0

backgroundPanorama = np.zeros((projectedHeight, projectedWidth, 3))
print("Starting")
for a in range(projectedHeight):
    for b in range(projectedWidth):
        for c in range(3):
            pixel = np.nonzero(panoramaFrames[:, a, b, c])
            # Calculate Median and populate for output
            if len(pixel) > 0:
                backgroundPanorama[a][b][c] = np.median(panoramaFrames[pixel,␣
 ↪a, b, c])

print("Done")
```

```
/usr/local/lib/python3.7/site-packages/numpy/core/fromnumeric.py:3257:
RuntimeWarning: Mean of empty slice.
  out=out, **kwargs)
/usr/local/lib/python3.7/site-packages/numpy/core/_methods.py:161:
RuntimeWarning: invalid value encountered in true_divide
  ret = ret.dtype.type(ret / rcount)
```

[20]:
```
fig, axes = plt.subplots(1, 1)
fig.set_size_inches(15, 7)

axes.imshow(backgroundPanorama)
axes.set_title("4 - Background Panamora")
cv2.imwrite("part4.jpg", backgroundPanorama*255)
```

[20]: True



1.1.5 Part 5: Create background movie

Map the background panorama to the movie coordinates. For each frame of the movie, say frame 1, you need to estimate a projection from the panorama to frame 1. Note, you should be able to re-use the homographies that you estimated in **Part 3**. Perform this for all frames and generate a movie that looks like the input movie but shows only background pixels. All moving objects that belong to the foreground must be removed.

[21]:
```
import os
import cv2
import numpy as np
```

[22]:
```
# https://piazza.com/class/k5cumohrew35en?cid=876
os.mkdir("part5")

# background_images = np.zeros((900,projectedHeight, projectedWidth,3))
background_images = np.zeros((900, 360, 480,3))

for idx in range(0, 900):
    invertedH = np.linalg.inv(H_t.dot(pastHomographies[idx][450]))
#     projected_image = cv2.warpPerspective(backgroundPanorama, invertedH,␣
↪(projectedWidth, projectedHeight))
    projected_image = cv2.warpPerspective(backgroundPanorama, invertedH, (480,␣
↪360))
    background_images[idx,:,:,:] = projected_image
    cv2.imwrite('part5/a{:04d}.jpg'.format(idx), projected_image*255)
```

[23]:
```
utils.imageFolder2mpeg('part5', fps=30)
```

1.1.6 Part 6: Create foreground movie

In the background video, moving objects are removed. In each frame, those pixels that are different enough than the background color are considered foreground. For each frame determine foreground pixels and generate a movie that only includes foreground pixels.

[24]:
```
import os
import cv2
import numpy as np
```

[25]:
```
os.mkdir("part6")
for idx in range(0, 900):
    difference = frames[idx] - background_images[idx]
    cv2.imwrite('part6/a{:04d}.jpg'.format(idx), difference*255)
```

[26]:
```
utils.imageFolder2mpeg('part6', fps=30)
```

1.2 Bells and whistles

1.2.1 Part 7*: Generate a wide video [10 pts]

In Part 5 you created a background movie by projecting back the panorama background to each frame plane. If you map a wider area you will get a wider background movie. You can use this background movie to extend the borders of your video and make it wider. The extended video must be at least 50% wider. You can keep the same height.

[32]:
```
# https://piazza.com/class/k5cumohrew35en?cid=876
os.mkdir("wide")
background_images2 = np.zeros((900,projectedHeight, projectedWidth,3))
for idx in range(0, 900):
    invertedH = np.linalg.inv(H_t.dot(pastHomographies[idx][450]))
    projected_image2 = cv2.warpPerspective(backgroundPanorama, invertedH,␣
↪(1750, 850))
    background_images2[idx,:,:,:] = projected_image2
    cv2.imwrite('wide/a{:04d}.jpg'.format(idx), projected_image2*255)
```

[33]:
```
utils.imageFolder2mpeg('wide', fps=30)
```

1.2.2 Part 8: Process two more videos [up to 40 points]

You can apply the seven parts of the main project on two other videos. You get 20 points for processing one additional video and 40 points for processing two. If you do two additional videos, one of them must be your own. You get full points if you complete the first six parts of the main project.

I will be processing one additional video

Part 8.1: Stitch two key frames

[34]:
```
import cv2
import numpy as np
from numpy.linalg import svd, inv
import utils
%matplotlib inline
from matplotlib import pyplot as plt
```

[35]:
```
# Outputs 2711
# MANUALLY DELETE, keep 2000-2722, rename the images used for reference to␣
 ↪reflence.
# f2070, f2210, f2350, f2490, f2630

# os.mkdir("convertedImages2")
# utils.video2imageFolder('street1.mp4', "convertedImages2")
```

[36]:
```
# images location
im1 = './images/input/frames2/f2210.jpg'
im2 = './images/input/frames2/f2350.jpg'

