Project #5: Video Stitching and Processing

CS445: Computational Photography

Setup

In [2]:

```
import ffmpeg
import cv2
import numpy as np
import os
from numpy.linalg import svd, inv
import utils
%matplotlib inline
from matplotlib import pyplot as plt
```

Part I: Stitch two key frames

This involves:

1. compute homography H between two frames (reference frame: 450);
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.

In [3]:

```
def score_projection(pt1, pt2):
  '''
  Score corresponding to the number of inliers for RANSAC
  Input: pt1 and pt2 are 2xN arrays of N points such that pt1[:, i] and pt2[:,i] should
  Outputs: score (scalar count of inliers) and inliers (1xN logical array)
  '''

  # TO DO
  threshold = 3
  score = 0
  inliers = np.zeros(pt1.shape[1], dtype=bool)
  for i in range(pt1.shape[1]):
    if np.linalg.norm(pt1[:,i] - pt2[:,i]) < threshold:
      score += 1
      inliers[i] = True

  return score, inliers
```

In [4]:

```
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()
    sift = cv2.SIFT_create() # for opencv 4.1.8

    # 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
    n_to_sample = 4 # Put the correct number of points here

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

        # print("numMatches: ", numMatches)
        # print("pts1: ", pts1)
        # print("pts2: ", pts2)

        H_t = homography_func(pts1, pts2, normalization_func) # edit helper code below (


        # score homography
        Xb_ = np.dot(H_t, Xa) # project points from first image to second using H

        score_t, inliers_t = score_projection(Xb[:2,:]/Xb[2,:], Xb_[:2,:]/Xb_[2,:])

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

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

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

    return H
```

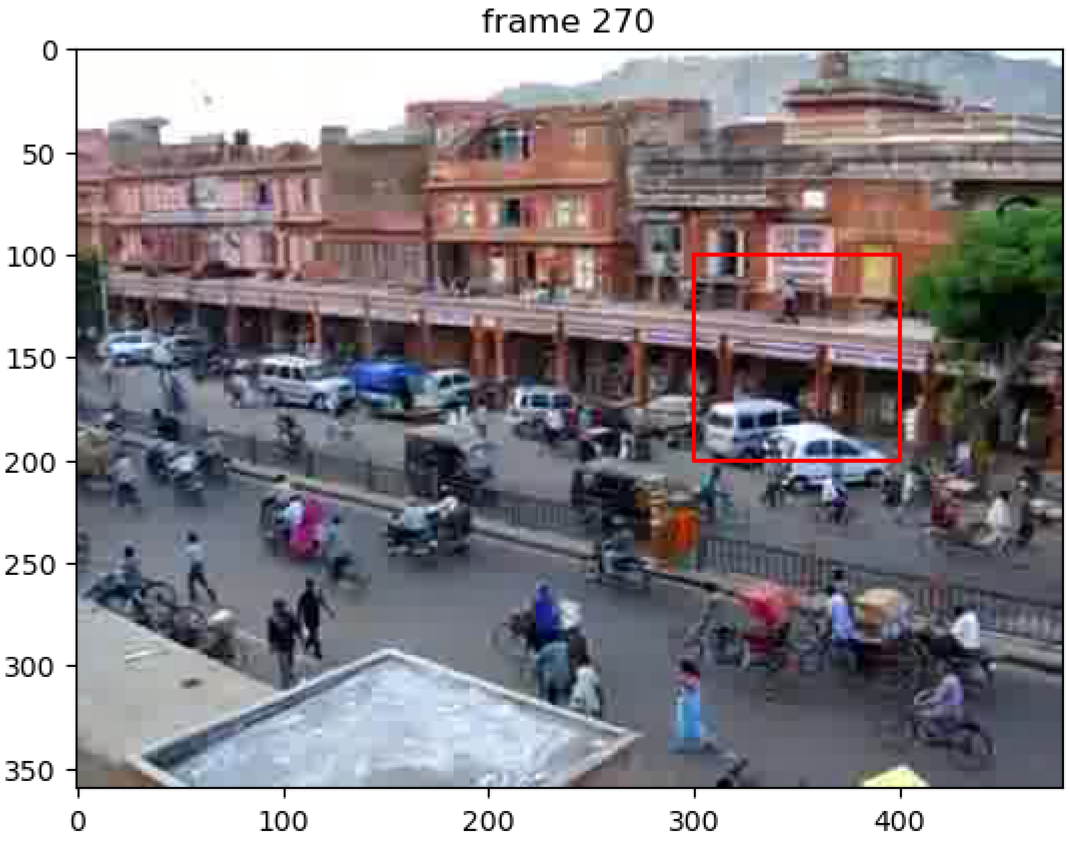
In [5]:

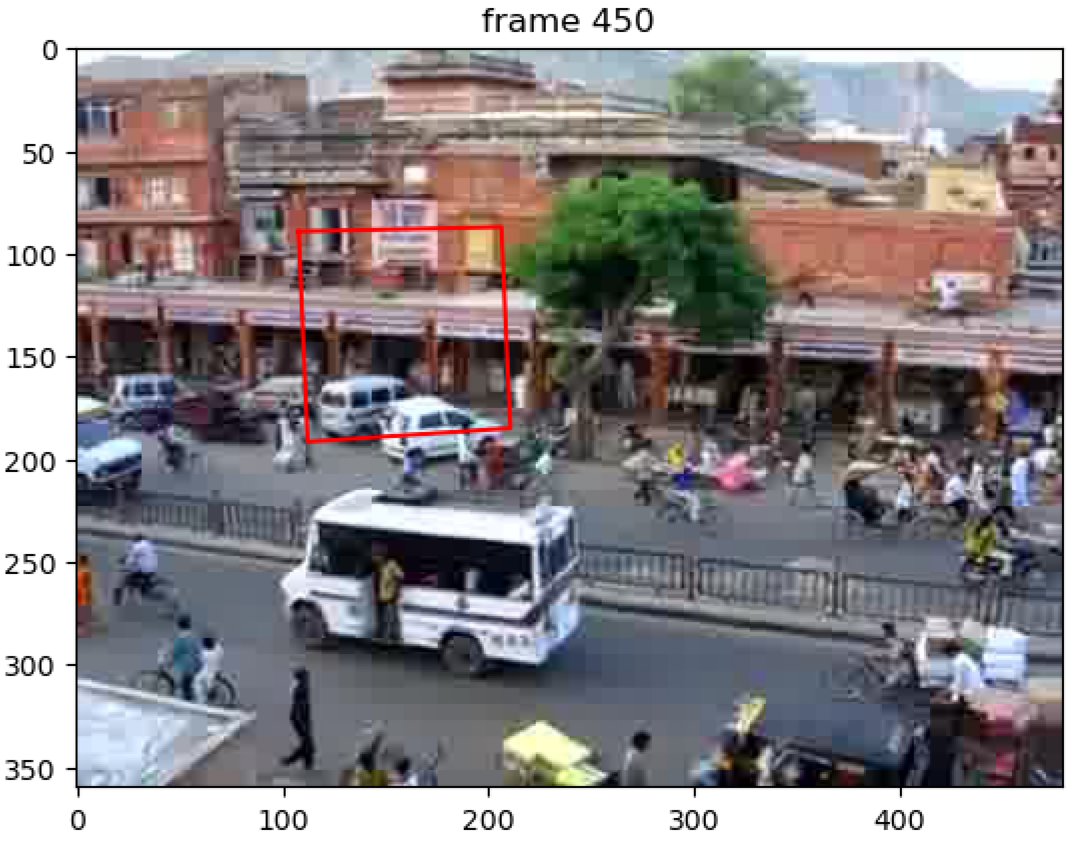
```python
def computeHomography(pts1, pts2,normalization_func=None):
    '''
    Compute homography that maps from pts1 to pts2 using SVD. Normalization is optional.

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

    Output: H is a 3x3 matrix, such that pts2~=H*pts1
    '''
    # TO DO
    if normalization_func is not None:
      pts1 = normalization_func(pts1)
      pts2 = normalization_func(pts2)

    # construct A matrix
    A = []
    for i in range(pts1.shape[1]):
        u1, v1 = pts1[0,i], pts1[1,i]
        u2, v2 = pts2[0,i], pts2[1,i]
        A.append([u1, v1, 1, 0, 0, 0, -u2*u1, -u2*v1, -u2])
        A.append([0, 0, 0, u1, v1, 1, -v2*u1, -v2*v1, -v2])
    A = np.array(A)

    # Apply SVD
    U, S, V = np.linalg.svd(A)

    # h is smallest singular value in V
    h = V[-1,:]
    H = h.reshape(3,3)

    return H
```