# Load an color image in grayscale
im1 = cv2.imread(im1)
im2 = cv2.imread(im2)

im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2RGB)
```

[37]:
```
H = auto_homography(im2,im1, computeHomography)
```

```
best score: 383.000000
```

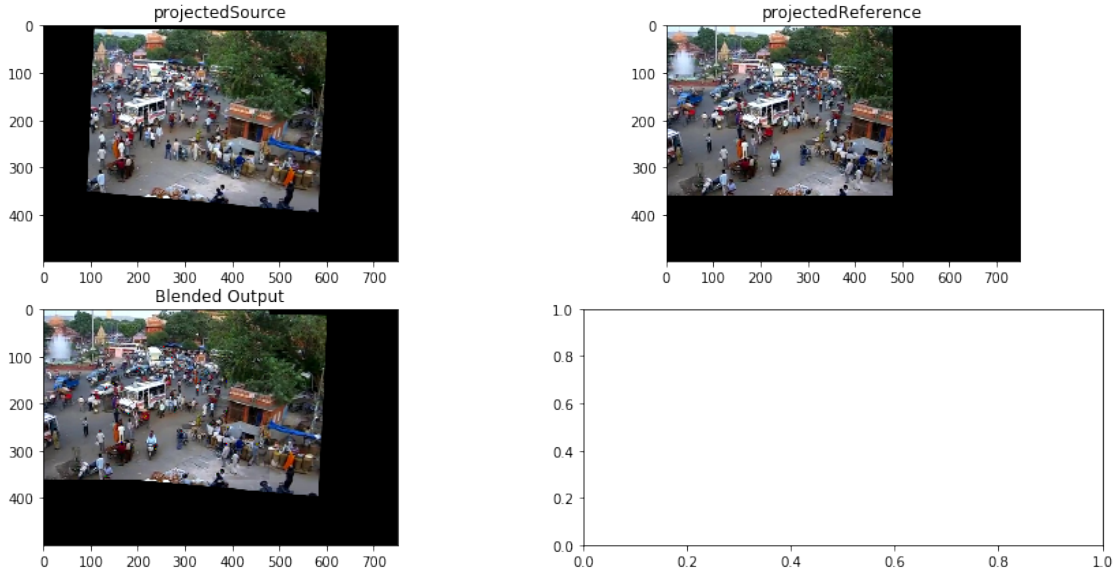
[38]:
```
# Checking
plotA = np.array([270, 450, 450, 270, 270])
plotB = np.array([50, 50, 270, 270, 50])
ones = np.ones(5)
plotC = np.array([plotA, plotB, ones])
plotD = np.dot(H, plotC)
plotD = plotD / plotD[-1]

fig, axes = plt.subplots(1, 2)
fig.set_size_inches(15, 7)

axes[0].imshow(im1)
axes[0].plot(plotA, plotB, 'r') #use red instead of blue
axes[1].imshow(im2)
axes[1].plot(plotD[0], plotD[1], 'r')
```

[38]:
```
[<matplotlib.lines.Line2D at 0x11ac97190>]
```

[39]:
```
# # Example usage of cv2.warpPerspective
# projectedWidth = 1600
# projectedHeight = 500

projectedWidth = 750
projectedHeight = 500

sourceHomography = H
refHomography = np.identity(3)

projectedSource = cv2.warpPerspective(im2, sourceHomography, (projectedWidth,␣
 ↪projectedHeight))
projectedReference = cv2.warpPerspective(im1, refHomography, (projectedWidth,␣
 ↪projectedHeight))

fig, axes = plt.subplots(2, 2)
fig.set_size_inches(15, 7)

axes[0, 0].imshow(projectedSource)
axes[0, 0].set_title("projectedSource")
axes[0, 1].imshow(projectedReference)
axes[0, 1].set_title("projectedReference")

# Example usage of utils.blendImages
blendedOutput = utils.blendImages(projectedSource, projectedReference)
# blendedOutput = cv2.cvtColor(blendedOutput, cv2.COLOR_BGR2RGB)
# plt.figure()
# plt.imshow(blendedOutput)
axes[1, 0].imshow(blendedOutput)
axes[1, 0].set_title("Blended Output")