In [6]:

```python
# 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)

H = auto_homography(im1,im2, computeHomography)
print(H/H.max())

# plot the frames here
box_pts = np.array([[300, 400, 400, 300, 300], [100, 100, 200, 200, 100], [1, 1, 1, 1, 1
plt.figure()
plt.imshow(im1[:,:,[2,1,0]])
plt.plot(box_pts[0,:], box_pts[1, :], 'r-')
plt.title('frame 270')

# TO DO: project points into im2 and display the projected lines on im2
proj_box_pts = np.dot(H, box_pts)
proj_box_pts = proj_box_pts/proj_box_pts[2,:]
plt.figure()
plt.imshow(im2[:,:,[2,1,0]])
plt.plot(proj_box_pts[0,:], proj_box_pts[1, :], 'r-')
plt.title('frame 450')
plt.show()
```

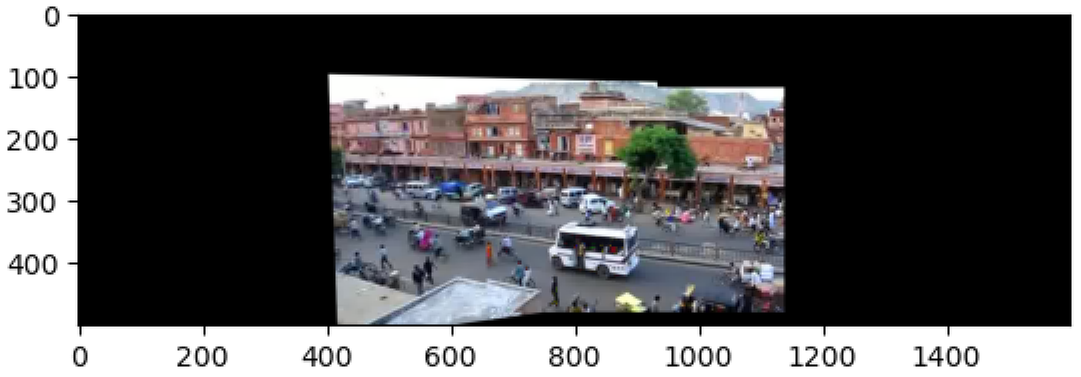
```
best score: 214.000000
[[-4.89723686e-03 -1.88955395e-04  1.00000000e+00]
 [-8.23767398e-05 -4.62337601e-03  8.32232803e-02]
 [-2.05922666e-06  9.39652600e-08 -3.92221457e-03]]
```


frame 270


frame 450

In [7]:

```
projectedWidth = 1600
projectedHeight = 500
Tr = np.array([[1, 0, 660], [0, 1, 120], [0, 0, 1]], dtype=np.float32)

# TO DO: warp and blend the two images
im1_warped = cv2.warpPerspective(im1, np.dot(Tr, H), (projectedWidth, projectedHeight))
im2_warped = cv2.warpPerspective(im2, Tr, (projectedWidth, projectedHeight))
blendOut = utils.blendImages(im1_warped, im2_warped)

plt.figure()
plt.imshow(blendOut[:,:,[2,1,0]])
plt.show()
```



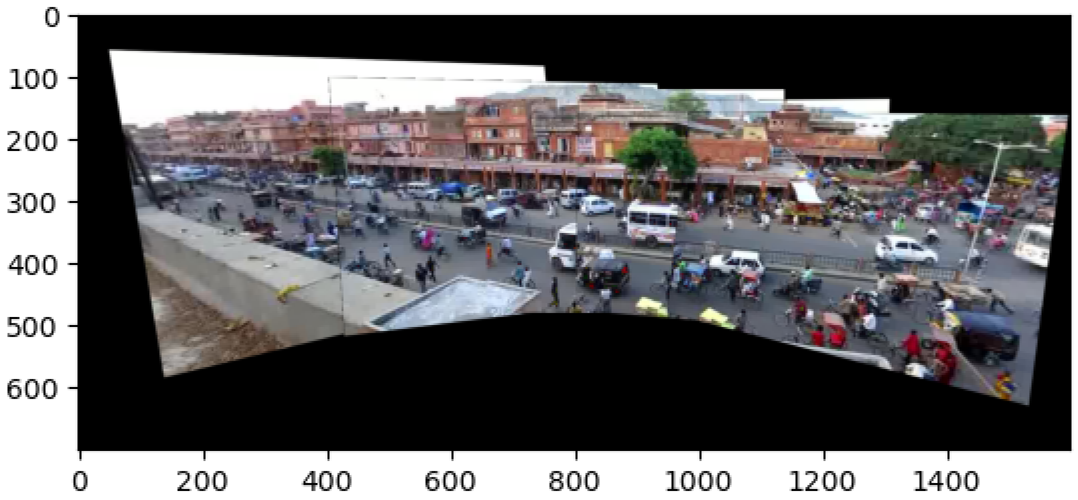
Part II: Panorama using five key frames

Produce a panorama by mapping five key frames [90, 270, 450, 630, 810] onto the same reference frame 450.

In [8]:

```
key_frames_idx = np.array([90, 270, 450, 630, 810])-1

frames = np.zeros((len(key_frames_idx), im1.shape[0], im1.shape[1], im1.shape[2]),dtype=
for n in range(len(key_frames_idx)):
  frames[n] = cv2.imread("./images/input/frames/f0{num}.jpg".format(num=str(key_frames_i

### TO DO solution
projectedWidth = 1600
projectedHeight = 700
Tr = np.array([[1, 0, 660], [0, 1, 120], [0, 0, 1]], dtype=np.float32)

homographies = [None] * 5
homographies[2] = np.eye(3) # reference frame
ref_frame = frames[2] # frame 450

# homographics for frames 270 and 630
homographies[1] = auto_homography(frames[1], ref_frame, computeHomography)
homographies[3] = auto_homography(frames[3], ref_frame, computeHomography)

# homographics for frames 90 and 810
# two stage mapping by using frame 270/630 as a guide
H_90_to_270 = auto_homography(frames[0], frames[1], computeHomography)
homographies[0] = np.dot(homographies[1], H_90_to_270)

H_810_to_630 = auto_homography(frames[4], frames[3], computeHomography)
homographies[4] = np.dot(homographies[3], H_810_to_630)

# warp and blend the frames
warped_frames = [None] * 5
for n in range(5):
  warped_frames[n] = cv2.warpPerspective(frames[n], np.dot(Tr, homographies[n]), (projec

# blend the frames
panorama = np.zeros((projectedHeight, projectedWidth, 3), dtype='uint8')
for n in range(5):
  panorama = utils.blendImages(panorama, warped_frames[n])

plt.figure()
plt.imshow(panorama[:,:,[2,1,0]])
plt.show()
```

```
best score: 213.000000
best score: 211.000000
best score: 256.000000
best score: 149.000000
```