# cv2.imwrite("2-part1_projectedSource.jpg", projectedSource)
```

```
# cv2.imwrite("2-part1_projectedReference.jpg", projectedReference)
# cv2.imwrite("2-part1_blended.jpg", blendedOutput)
```

[39]:
```
Text(0.5, 1.0, 'Blended Output')
```



Part 8.2: Map the video to the reference plane

[40]:
```
import cv2
import numpy as np
```

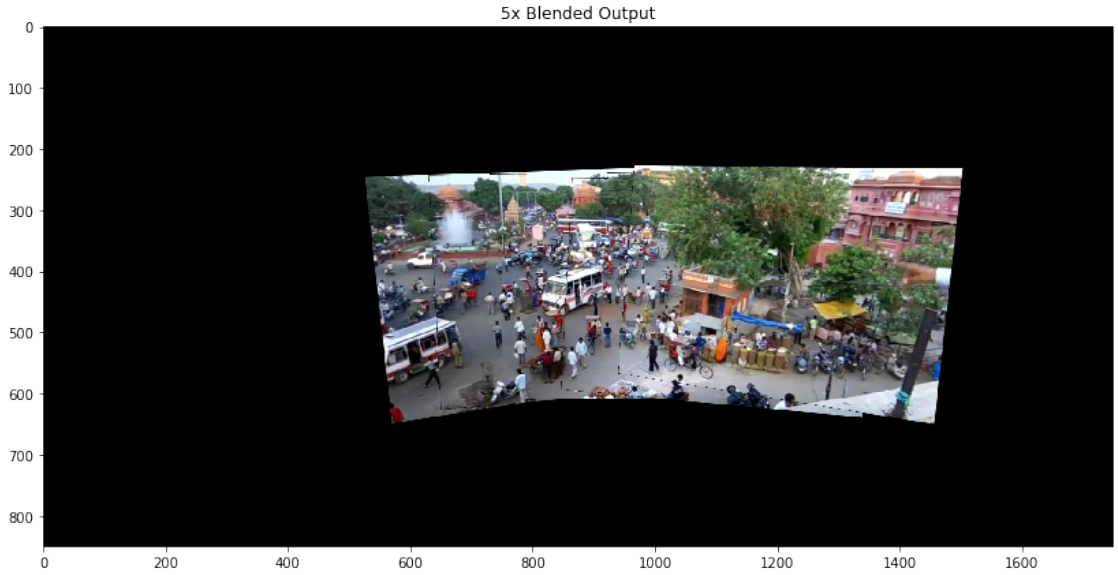
[41]:
```
master_frames = [70, 210, 350, 490, 630]
reference_frame = 350
reference_idx = master_frames.index(reference_frame)

im1 = cv2.imread('./images/input/frames2/f2070.jpg')
im2 = cv2.imread('./images/input/frames2/f2210.jpg')
im3 = cv2.imread('./images/input/frames2/f2350.jpg')
im4 = cv2.imread('./images/input/frames2/f2490.jpg')
im5 = cv2.imread('./images/input/frames2/f2630.jpg')

im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2RGB)
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2RGB)
im3 = cv2.cvtColor(im3, cv2.COLOR_BGR2RGB)
im4 = cv2.cvtColor(im4, cv2.COLOR_BGR2RGB)
im5 = cv2.cvtColor(im5, cv2.COLOR_BGR2RGB)
```

[42]:
```
fig, axes = plt.subplots(1, 1)
fig.set_size_inches(15, 7)
```

```
projectedWidth = 1750
projectedHeight = 850

H_12 = auto_homography(im1,im2, computeHomography)
H_23 = auto_homography(im2,im3, computeHomography)
H_34 = auto_homography(im4,im3, computeHomography)
H_45 = auto_homography(im5,im4, computeHomography)

H_t = np.array([[1, 0, 750], [0, 1, 250], [0, 0, 1]])
refHomography = np.identity(3)

H_13 = np.dot(H_23,H_12)
H_35 = np.dot(H_45,H_34)

warped_1 = cv2.warpPerspective(im1, H_t.dot(H_13), (projectedWidth,␣
 ↪projectedHeight))
warped_2 = cv2.warpPerspective(im2, H_t.dot(H_23), (projectedWidth,␣
 ↪projectedHeight))
warped_3 = cv2.warpPerspective(im3, H_t.dot(refHomography), (projectedWidth,␣
 ↪projectedHeight))
warped_4 = cv2.warpPerspective(im4, H_t.dot(H_34), (projectedWidth,␣
 ↪projectedHeight))
warped_5 = cv2.warpPerspective(im5, H_t.dot(H_35), (projectedWidth,␣
 ↪projectedHeight))

blend_1 = utils.blendImages(warped_1, warped_2)
blend_2 = utils.blendImages(blend_1, warped_3)
blend_3 = utils.blendImages(blend_2, warped_4)
blend_4 = utils.blendImages(blend_3, warped_5)
axes.imshow(blend_4)
axes.set_title("5x Blended Output")

# cv2.imwrite("2-part2.jpg", blend_4)
```

```
best score: 348.000000
best score: 375.000000
best score: 590.000000
best score: 752.000000
```

[42]: Text(0.5, 1.0, '5x Blended Output')



Part 8.3: Map the video to the reference plane

[43]:
```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from math import floor

import utils
```

[44]:
```
dir_frames = 'images/input/frames2'
filenames = []
filesinfo = os.scandir(dir_frames)
```

[45]:
```
filenames = [f.path for f in filesinfo if f.name.endswith(".jpg")]
filenames.sort(key=lambda f: int(''.join(filter(str.isdigit, f))))
```

[46]:
```
frameCount = len(filenames)
frameHeight, frameWidth, frameChannels = cv2.imread(filenames[0]).shape
frames = np.zeros((frameCount, frameHeight, frameWidth, frameChannels),dtype=np.
↪float32)
```

[47]:
```
for idx, file_i in enumerate(filenames):
    frames[idx] = cv2.cvtColor(cv2.imread(file_i), cv2.COLOR_BGR2RGB) / 255.0
print("Number of Pictures to Generate2", len(frames)) #User line check
```

```
Number of Pictures to Generate2 723
```

[48]:
```
os.mkdir("aligned_frames2")

pastHomographies = np.zeros((len(filenames),len(filenames), 3, 3),dtype=np.
 ↪float32)
for idx, file_i in enumerate(filenames):
    img = cv2.imread(file_i)
    if idx < 70:
        H = auto_homography(img,im1, computeHomography)
        H = np.dot(H,H_13)
        pastHomographies[idx] = H
        warped_1 = cv2.warpPerspective(img, np.dot(H_t,H), (projectedWidth,␣
 ↪projectedHeight))
        cv2.imwrite("aligned_frames2/a{:04d}.jpg".format(idx), warped_1)
    elif (idx < 210) and (idx >= 70):
        H = auto_homography(img,im2, computeHomography)
        H = np.dot(H,H_23)
        pastHomographies[idx] = H
        warped_2 = cv2.warpPerspective(img, np.dot(H_t,H), (projectedWidth,␣
 ↪projectedHeight))
        cv2.imwrite("aligned_frames2/a{:04d}.jpg".format(idx), warped_2)
    elif (idx >= 210) and (idx <= 490):
        H = auto_homography(img,im3, computeHomography)
        pastHomographies[idx] = H
        warped_3 = cv2.warpPerspective(img, np.dot(H_t,H), (projectedWidth,␣
 ↪projectedHeight))
        cv2.imwrite("aligned_frames2/a{:04d}.jpg".format(idx), warped_3)
    elif (idx > 490) and (idx <= 630):
        H = auto_homography(img,im4, computeHomography)
        H = np.dot(H,H_34)
        pastHomographies[idx] = H
        warped_4 = cv2.warpPerspective(img, np.dot(H_t,H), (projectedWidth,␣
 ↪projectedHeight))
        cv2.imwrite("aligned_frames2/a{:04d}.jpg".format(idx), warped_4)
    elif idx > 630:
        H = auto_homography(img,im5, computeHomography)
        H = np.dot(H,H_45)
        pastHomographies[idx] = H
        warped_5 = cv2.warpPerspective(img, np.dot(H_t,H), (projectedWidth,␣
 ↪projectedHeight))
        cv2.imwrite("aligned_frames2/a{:04d}.jpg".format(idx), warped_5)
```

```
best score: 439.000000
best score: 416.000000
best score: 450.000000
best score: 415.000000
best score: 415.000000
best score: 446.000000
```

```
best score: 457.000000
best score: 475.000000
best score: 428.000000
best score: 447.000000
best score: 455.000000
best score: 462.000000
best score: 468.000000
best score: 472.000000
best score: 471.000000
best score: 452.000000
best score: 474.000000
best score: 492.000000
best score: 487.000000
best score: 489.000000
best score: 489.000000
best score: 498.000000
best score: 508.000000
best score: 496.000000
best score: 474.000000
best score: 477.000000
best score: 504.000000
best score: 491.000000
best score: 510.000000
best score: 516.000000
best score: 509.000000
best score: 531.000000
best score: 534.000000
best score: 513.000000
best score: 509.000000
best score: 502.000000
best score: 499.000000
best score: 501.000000
best score: 520.000000
best score: 507.000000
best score: 526.000000
best score: 497.000000
best score: 555.000000
best score: 580.000000
best score: 564.000000
best score: 604.000000
best score: 642.000000
best score: 637.000000
best score: 661.000000
best score: 662.000000
best score: 671.000000
best score: 723.000000
best score: 717.000000
best score: 712.000000
```

```
best score: 745.000000
best score: 801.000000
best score: 780.000000
best score: 868.000000
best score: 880.000000
best score: 907.000000
best score: 960.000000
best score: 1002.000000
best score: 1065.000000
best score: 1081.000000
best score: 1198.000000
best score: 1324.000000
best score: 1447.000000
best score: 1573.000000
best score: 1846.000000
best score: 2167.000000
best score: 352.000000
best score: 334.000000
best score: 360.000000
best score: 367.000000
best score: 358.000000
best score: 359.000000
best score: 382.000000
best score: 364.000000
best score: 384.000000
best score: 358.000000
best score: 368.000000
best score: 368.000000
best score: 359.000000
best score: 381.000000
best score: 340.000000
best score: 357.000000
best score: 364.000000
best score: 354.000000
best score: 337.000000
best score: 371.000000
best score: 349.000000
best score: 357.000000
best score: 387.000000
best score: 357.000000
best score: 356.000000
best score: 350.000000
best score: 359.000000
best score: 348.000000
best score: 357.000000
best score: 357.000000
best score: 369.000000
best score: 310.000000
```