Part 3: Map the video to the reference plane

Project each frame onto the reference frame (using same size panorama) to create a video that shows the portion of the panorama revealed by each frame

In [9]:

```
# read all the images
import os
dir_frames = 'images/input/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
frames = np.zeros((frameCount, frameHeight, frameWidth, frameChannels),dtype='uint8')

for idx, file_i in enumerate(filenames):
  frames[idx] = cv2.imread(file_i)
```

In [12]:

```
### TO DO part 3 solution
# key frames
key_frames_idx = np.array([90, 270, 450, 630, 810])-1
key_homographies = homographies.copy() # homographies from part 2

# construct the homography for each frame
all_homographies = [None] * frameCount
for i in range(frameCount):
    if i in key_frames_idx:
        all_homographies[i] = key_homographies[np.where(key_frames_idx == i)[0][0]]
    else:
        # find the nearest key frame
        nearest_key_frame_idx = np.argmin(np.abs(key_frames_idx - i))
        nearest_key_frame = key_frames_idx[nearest_key_frame_idx]
        print("i: ", i, "; nearest_key_frame: ", nearest_key_frame+1)
        # compute the homography for the ith frame
        H_i_to_key = auto_homography(frames[i], frames[nearest_key_frame], computeHomogr
        all_homographies[i] = np.dot(key_homographies[nearest_key_frame_idx], H_i_to_key

# save the all_homographies
np.save('all_homographies.npy', all_homographies)
```

```
i:  0 ; nearest_key_frame:  90
best score: 427.000000
i:  1 ; nearest_key_frame:  90
best score: 427.000000
i:  2 ; nearest_key_frame:  90
best score: 452.000000
i:  3 ; nearest_key_frame:  90
best score: 439.000000
i:  4 ; nearest_key_frame:  90
best score: 460.000000
i:  5 ; nearest_key_frame:  90
best score: 449.000000
i:  6 ; nearest_key_frame:  90
best score: 481.000000
i:  7 ; nearest_key_frame:  90
best score: 456.000000
i:  8 ; nearest_key_frame:  90
best score: 476.000000
i:  9 ; nearest_key_frame:  90
best score: 495.000000
i:  10 ; nearest_key_frame:  90
best score: 493.000000
i:  11 ; nearest_key_frame:  90
best score: 479.000000
i:  12 ; nearest_key_frame:  90
best score: 494.000000
i:  13 ; nearest_key_frame:  90
best score: 477.000000
i:  14 ; nearest_key_frame:  90
best score: 508.000000
i:  15 ; nearest_key_frame:  90
best score: 505.000000
i:  16 ; nearest_key_frame:  90
best score: 515.000000
i:  17 ; nearest_key_frame:  90
best score: 507.000000
i:  18 ; nearest_key_frame:  90
best score: 490.000000
i:  19 ; nearest_key_frame:  90
best score: 495.000000
i:  20 ; nearest_key_frame:  90
best score: 479.000000
i:  21 ; nearest_key_frame:  90
best score: 490.000000
i:  22 ; nearest_key_frame:  90
best score: 503.000000
i:  23 ; nearest_key_frame:  90
best score: 492.000000
i:  24 ; nearest_key_frame:  90
best score: 516.000000
i:  25 ; nearest_key_frame:  90
best score: 501.000000
i:  26 ; nearest_key_frame:  90
best score: 496.000000
i:  27 ; nearest_key_frame:  90
best score: 524.000000
i:  28 ; nearest_key_frame:  90
best score: 527.000000
i:  29 ; nearest_key_frame:  90
best score: 512.000000
i:  30 ; nearest_key_frame:  90
best score: 521.000000
i:  31 ; nearest_key_frame:  90
best score: 523.000000
i:  32 ; nearest_key_frame:  90
best score: 532.000000
```

```
i:  33 ; nearest_key_frame:  90
best score: 541.000000
i:  34 ; nearest_key_frame:  90
best score: 535.000000
i:  35 ; nearest_key_frame:  90
best score: 548.000000
i:  36 ; nearest_key_frame:  90
best score: 540.000000
i:  37 ; nearest_key_frame:  90
best score: 527.000000
i:  38 ; nearest_key_frame:  90
best score: 521.000000
i:  39 ; nearest_key_frame:  90
best score: 550.000000
i:  40 ; nearest_key_frame:  90
best score: 552.000000
i:  41 ; nearest_key_frame:  90
best score: 550.000000
i:  42 ; nearest_key_frame:  90
best score: 534.000000
i:  43 ; nearest_key_frame:  90
best score: 540.000000
i:  44 ; nearest_key_frame:  90
best score: 545.000000
i:  45 ; nearest_key_frame:  90
best score: 518.000000
i:  46 ; nearest_key_frame:  90
best score: 565.000000
i:  47 ; nearest_key_frame:  90
best score: 555.000000
i:  48 ; nearest_key_frame:  90
best score: 544.000000
i:  49 ; nearest_key_frame:  90
best score: 554.000000
i:  50 ; nearest_key_frame:  90
best score: 541.000000
i:  51 ; nearest_key_frame:  90
best score: 557.000000
i:  52 ; nearest_key_frame:  90
best score: 542.000000
i:  53 ; nearest_key_frame:  90
best score: 564.000000
i:  54 ; nearest_key_frame:  90
best score: 566.000000
i:  55 ; nearest_key_frame:  90
best score: 541.000000
i:  56 ; nearest_key_frame:  90
best score: 564.000000
i:  57 ; nearest_key_frame:  90
best score: 594.000000
i:  58 ; nearest_key_frame:  90
best score: 565.000000
i:  59 ; nearest_key_frame:  90
best score: 596.000000
i:  60 ; nearest_key_frame:  90
best score: 561.000000
i:  61 ; nearest_key_frame:  90
best score: 582.000000
i:  62 ; nearest_key_frame:  90
best score: 619.000000
i:  63 ; nearest_key_frame:  90
best score: 597.000000
i:  64 ; nearest_key_frame:  90
best score: 612.000000
i:  65 ; nearest_key_frame:  90
best score: 614.000000
```

```
i:  66 ; nearest_key_frame:  90
best score: 642.000000
i:  67 ; nearest_key_frame:  90
best score: 653.000000
i:  68 ; nearest_key_frame:  90
best score: 638.000000
i:  69 ; nearest_key_frame:  90
best score: 627.000000
i:  70 ; nearest_key_frame:  90
best score: 635.000000
i:  71 ; nearest_key_frame:  90
best score: 617.000000
i:  72 ; nearest_key_frame:  90
best score: 684.000000
i:  73 ; nearest_key_frame:  90
best score: 624.000000
i:  74 ; nearest_key_frame:  90
best score: 674.000000
i:  75 ; nearest_key_frame:  90
best score: 674.000000
i:  76 ; nearest_key_frame:  90
best score: 661.000000
i:  77 ; nearest_key_frame:  90
best score: 645.000000
i:  78 ; nearest_key_frame:  90
best score: 724.000000
i:  79 ; nearest_key_frame:  90
best score: 767.000000
i:  80 ; nearest_key_frame:  90
best score: 730.000000
i:  81 ; nearest_key_frame:  90
best score: 675.000000
i:  82 ; nearest_key_frame:  90
best score: 762.000000
i:  83 ; nearest_key_frame:  90
best score: 750.000000
i:  84 ; nearest_key_frame:  90
best score: 837.000000
i:  85 ; nearest_key_frame:  90
best score: 878.000000
i:  86 ; nearest_key_frame:  90
best score: 969.000000
i:  87 ; nearest_key_frame:  90
best score: 1072.000000
i:  88 ; nearest_key_frame:  90
best score: 1265.000000
i:  90 ; nearest_key_frame:  90
best score: 1194.000000
i:  91 ; nearest_key_frame:  90
best score: 1117.000000
i:  92 ; nearest_key_frame:  90
best score: 851.000000
i:  93 ; nearest_key_frame:  90
best score: 729.000000
i:  94 ; nearest_key_frame:  90
best score: 783.000000
i:  95 ; nearest_key_frame:  90
best score: 662.000000
i:  96 ; nearest_key_frame:  90
best score: 698.000000
i:  97 ; nearest_key_frame:  90
best score: 622.000000
i:  98 ; nearest_key_frame:  90
best score: 656.000000
i:  99 ; nearest_key_frame:  90
best score: 633.000000
```

```
i:  100 ; nearest_key_frame:  90
best score: 610.000000
i:  101 ; nearest_key_frame:  90
best score: 591.000000
i:  102 ; nearest_key_frame:  90
best score: 597.000000
i:  103 ; nearest_key_frame:  90
best score: 682.000000
i:  104 ; nearest_key_frame:  90
best score: 554.000000
i:  105 ; nearest_key_frame:  90
best score: 494.000000
i:  106 ; nearest_key_frame:  90
best score: 538.000000
i:  107 ; nearest_key_frame:  90
best score: 471.000000
i:  108 ; nearest_key_frame:  90
best score: 528.000000
i:  109 ; nearest_key_frame:  90
best score: 473.000000
i:  110 ; nearest_key_frame:  90
best score: 488.000000
i:  111 ; nearest_key_frame:  90
best score: 472.000000
i:  112 ; nearest_key_frame:  90
best score: 492.000000
i:  113 ; nearest_key_frame:  90
best score: 512.000000
i:  114 ; nearest_key_frame:  90
best score: 477.000000
i:  115 ; nearest_key_frame:  90
best score: 530.000000
i:  116 ; nearest_key_frame:  90
best score: 479.000000
i:  117 ; nearest_key_frame:  90
best score: 429.000000
i:  118 ; nearest_key_frame:  90
best score: 465.000000
i:  119 ; nearest_key_frame:  90
best score: 417.000000
i:  120 ; nearest_key_frame:  90
best score: 464.000000
i:  121 ; nearest_key_frame:  90
best score: 423.000000
i:  122 ; nearest_key_frame:  90
best score: 405.000000
i:  123 ; nearest_key_frame:  90
best score: 420.000000
i:  124 ; nearest_key_frame:  90
best score: 419.000000
i:  125 ; nearest_key_frame:  90
best score: 419.000000
i:  126 ; nearest_key_frame:  90
best score: 384.000000
i:  127 ; nearest_key_frame:  90
best score: 430.000000
i:  128 ; nearest_key_frame:  90
best score: 411.000000
i:  129 ; nearest_key_frame:  90
best score: 387.000000
i:  130 ; nearest_key_frame:  90
best score: 388.000000
i:  131 ; nearest_key_frame:  90
best score: 349.000000
i:  132 ; nearest_key_frame:  90
best score: 421.000000
```

```
i:  133 ; nearest_key_frame:  90
best score: 365.000000
i:  134 ; nearest_key_frame:  90
best score: 372.000000
i:  135 ; nearest_key_frame:  90
best score: 350.000000
i:  136 ; nearest_key_frame:  90
best score: 364.000000
i:  137 ; nearest_key_frame:  90
best score: 360.000000
i:  138 ; nearest_key_frame:  90
best score: 347.000000
i:  139 ; nearest_key_frame:  90
best score: 399.000000
i:  140 ; nearest_key_frame:  90
best score: 364.000000
i:  141 ; nearest_key_frame:  90
best score: 345.000000
i:  142 ; nearest_key_frame:  90
best score: 360.000000
i:  143 ; nearest_key_frame:  90
best score: 369.000000
i:  144 ; nearest_key_frame:  90
best score: 368.000000
i:  145 ; nearest_key_frame:  90
best score: 346.000000
i:  146 ; nearest_key_frame:  90
best score: 365.000000
i:  147 ; nearest_key_frame:  90
best score: 361.000000
i:  148 ; nearest_key_frame:  90
best score: 370.000000
i:  149 ; nearest_key_frame:  90
best score: 353.000000
i:  150 ; nearest_key_frame:  90
best score: 349.000000
i:  151 ; nearest_key_frame:  90
best score: 346.000000
i:  152 ; nearest_key_frame:  90
best score: 349.000000
i:  153 ; nearest_key_frame:  90
best score: 346.000000
i:  154 ; nearest_key_frame:  90
best score: 348.000000
i:  155 ; nearest_key_frame:  90
best score: 342.000000
i:  156 ; nearest_key_frame:  90
best score: 330.000000
i:  157 ; nearest_key_frame:  90
best score: 361.000000
i:  158 ; nearest_key_frame:  90
best score: 356.000000
i:  159 ; nearest_key_frame:  90
best score: 347.000000
i:  160 ; nearest_key_frame:  90
best score: 352.000000
i:  161 ; nearest_key_frame:  90
best score: 361.000000
i:  162 ; nearest_key_frame:  90
best score: 358.000000
i:  163 ; nearest_key_frame:  90
best score: 344.000000
i:  164 ; nearest_key_frame:  90
best score: 352.000000
i:  165 ; nearest_key_frame:  90
best score: 357.000000
```

```
i:  166 ; nearest_key_frame:  90
best score: 316.000000
i:  167 ; nearest_key_frame:  90
best score: 359.000000
i:  168 ; nearest_key_frame:  90
best score: 330.000000
i:  169 ; nearest_key_frame:  90
best score: 347.000000
i:  170 ; nearest_key_frame:  90
best score: 327.000000
i:  171 ; nearest_key_frame:  90
best score: 335.000000
i:  172 ; nearest_key_frame:  90
best score: 334.000000
i:  173 ; nearest_key_frame:  90
best score: 327.000000
i:  174 ; nearest_key_frame:  90
best score: 347.000000
i:  175 ; nearest_key_frame:  90
best score: 333.000000
i:  176 ; nearest_key_frame:  90
best score: 340.000000
i:  177 ; nearest_key_frame:  90
best score: 312.000000
i:  178 ; nearest_key_frame:  90
best score: 331.000000
i:  179 ; nearest_key_frame:  90
best score: 298.000000
i:  180 ; nearest_key_frame:  270
best score: 261.000000
i:  181 ; nearest_key_frame:  270
best score: 262.000000
i:  182 ; nearest_key_frame:  270
best score: 273.000000
i:  183 ; nearest_key_frame:  270
best score: 270.000000
i:  184 ; nearest_key_frame:  270
best score: 266.000000
i:  185 ; nearest_key_frame:  270
best score: 291.000000
i:  186 ; nearest_key_frame:  270
best score: 295.000000
i:  187 ; nearest_key_frame:  270
best score: 281.000000
i:  188 ; nearest_key_frame:  270
best score: 264.000000
i:  189 ; nearest_key_frame:  270
best score: 271.000000
i:  190 ; nearest_key_frame:  270
best score: 306.000000
i:  191 ; nearest_key_frame:  270
best score: 292.000000
i:  192 ; nearest_key_frame:  270
best score: 296.000000
i:  193 ; nearest_key_frame:  270
best score: 299.000000
i:  194 ; nearest_key_frame:  270
best score: 299.000000
i:  195 ; nearest_key_frame:  270
best score: 314.000000
i:  196 ; nearest_key_frame:  270
best score: 297.000000
i:  197 ; nearest_key_frame:  270
best score: 317.000000
i:  198 ; nearest_key_frame:  270
best score: 301.000000
```

```
i:  199 ; nearest_key_frame:  270
best score: 301.000000