```
best score: 316.000000
best score: 336.000000
best score: 345.000000
best score: 338.000000
best score: 309.000000
best score: 327.000000
best score: 302.000000
best score: 302.000000
best score: 295.000000
best score: 307.000000
best score: 323.000000
best score: 323.000000
best score: 295.000000
best score: 300.000000
best score: 313.000000
best score: 323.000000
best score: 359.000000
best score: 344.000000
best score: 356.000000
best score: 334.000000
best score: 349.000000
best score: 347.000000
best score: 343.000000
best score: 380.000000
best score: 358.000000
best score: 373.000000
best score: 350.000000
best score: 380.000000
best score: 388.000000
best score: 362.000000
best score: 396.000000
best score: 397.000000
best score: 420.000000
best score: 427.000000
best score: 440.000000
best score: 432.000000
best score: 444.000000
best score: 462.000000
best score: 455.000000
best score: 462.000000
best score: 461.000000
best score: 493.000000
best score: 483.000000
best score: 477.000000
best score: 484.000000
best score: 512.000000
best score: 502.000000
best score: 492.000000
```

```
best score: 512.000000
best score: 528.000000
best score: 517.000000
best score: 564.000000
best score: 545.000000
best score: 521.000000
best score: 562.000000
best score: 551.000000
best score: 549.000000
best score: 562.000000
best score: 557.000000
best score: 538.000000
best score: 551.000000
best score: 533.000000
best score: 571.000000
best score: 577.000000
best score: 577.000000
best score: 562.000000
best score: 594.000000
best score: 609.000000
best score: 604.000000
best score: 570.000000
best score: 611.000000
best score: 632.000000
best score: 615.000000
best score: 645.000000
best score: 679.000000
best score: 669.000000
best score: 666.000000
best score: 683.000000
best score: 685.000000
best score: 689.000000
best score: 736.000000
best score: 742.000000
best score: 775.000000
best score: 781.000000
best score: 782.000000
best score: 770.000000
best score: 822.000000
best score: 853.000000
best score: 856.000000
best score: 855.000000
best score: 873.000000
best score: 920.000000
best score: 931.000000
best score: 973.000000
best score: 965.000000
best score: 1022.000000
```

```
best score: 1033.000000
best score: 1044.000000
best score: 1111.000000
best score: 1116.000000
best score: 1160.000000
best score: 1200.000000
best score: 1265.000000
best score: 1339.000000
best score: 1435.000000
best score: 1593.000000
best score: 1810.000000
best score: 2128.000000
best score: 370.000000
best score: 345.000000
best score: 370.000000
best score: 388.000000
best score: 402.000000
best score: 400.000000
best score: 403.000000
best score: 402.000000
best score: 417.000000
best score: 437.000000
best score: 397.000000
best score: 404.000000
best score: 408.000000
best score: 425.000000
best score: 419.000000
best score: 421.000000
best score: 408.000000
best score: 387.000000
best score: 401.000000
best score: 416.000000
best score: 438.000000
best score: 441.000000
best score: 457.000000
best score: 438.000000
best score: 435.000000
best score: 440.000000
best score: 456.000000
best score: 472.000000
best score: 462.000000
best score: 451.000000
best score: 464.000000
best score: 466.000000
best score: 457.000000
best score: 473.000000
best score: 470.000000
best score: 450.000000
```

```
best score: 486.000000
best score: 489.000000
best score: 503.000000
best score: 459.000000
best score: 488.000000
best score: 449.000000
best score: 453.000000
best score: 492.000000
best score: 477.000000
best score: 485.000000
best score: 528.000000
best score: 518.000000
best score: 504.000000
best score: 539.000000
best score: 554.000000
best score: 589.000000
best score: 583.000000
best score: 578.000000
best score: 557.000000
best score: 574.000000
best score: 608.000000
best score: 574.000000
best score: 562.000000
best score: 560.000000
best score: 600.000000
best score: 611.000000
best score: 596.000000
best score: 603.000000
best score: 662.000000
best score: 626.000000
best score: 649.000000
best score: 657.000000
best score: 667.000000
best score: 669.000000
best score: 668.000000
best score: 671.000000
best score: 669.000000
best score: 661.000000
best score: 691.000000
best score: 713.000000
best score: 728.000000
best score: 744.000000
best score: 710.000000
best score: 711.000000
best score: 703.000000
best score: 716.000000
best score: 733.000000
best score: 714.000000
```