i:  200 ; nearest_key_frame:  270
best score: 311.000000
i:  201 ; nearest_key_frame:  270
best score: 292.000000
i:  202 ; nearest_key_frame:  270
best score: 325.000000
i:  203 ; nearest_key_frame:  270
best score: 313.000000
i:  204 ; nearest_key_frame:  270
best score: 309.000000
i:  205 ; nearest_key_frame:  270
best score: 300.000000
i:  206 ; nearest_key_frame:  270
best score: 309.000000
i:  207 ; nearest_key_frame:  270
best score: 317.000000
i:  208 ; nearest_key_frame:  270
best score: 299.000000
i:  209 ; nearest_key_frame:  270
best score: 327.000000
i:  210 ; nearest_key_frame:  270
best score: 342.000000
i:  211 ; nearest_key_frame:  270
best score: 293.000000
i:  212 ; nearest_key_frame:  270
best score: 334.000000
i:  213 ; nearest_key_frame:  270
best score: 330.000000
i:  214 ; nearest_key_frame:  270
best score: 313.000000
i:  215 ; nearest_key_frame:  270
best score: 316.000000
i:  216 ; nearest_key_frame:  270
best score: 330.000000
i:  217 ; nearest_key_frame:  270
best score: 321.000000
i:  218 ; nearest_key_frame:  270
best score: 340.000000
i:  219 ; nearest_key_frame:  270
best score: 350.000000
i:  220 ; nearest_key_frame:  270
best score: 366.000000
i:  221 ; nearest_key_frame:  270
best score: 339.000000
i:  222 ; nearest_key_frame:  270
best score: 378.000000
i:  223 ; nearest_key_frame:  270
best score: 331.000000
i:  224 ; nearest_key_frame:  270
best score: 334.000000
i:  225 ; nearest_key_frame:  270
best score: 351.000000
i:  226 ; nearest_key_frame:  270
best score: 340.000000
i:  227 ; nearest_key_frame:  270
best score: 330.000000
i:  228 ; nearest_key_frame:  270
best score: 327.000000
i:  229 ; nearest_key_frame:  270
best score: 345.000000
i:  230 ; nearest_key_frame:  270
best score: 336.000000
i:  231 ; nearest_key_frame:  270
best score: 347.000000
```

```
i:  232 ; nearest_key_frame:  270
best score: 292.000000
i:  233 ; nearest_key_frame:  270
best score: 341.000000
i:  234 ; nearest_key_frame:  270
best score: 344.000000
i:  235 ; nearest_key_frame:  270
best score: 319.000000
i:  236 ; nearest_key_frame:  270
best score: 313.000000
i:  237 ; nearest_key_frame:  270
best score: 332.000000
i:  238 ; nearest_key_frame:  270
best score: 359.000000
i:  239 ; nearest_key_frame:  270
best score: 359.000000
i:  240 ; nearest_key_frame:  270
best score: 328.000000
i:  241 ; nearest_key_frame:  270
best score: 346.000000
i:  242 ; nearest_key_frame:  270
best score: 367.000000
i:  243 ; nearest_key_frame:  270
best score: 392.000000
i:  244 ; nearest_key_frame:  270
best score: 358.000000
i:  245 ; nearest_key_frame:  270
best score: 338.000000
i:  246 ; nearest_key_frame:  270
best score: 375.000000
i:  247 ; nearest_key_frame:  270
best score: 391.000000
i:  248 ; nearest_key_frame:  270
best score: 395.000000
i:  249 ; nearest_key_frame:  270
best score: 404.000000
i:  250 ; nearest_key_frame:  270
best score: 396.000000
i:  251 ; nearest_key_frame:  270
best score: 407.000000
i:  252 ; nearest_key_frame:  270
best score: 420.000000
i:  253 ; nearest_key_frame:  270
best score: 436.000000
i:  254 ; nearest_key_frame:  270
best score: 406.000000
i:  255 ; nearest_key_frame:  270
best score: 407.000000
i:  256 ; nearest_key_frame:  270
best score: 468.000000
i:  257 ; nearest_key_frame:  270
best score: 454.000000
i:  258 ; nearest_key_frame:  270
best score: 461.000000
i:  259 ; nearest_key_frame:  270
best score: 426.000000
i:  260 ; nearest_key_frame:  270
best score: 394.000000
i:  261 ; nearest_key_frame:  270
best score: 441.000000
i:  262 ; nearest_key_frame:  270
best score: 497.000000
i:  263 ; nearest_key_frame:  270
best score: 472.000000
i:  264 ; nearest_key_frame:  270
best score: 479.000000
```

```
i:  265 ; nearest_key_frame:  270
best score: 529.000000
i:  266 ; nearest_key_frame:  270
best score: 538.000000
i:  267 ; nearest_key_frame:  270
best score: 607.000000
i:  268 ; nearest_key_frame:  270
best score: 753.000000
i:  270 ; nearest_key_frame:  270
best score: 832.000000
i:  271 ; nearest_key_frame:  270
best score: 663.000000
i:  272 ; nearest_key_frame:  270
best score: 620.000000
i:  273 ; nearest_key_frame:  270
best score: 559.000000
i:  274 ; nearest_key_frame:  270
best score: 543.000000
i:  275 ; nearest_key_frame:  270
best score: 457.000000
i:  276 ; nearest_key_frame:  270
best score: 480.000000
i:  277 ; nearest_key_frame:  270
best score: 440.000000
i:  278 ; nearest_key_frame:  270
best score: 406.000000
i:  279 ; nearest_key_frame:  270
best score: 427.000000
i:  280 ; nearest_key_frame:  270
best score: 435.000000
i:  281 ; nearest_key_frame:  270
best score: 408.000000
i:  282 ; nearest_key_frame:  270
best score: 419.000000
i:  283 ; nearest_key_frame:  270
best score: 424.000000
i:  284 ; nearest_key_frame:  270
best score: 388.000000
i:  285 ; nearest_key_frame:  270
best score: 398.000000
i:  286 ; nearest_key_frame:  270
best score: 381.000000
i:  287 ; nearest_key_frame:  270
best score: 399.000000
i:  288 ; nearest_key_frame:  270
best score: 405.000000
i:  289 ; nearest_key_frame:  270
best score: 387.000000
i:  290 ; nearest_key_frame:  270
best score: 387.000000
i:  291 ; nearest_key_frame:  270
best score: 381.000000
i:  292 ; nearest_key_frame:  270
best score: 374.000000
i:  293 ; nearest_key_frame:  270
best score: 383.000000
i:  294 ; nearest_key_frame:  270
best score: 396.000000
i:  295 ; nearest_key_frame:  270
best score: 371.000000
i:  296 ; nearest_key_frame:  270
best score: 356.000000
i:  297 ; nearest_key_frame:  270
best score: 363.000000
i:  298 ; nearest_key_frame:  270
best score: 374.000000
```

```
i:  299 ; nearest_key_frame:  270
best score: 387.000000
i:  300 ; nearest_key_frame:  270
best score: 357.000000
i:  301 ; nearest_key_frame:  270
best score: 350.000000
i:  302 ; nearest_key_frame:  270
best score: 329.000000
i:  303 ; nearest_key_frame:  270
best score: 363.000000
i:  304 ; nearest_key_frame:  270
best score: 342.000000
i:  305 ; nearest_key_frame:  270
best score: 367.000000
i:  306 ; nearest_key_frame:  270
best score: 360.000000
i:  307 ; nearest_key_frame:  270
best score: 366.000000
i:  308 ; nearest_key_frame:  270
best score: 371.000000
i:  309 ; nearest_key_frame:  270
best score: 373.000000
i:  310 ; nearest_key_frame:  270
best score: 379.000000
i:  311 ; nearest_key_frame:  270
best score: 399.000000
i:  312 ; nearest_key_frame:  270
best score: 369.000000
i:  313 ; nearest_key_frame:  270
best score: 357.000000
i:  314 ; nearest_key_frame:  270
best score: 363.000000
i:  315 ; nearest_key_frame:  270
best score: 344.000000
i:  316 ; nearest_key_frame:  270
best score: 351.000000
i:  317 ; nearest_key_frame:  270
best score: 341.000000
i:  318 ; nearest_key_frame:  270
best score: 346.000000
i:  319 ; nearest_key_frame:  270
best score: 347.000000
i:  320 ; nearest_key_frame:  270
best score: 364.000000
i:  321 ; nearest_key_frame:  270
best score: 338.000000
i:  322 ; nearest_key_frame:  270
best score: 342.000000
i:  323 ; nearest_key_frame:  270
best score: 345.000000
i:  324 ; nearest_key_frame:  270
best score: 339.000000
i:  325 ; nearest_key_frame:  270
best score: 349.000000
i:  326 ; nearest_key_frame:  270
best score: 369.000000
i:  327 ; nearest_key_frame:  270
best score: 320.000000
i:  328 ; nearest_key_frame:  270
best score: 320.000000
i:  329 ; nearest_key_frame:  270
best score: 338.000000
i:  330 ; nearest_key_frame:  270
best score: 342.000000
i:  331 ; nearest_key_frame:  270
best score: 322.000000
```

```
i:  332 ; nearest_key_frame:  270
best score: 305.000000
i:  333 ; nearest_key_frame:  270
best score: 303.000000
i:  334 ; nearest_key_frame:  270
best score: 329.000000
i:  335 ; nearest_key_frame:  270
best score: 305.000000
i:  336 ; nearest_key_frame:  270
best score: 311.000000
i:  337 ; nearest_key_frame:  270
best score: 330.000000
i:  338 ; nearest_key_frame:  270
best score: 318.000000
i:  339 ; nearest_key_frame:  270
best score: 283.000000
i:  340 ; nearest_key_frame:  270
best score: 286.000000
i:  341 ; nearest_key_frame:  270
best score: 282.000000
i:  342 ; nearest_key_frame:  270
best score: 290.000000
i:  343 ; nearest_key_frame:  270
best score: 277.000000
i:  344 ; nearest_key_frame:  270
best score: 293.000000
i:  345 ; nearest_key_frame:  270
best score: 273.000000
i:  346 ; nearest_key_frame:  270
best score: 288.000000
i:  347 ; nearest_key_frame:  270
best score: 283.000000
i:  348 ; nearest_key_frame:  270
best score: 279.000000
i:  349 ; nearest_key_frame:  270
best score: 300.000000
i:  350 ; nearest_key_frame:  270
best score: 269.000000
i:  351 ; nearest_key_frame:  270
best score: 302.000000
i:  352 ; nearest_key_frame:  270
best score: 309.000000
i:  353 ; nearest_key_frame:  270
best score: 309.000000
i:  354 ; nearest_key_frame:  270
best score: 284.000000
i:  355 ; nearest_key_frame:  270
best score: 279.000000
i:  356 ; nearest_key_frame:  270
best score: 306.000000
i:  357 ; nearest_key_frame:  270
best score: 307.000000
i:  358 ; nearest_key_frame:  270
best score: 300.000000
i:  359 ; nearest_key_frame:  270
best score: 302.000000
i:  360 ; nearest_key_frame:  450
best score: 328.000000
i:  361 ; nearest_key_frame:  450
best score: 339.000000
i:  362 ; nearest_key_frame:  450
best score: 336.000000
i:  363 ; nearest_key_frame:  450
best score: 351.000000
i:  364 ; nearest_key_frame:  450
best score: 322.000000
```

```
i:  365 ; nearest_key_frame:  450
best score: 309.000000
i:  366 ; nearest_key_frame:  450
best score: 294.000000
i:  367 ; nearest_key_frame:  450
best score: 306.000000
i:  368 ; nearest_key_frame:  450
best score: 331.000000
i:  369 ; nearest_key_frame:  450
best score: 317.000000
i:  370 ; nearest_key_frame:  450
best score: 297.000000
i:  371 ; nearest_key_frame:  450
best score: 362.000000
i:  372 ; nearest_key_frame:  450
best score: 320.000000
i:  373 ; nearest_key_frame:  450
best score: 339.000000
i:  374 ; nearest_key_frame:  450
best score: 356.000000
i:  375 ; nearest_key_frame:  450
best score: 327.000000
i:  376 ; nearest_key_frame:  450
best score: 289.000000
i:  377 ; nearest_key_frame:  450
best score: 350.000000
i:  378 ; nearest_key_frame:  450
best score: 334.000000
i:  379 ; nearest_key_frame:  450
best score: 303.000000
i:  380 ; nearest_key_frame:  450
best score: 319.000000
i:  381 ; nearest_key_frame:  450
best score: 325.000000
i:  382 ; nearest_key_frame:  450
best score: 336.000000
i:  383 ; nearest_key_frame:  450
best score: 322.000000
i:  384 ; nearest_key_frame:  450
best score: 338.000000
i:  385 ; nearest_key_frame:  450
best score: 336.000000
i:  386 ; nearest_key_frame:  450
best score: 356.000000
i:  387 ; nearest_key_frame:  450
best score: 382.000000
i:  388 ; nearest_key_frame:  450
best score: 359.000000
i:  389 ; nearest_key_frame:  450
best score: 341.000000
i:  390 ; nearest_key_frame:  450
best score: 342.000000
i:  391 ; nearest_key_frame:  450
best score: 338.000000
i:  392 ; nearest_key_frame:  450
best score: 339.000000
i:  393 ; nearest_key_frame:  450
best score: 358.000000
i:  394 ; nearest_key_frame:  450
best score: 365.000000
i:  395 ; nearest_key_frame:  450
best score: 396.000000
i:  396 ; nearest_key_frame:  450
best score: 382.000000
i:  397 ; nearest_key_frame:  450
best score: 365.000000
```

```
i:  398 ; nearest_key_frame:  450
best score: 334.000000
i:  399 ; nearest_key_frame:  450
best score: 357.000000
i:  400 ; nearest_key_frame:  450
best score: 344.000000
i:  401 ; nearest_key_frame:  450
best score: 346.000000
i:  402 ; nearest_key_frame:  450
best score: 367.000000
i:  403 ; nearest_key_frame:  450
best score: 367.000000
i:  404 ; nearest_key_frame:  450
best score: 376.000000
i:  405 ; nearest_key_frame:  450
best score: 345.000000
i:  406 ; nearest_key_frame:  450
best score: 344.000000
i:  407 ; nearest_key_frame:  450
best score: 347.000000
i:  408 ; nearest_key_frame:  450
best score: 404.000000
i:  409 ; nearest_key_frame:  450
best score: 407.000000
i:  410 ; nearest_key_frame:  450
best score: 401.000000
i:  411 ; nearest_key_frame:  450
best score: 395.000000
i:  412 ; nearest_key_frame:  450
best score: 344.000000
i:  413 ; nearest_key_frame:  450
best score: 422.000000
i:  414 ; nearest_key_frame:  450
best score: 430.000000
i:  415 ; nearest_key_frame:  450
best score: 436.000000
i:  416 ; nearest_key_frame:  450
best score: 434.000000
i:  417 ; nearest_key_frame:  450
best score: 436.000000
i:  418 ; nearest_key_frame:  450
best score: 452.000000
i:  419 ; nearest_key_frame:  450
best score: 432.000000
i:  420 ; nearest_key_frame:  450
best score: 407.000000