```
best score: 708.000000
best score: 721.000000
best score: 751.000000
best score: 732.000000
best score: 788.000000
best score: 755.000000
best score: 757.000000
best score: 781.000000
best score: 763.000000
best score: 806.000000
best score: 782.000000
best score: 791.000000
best score: 791.000000
best score: 804.000000
best score: 799.000000
best score: 824.000000
best score: 825.000000
best score: 731.000000
best score: 763.000000
best score: 780.000000
best score: 799.000000
best score: 843.000000
best score: 867.000000
best score: 875.000000
best score: 867.000000
best score: 875.000000
best score: 932.000000
best score: 926.000000
best score: 910.000000
best score: 953.000000
best score: 964.000000
best score: 967.000000
best score: 1026.000000
best score: 1041.000000
best score: 1062.000000
best score: 1049.000000
best score: 1050.000000
best score: 1073.000000
best score: 1116.000000
best score: 1138.000000
best score: 1156.000000
best score: 1191.000000
best score: 1235.000000
best score: 1308.000000
best score: 1332.000000
best score: 1352.000000
best score: 1380.000000
best score: 1405.000000
```

```
best score: 1525.000000
best score: 1601.000000
best score: 1659.000000
best score: 1685.000000
best score: 1775.000000
best score: 2031.000000
best score: 2150.000000
best score: 2359.000000
best score: 3028.000000
best score: 2501.000000
best score: 2162.000000
best score: 1947.000000
best score: 1828.000000
best score: 1731.000000
best score: 1670.000000
best score: 1577.000000
best score: 1530.000000
best score: 1433.000000
best score: 1416.000000
best score: 1402.000000
best score: 1367.000000
best score: 1318.000000
best score: 1282.000000
best score: 1253.000000
best score: 1224.000000
best score: 1180.000000
best score: 1144.000000
best score: 1088.000000
best score: 1102.000000
best score: 788.000000
best score: 782.000000
best score: 785.000000
best score: 836.000000
best score: 787.000000
best score: 785.000000
best score: 812.000000
best score: 814.000000
best score: 794.000000
best score: 790.000000
best score: 810.000000
best score: 801.000000
best score: 830.000000
best score: 842.000000
best score: 824.000000
best score: 826.000000
best score: 838.000000
best score: 865.000000
best score: 828.000000
```

```
best score: 865.000000
best score: 868.000000
best score: 851.000000
best score: 854.000000
best score: 846.000000
best score: 799.000000
best score: 834.000000
best score: 825.000000
best score: 831.000000
best score: 854.000000
best score: 840.000000
best score: 840.000000
best score: 850.000000
best score: 810.000000
best score: 809.000000
best score: 809.000000
best score: 796.000000
best score: 775.000000
best score: 789.000000
best score: 799.000000
best score: 779.000000
best score: 777.000000
best score: 794.000000
best score: 777.000000
best score: 779.000000
best score: 764.000000
best score: 762.000000
best score: 780.000000
best score: 761.000000
best score: 740.000000
best score: 725.000000
best score: 733.000000
best score: 739.000000
best score: 748.000000
best score: 780.000000
best score: 744.000000
best score: 778.000000
best score: 792.000000
best score: 785.000000
best score: 766.000000
best score: 738.000000
best score: 641.000000
best score: 651.000000
best score: 665.000000
best score: 652.000000
best score: 661.000000
best score: 678.000000
best score: 663.000000
```

```
best score: 676.000000
best score: 680.000000
best score: 705.000000
best score: 708.000000
best score: 732.000000
best score: 747.000000
best score: 739.000000
best score: 715.000000
best score: 690.000000
best score: 714.000000
best score: 725.000000
best score: 712.000000
best score: 681.000000
best score: 703.000000
best score: 706.000000
best score: 653.000000
best score: 670.000000
best score: 645.000000
best score: 645.000000
best score: 667.000000
best score: 635.000000
best score: 673.000000
best score: 669.000000
best score: 651.000000
best score: 659.000000
best score: 660.000000
best score: 626.000000
best score: 629.000000
best score: 617.000000
best score: 622.000000
best score: 642.000000
best score: 637.000000
best score: 599.000000
best score: 582.000000
best score: 600.000000
best score: 579.000000
best score: 581.000000
best score: 594.000000
best score: 606.000000
best score: 547.000000
best score: 603.000000
best score: 554.000000
best score: 586.000000
best score: 581.000000
best score: 573.000000
best score: 560.000000
best score: 565.000000
best score: 538.000000
```

```
best score: 535.000000
best score: 519.000000
best score: 523.000000
best score: 546.000000
best score: 578.000000
best score: 1471.000000
best score: 1439.000000
best score: 1432.000000
best score: 1355.000000
best score: 1339.000000
best score: 1321.000000
best score: 1335.000000
best score: 1336.000000
best score: 1340.000000
best score: 1306.000000
best score: 1293.000000
best score: 1275.000000
best score: 1262.000000
best score: 1260.000000
best score: 1233.000000
best score: 1236.000000
best score: 1242.000000
best score: 1224.000000
best score: 1202.000000
best score: 1161.000000
best score: 1124.000000
best score: 1125.000000
best score: 1132.000000
best score: 1108.000000
best score: 1115.000000
best score: 1109.000000
best score: 1090.000000
best score: 1077.000000
best score: 1074.000000
best score: 1070.000000
best score: 1031.000000
best score: 1070.000000
best score: 1057.000000
best score: 1048.000000
best score: 1103.000000
best score: 1061.000000
best score: 1031.000000
best score: 1043.000000
best score: 1041.000000
best score: 1047.000000
best score: 1056.000000
best score: 1064.000000
best score: 1025.000000
```