i:  421 ; nearest_key_frame:  450
best score: 379.000000
i:  422 ; nearest_key_frame:  450
best score: 381.000000
i:  423 ; nearest_key_frame:  450
best score: 399.000000
i:  424 ; nearest_key_frame:  450
best score: 389.000000
i:  425 ; nearest_key_frame:  450
best score: 411.000000
i:  426 ; nearest_key_frame:  450
best score: 453.000000
i:  427 ; nearest_key_frame:  450
best score: 438.000000
i:  428 ; nearest_key_frame:  450
best score: 434.000000
i:  429 ; nearest_key_frame:  450
best score: 405.000000
i:  430 ; nearest_key_frame:  450
best score: 437.000000
```

```
i:  431 ; nearest_key_frame:  450
best score: 456.000000
i:  432 ; nearest_key_frame:  450
best score: 422.000000
i:  433 ; nearest_key_frame:  450
best score: 451.000000
i:  434 ; nearest_key_frame:  450
best score: 487.000000
i:  435 ; nearest_key_frame:  450
best score: 497.000000
i:  436 ; nearest_key_frame:  450
best score: 484.000000
i:  437 ; nearest_key_frame:  450
best score: 451.000000
i:  438 ; nearest_key_frame:  450
best score: 443.000000
i:  439 ; nearest_key_frame:  450
best score: 502.000000
i:  440 ; nearest_key_frame:  450
best score: 527.000000
i:  441 ; nearest_key_frame:  450
best score: 486.000000
i:  442 ; nearest_key_frame:  450
best score: 521.000000
i:  443 ; nearest_key_frame:  450
best score: 532.000000
i:  444 ; nearest_key_frame:  450
best score: 525.000000
i:  445 ; nearest_key_frame:  450
best score: 532.000000
i:  446 ; nearest_key_frame:  450
best score: 578.000000
i:  447 ; nearest_key_frame:  450
best score: 644.000000
i:  448 ; nearest_key_frame:  450
best score: 905.000000
i:  450 ; nearest_key_frame:  450
best score: 654.000000
i:  451 ; nearest_key_frame:  450
best score: 558.000000
i:  452 ; nearest_key_frame:  450
best score: 506.000000
i:  453 ; nearest_key_frame:  450
best score: 494.000000
i:  454 ; nearest_key_frame:  450
best score: 444.000000
i:  455 ; nearest_key_frame:  450
best score: 435.000000
i:  456 ; nearest_key_frame:  450
best score: 412.000000
i:  457 ; nearest_key_frame:  450
best score: 418.000000
i:  458 ; nearest_key_frame:  450
best score: 417.000000
i:  459 ; nearest_key_frame:  450
best score: 418.000000
i:  460 ; nearest_key_frame:  450
best score: 484.000000
i:  461 ; nearest_key_frame:  450
best score: 463.000000
i:  462 ; nearest_key_frame:  450
best score: 480.000000
i:  463 ; nearest_key_frame:  450
best score: 464.000000
i:  464 ; nearest_key_frame:  450
best score: 421.000000
```

```
i:  465 ; nearest_key_frame:  450
best score: 396.000000
i:  466 ; nearest_key_frame:  450
best score: 404.000000
i:  467 ; nearest_key_frame:  450
best score: 392.000000
i:  468 ; nearest_key_frame:  450
best score: 416.000000
i:  469 ; nearest_key_frame:  450
best score: 357.000000
i:  470 ; nearest_key_frame:  450
best score: 390.000000
i:  471 ; nearest_key_frame:  450
best score: 380.000000
i:  472 ; nearest_key_frame:  450
best score: 414.000000
i:  473 ; nearest_key_frame:  450
best score: 420.000000
i:  474 ; nearest_key_frame:  450
best score: 391.000000
i:  475 ; nearest_key_frame:  450
best score: 368.000000
i:  476 ; nearest_key_frame:  450
best score: 361.000000
i:  477 ; nearest_key_frame:  450
best score: 379.000000
i:  478 ; nearest_key_frame:  450
best score: 361.000000
i:  479 ; nearest_key_frame:  450
best score: 372.000000
i:  480 ; nearest_key_frame:  450
best score: 394.000000
i:  481 ; nearest_key_frame:  450
best score: 390.000000
i:  482 ; nearest_key_frame:  450
best score: 386.000000
i:  483 ; nearest_key_frame:  450
best score: 375.000000
i:  484 ; nearest_key_frame:  450
best score: 358.000000
i:  485 ; nearest_key_frame:  450
best score: 366.000000
i:  486 ; nearest_key_frame:  450
best score: 390.000000
i:  487 ; nearest_key_frame:  450
best score: 402.000000
i:  488 ; nearest_key_frame:  450
best score: 368.000000
i:  489 ; nearest_key_frame:  450
best score: 377.000000
i:  490 ; nearest_key_frame:  450
best score: 379.000000
i:  491 ; nearest_key_frame:  450
best score: 358.000000
i:  492 ; nearest_key_frame:  450
best score: 362.000000
i:  493 ; nearest_key_frame:  450
best score: 356.000000
i:  494 ; nearest_key_frame:  450
best score: 356.000000
i:  495 ; nearest_key_frame:  450
best score: 362.000000
i:  496 ; nearest_key_frame:  450
best score: 362.000000
i:  497 ; nearest_key_frame:  450
best score: 351.000000
```

```
i:  498 ; nearest_key_frame:  450
best score: 341.000000
i:  499 ; nearest_key_frame:  450
best score: 319.000000
i:  500 ; nearest_key_frame:  450
best score: 335.000000
i:  501 ; nearest_key_frame:  450
best score: 335.000000
i:  502 ; nearest_key_frame:  450
best score: 330.000000
i:  503 ; nearest_key_frame:  450
best score: 332.000000
i:  504 ; nearest_key_frame:  450
best score: 316.000000
i:  505 ; nearest_key_frame:  450
best score: 318.000000
i:  506 ; nearest_key_frame:  450
best score: 349.000000
i:  507 ; nearest_key_frame:  450
best score: 341.000000
i:  508 ; nearest_key_frame:  450
best score: 336.000000
i:  509 ; nearest_key_frame:  450
best score: 335.000000
i:  510 ; nearest_key_frame:  450
best score: 343.000000
i:  511 ; nearest_key_frame:  450
best score: 333.000000
i:  512 ; nearest_key_frame:  450
best score: 316.000000
i:  513 ; nearest_key_frame:  450
best score: 351.000000
i:  514 ; nearest_key_frame:  450
best score: 331.000000
i:  515 ; nearest_key_frame:  450
best score: 322.000000
i:  516 ; nearest_key_frame:  450
best score: 327.000000
i:  517 ; nearest_key_frame:  450
best score: 306.000000
i:  518 ; nearest_key_frame:  450
best score: 302.000000
i:  519 ; nearest_key_frame:  450
best score: 317.000000
i:  520 ; nearest_key_frame:  450
best score: 320.000000
i:  521 ; nearest_key_frame:  450
best score: 339.000000
i:  522 ; nearest_key_frame:  450
best score: 346.000000
i:  523 ; nearest_key_frame:  450
best score: 340.000000
i:  524 ; nearest_key_frame:  450
best score: 342.000000
i:  525 ; nearest_key_frame:  450
best score: 315.000000
i:  526 ; nearest_key_frame:  450
best score: 307.000000
i:  527 ; nearest_key_frame:  450
best score: 299.000000
i:  528 ; nearest_key_frame:  450
best score: 300.000000
i:  529 ; nearest_key_frame:  450
best score: 309.000000
i:  530 ; nearest_key_frame:  450
best score: 322.000000
```

```
i:  531 ; nearest_key_frame:  450