```
best score: 1068.000000
best score: 1002.000000
best score: 1006.000000
best score: 971.000000
best score: 942.000000
best score: 963.000000
best score: 952.000000
best score: 966.000000
best score: 978.000000
best score: 968.000000
best score: 908.000000
best score: 937.000000
best score: 970.000000
best score: 937.000000
best score: 941.000000
best score: 914.000000
best score: 943.000000
best score: 715.000000
best score: 774.000000
best score: 782.000000
best score: 784.000000
best score: 817.000000
best score: 814.000000
best score: 867.000000
best score: 854.000000
best score: 851.000000
best score: 829.000000
best score: 875.000000
best score: 868.000000
best score: 843.000000
best score: 857.000000
best score: 832.000000
best score: 830.000000
best score: 841.000000
best score: 852.000000
best score: 865.000000
best score: 853.000000
best score: 820.000000
best score: 819.000000
best score: 843.000000
best score: 864.000000
best score: 831.000000
best score: 848.000000
best score: 803.000000
best score: 849.000000
best score: 820.000000
best score: 830.000000
best score: 792.000000
```

```
best score: 766.000000
best score: 768.000000
best score: 793.000000
best score: 800.000000
best score: 831.000000
best score: 802.000000
best score: 797.000000
best score: 798.000000
best score: 817.000000
best score: 804.000000
best score: 829.000000
best score: 831.000000
best score: 848.000000
best score: 774.000000
best score: 817.000000
best score: 820.000000
best score: 763.000000
best score: 782.000000
best score: 821.000000
best score: 780.000000
best score: 849.000000
best score: 792.000000
best score: 814.000000
best score: 775.000000
best score: 802.000000
best score: 805.000000
best score: 794.000000
best score: 791.000000
best score: 797.000000
best score: 646.000000
best score: 652.000000
best score: 698.000000
best score: 738.000000
best score: 692.000000
best score: 707.000000
best score: 680.000000
best score: 692.000000
best score: 710.000000
best score: 692.000000
best score: 711.000000
best score: 714.000000
best score: 726.000000
best score: 683.000000
best score: 701.000000
best score: 695.000000
best score: 698.000000
best score: 677.000000
best score: 688.000000
```

```
best score: 686.000000
best score: 2661.000000
best score: 2445.000000
best score: 2328.000000
best score: 2278.000000
best score: 2210.000000
best score: 2140.000000
best score: 2062.000000
best score: 1973.000000
best score: 1887.000000
best score: 1876.000000
best score: 1783.000000
best score: 1745.000000
best score: 1680.000000
best score: 1660.000000
best score: 1661.000000
best score: 1598.000000
best score: 1584.000000
best score: 1564.000000
best score: 1561.000000
best score: 1566.000000
best score: 1571.000000
best score: 1554.000000
best score: 1569.000000
best score: 1514.000000
best score: 1518.000000
best score: 1479.000000
best score: 1446.000000
best score: 1448.000000
best score: 1458.000000
best score: 1477.000000
best score: 1499.000000
best score: 1471.000000
best score: 1439.000000
best score: 1490.000000
best score: 1485.000000
best score: 1496.000000
best score: 1491.000000
best score: 1474.000000
best score: 1439.000000
best score: 1445.000000
best score: 1151.000000
best score: 1145.000000
best score: 1207.000000
best score: 1201.000000
best score: 1255.000000
best score: 1250.000000
best score: 1242.000000
```

```
best score: 1293.000000
best score: 1293.000000
best score: 1271.000000
best score: 1281.000000
best score: 1299.000000
best score: 1296.000000
best score: 1298.000000
best score: 1284.000000
best score: 1339.000000
best score: 1327.000000
best score: 1314.000000
best score: 1313.000000
best score: 1337.000000
best score: 1308.000000
best score: 1348.000000
best score: 1339.000000
best score: 1345.000000
best score: 1364.000000
best score: 1365.000000
best score: 1352.000000
best score: 1345.000000
best score: 1351.000000
best score: 1320.000000
best score: 1328.000000
best score: 1336.000000
best score: 1307.000000
best score: 1312.000000
best score: 1298.000000
best score: 1294.000000
best score: 1307.000000
best score: 1278.000000
best score: 1288.000000
best score: 1340.000000
best score: 1289.000000
best score: 1295.000000
best score: 1324.000000
best score: 1266.000000
best score: 1276.000000
best score: 1294.000000
best score: 1266.000000
best score: 1259.000000
best score: 1282.000000
best score: 1262.000000
best score: 1247.000000
best score: 1224.000000
```

[49]:
```
utils.imageFolder2mpeg('aligned_frames2', fps=30)
```

Part 4: Create background panorama

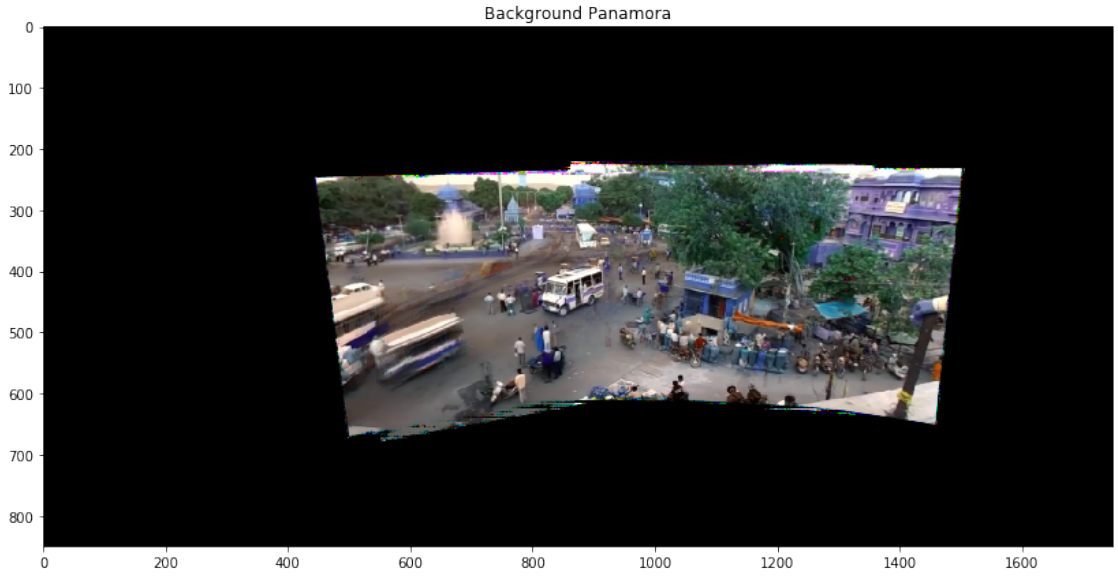
[50]:
```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

[51]:
```
dir_frames = 'aligned_frames2'
filenames = []
filesinfo = os.scandir(dir_frames)

filenames = [f.path for f in filesinfo if f.name.endswith(".jpg")]
filenames.sort(key=lambda f: int(''.join(filter(str.isdigit, f))))

frameCount = len(filenames)
panoramaFrames = np.zeros((frameCount, projectedHeight, projectedWidth,␣
↪frameChannels),dtype=np.float32)
print("Scanning")
for idx, file_i in enumerate(filenames):
    panoramaFrames[idx] = cv2.imread(file_i) / 255.0

backgroundPanorama = np.zeros((projectedHeight, projectedWidth, 3))
print("Starting")
for a in range(projectedHeight):
    for b in range(projectedWidth):
        for c in range(3):
            pixel = np.nonzero(panoramaFrames[:, a, b, c])
            if len(pixel) > 0:
                backgroundPanorama[a][b][c] = np.median(panoramaFrames[pixel,␣
↪a, b, c])

print("Done")
```

```
Scanning
Starting
Done
```

[52]:
```
fig, axes = plt.subplots(1, 1)
fig.set_size_inches(15, 7)

axes.imshow(backgroundPanorama)
axes.set_title("Background Panamora")
cv2.imwrite("part4_2.jpg", backgroundPanorama*255)
```

[52]:
```
True
```



Part 5: Create background movie

[53]:
```
import os
import cv2
import numpy as np
```

[54]:
```
os.mkdir("part5_2")
background_images = np.zeros((900, 360, 480,3))

for idx in range(0, 723):
    invertedH = np.linalg.inv(H_t.dot(pastHomographies[idx][450]))
    projected_image = cv2.warpPerspective(backgroundPanorama, invertedH, (480,␣
↪360))
    background_images[idx,:,:,:] = projected_image
    cv2.imwrite('part5_2/a{:04d}.jpg'.format(idx), projected_image*255)
```

[55]:
```
utils.imageFolder2mpeg('part5_2', fps=30)
```

Part 6: Create foreground movie

[56]:
```
import os
import cv2
import numpy as np
```

[57]:
```
os.mkdir("part6_2")
for idx in range(0, 723):
    difference = frames[idx] - background_images[idx]
    cv2.imwrite('part6_2/a{:04d}.jpg'.format(idx), difference*255)
```

[58]:
```
utils.imageFolder2mpeg('part6_2', fps=30)
```

[ ]:
```

```