best score: 300.000000
i:  532 ; nearest_key_frame:  450
best score: 323.000000
i:  533 ; nearest_key_frame:  450
best score: 322.000000
i:  534 ; nearest_key_frame:  450
best score: 348.000000
i:  535 ; nearest_key_frame:  450
best score: 344.000000
i:  536 ; nearest_key_frame:  450
best score: 311.000000
i:  537 ; nearest_key_frame:  450
best score: 273.000000
i:  538 ; nearest_key_frame:  450
best score: 284.000000
i:  539 ; nearest_key_frame:  450
best score: 299.000000
i:  540 ; nearest_key_frame:  630
best score: 241.000000
i:  541 ; nearest_key_frame:  630
best score: 260.000000
i:  542 ; nearest_key_frame:  630
best score: 260.000000
i:  543 ; nearest_key_frame:  630
best score: 272.000000
i:  544 ; nearest_key_frame:  630
best score: 273.000000
i:  545 ; nearest_key_frame:  630
best score: 289.000000
i:  546 ; nearest_key_frame:  630
best score: 269.000000
i:  547 ; nearest_key_frame:  630
best score: 248.000000
i:  548 ; nearest_key_frame:  630
best score: 284.000000
i:  549 ; nearest_key_frame:  630
best score: 244.000000
i:  550 ; nearest_key_frame:  630
best score: 230.000000
i:  551 ; nearest_key_frame:  630
best score: 253.000000
i:  552 ; nearest_key_frame:  630
best score: 249.000000
i:  553 ; nearest_key_frame:  630
best score: 239.000000
i:  554 ; nearest_key_frame:  630
best score: 240.000000
i:  555 ; nearest_key_frame:  630
best score: 249.000000
i:  556 ; nearest_key_frame:  630
best score: 252.000000
i:  557 ; nearest_key_frame:  630
best score: 266.000000
i:  558 ; nearest_key_frame:  630
best score: 267.000000
i:  559 ; nearest_key_frame:  630
best score: 257.000000
i:  560 ; nearest_key_frame:  630
best score: 283.000000
i:  561 ; nearest_key_frame:  630
best score: 268.000000
i:  562 ; nearest_key_frame:  630
best score: 271.000000
i:  563 ; nearest_key_frame:  630
best score: 264.000000
```

```
i:  564 ; nearest_key_frame:  630
best score: 269.000000
i:  565 ; nearest_key_frame:  630
best score: 262.000000
i:  566 ; nearest_key_frame:  630
best score: 257.000000
i:  567 ; nearest_key_frame:  630
best score: 258.000000
i:  568 ; nearest_key_frame:  630
best score: 269.000000
i:  569 ; nearest_key_frame:  630
best score: 240.000000
i:  570 ; nearest_key_frame:  630
best score: 273.000000
i:  571 ; nearest_key_frame:  630
best score: 258.000000
i:  572 ; nearest_key_frame:  630
best score: 257.000000
i:  573 ; nearest_key_frame:  630
best score: 274.000000
i:  574 ; nearest_key_frame:  630
best score: 270.000000
i:  575 ; nearest_key_frame:  630
best score: 262.000000
i:  576 ; nearest_key_frame:  630
best score: 247.000000
i:  577 ; nearest_key_frame:  630
best score: 261.000000
i:  578 ; nearest_key_frame:  630
best score: 279.000000
i:  579 ; nearest_key_frame:  630
best score: 261.000000
i:  580 ; nearest_key_frame:  630
best score: 261.000000
i:  581 ; nearest_key_frame:  630
best score: 247.000000
i:  582 ; nearest_key_frame:  630
best score: 267.000000
i:  583 ; nearest_key_frame:  630
best score: 259.000000
i:  584 ; nearest_key_frame:  630
best score: 267.000000
i:  585 ; nearest_key_frame:  630
best score: 295.000000
i:  586 ; nearest_key_frame:  630
best score: 275.000000
i:  587 ; nearest_key_frame:  630
best score: 260.000000
i:  588 ; nearest_key_frame:  630
best score: 261.000000
i:  589 ; nearest_key_frame:  630
best score: 266.000000
i:  590 ; nearest_key_frame:  630
best score: 260.000000
i:  591 ; nearest_key_frame:  630
best score: 258.000000
i:  592 ; nearest_key_frame:  630
best score: 284.000000
i:  593 ; nearest_key_frame:  630
best score: 259.000000
i:  594 ; nearest_key_frame:  630
best score: 269.000000
i:  595 ; nearest_key_frame:  630
best score: 262.000000
i:  596 ; nearest_key_frame:  630
best score: 285.000000
```

```
i:  597 ; nearest_key_frame:  630
best score: 276.000000
i:  598 ; nearest_key_frame:  630
best score: 251.000000
i:  599 ; nearest_key_frame:  630
best score: 246.000000
i:  600 ; nearest_key_frame:  630
best score: 284.000000
i:  601 ; nearest_key_frame:  630
best score: 279.000000
i:  602 ; nearest_key_frame:  630
best score: 283.000000
i:  603 ; nearest_key_frame:  630
best score: 274.000000
i:  604 ; nearest_key_frame:  630
best score: 281.000000
i:  605 ; nearest_key_frame:  630
best score: 311.000000
i:  606 ; nearest_key_frame:  630
best score: 335.000000
i:  607 ; nearest_key_frame:  630
best score: 358.000000
i:  608 ; nearest_key_frame:  630
best score: 345.000000
i:  609 ; nearest_key_frame:  630
best score: 329.000000
i:  610 ; nearest_key_frame:  630
best score: 354.000000
i:  611 ; nearest_key_frame:  630
best score: 357.000000
i:  612 ; nearest_key_frame:  630
best score: 325.000000
i:  613 ; nearest_key_frame:  630
best score: 347.000000
i:  614 ; nearest_key_frame:  630
best score: 351.000000
i:  615 ; nearest_key_frame:  630
best score: 376.000000
i:  616 ; nearest_key_frame:  630
best score: 441.000000
i:  617 ; nearest_key_frame:  630
best score: 420.000000
i:  618 ; nearest_key_frame:  630
best score: 417.000000
i:  619 ; nearest_key_frame:  630
best score: 370.000000
i:  620 ; nearest_key_frame:  630
best score: 385.000000
i:  621 ; nearest_key_frame:  630
best score: 420.000000
i:  622 ; nearest_key_frame:  630
best score: 515.000000
i:  623 ; nearest_key_frame:  630
best score: 472.000000
i:  624 ; nearest_key_frame:  630
best score: 480.000000
i:  625 ; nearest_key_frame:  630
best score: 548.000000
i:  626 ; nearest_key_frame:  630
best score: 643.000000
i:  627 ; nearest_key_frame:  630
best score: 730.000000
i:  628 ; nearest_key_frame:  630
best score: 923.000000
i:  630 ; nearest_key_frame:  630
best score: 774.000000
```

```
i:  631 ; nearest_key_frame:  630
best score: 586.000000
i:  632 ; nearest_key_frame:  630
best score: 538.000000
i:  633 ; nearest_key_frame:  630
best score: 489.000000
i:  634 ; nearest_key_frame:  630
best score: 465.000000
i:  635 ; nearest_key_frame:  630
best score: 432.000000
i:  636 ; nearest_key_frame:  630
best score: 414.000000
i:  637 ; nearest_key_frame:  630
best score: 394.000000
i:  638 ; nearest_key_frame:  630
best score: 383.000000
i:  639 ; nearest_key_frame:  630
best score: 348.000000
i:  640 ; nearest_key_frame:  630
best score: 385.000000
i:  641 ; nearest_key_frame:  630
best score: 490.000000
i:  642 ; nearest_key_frame:  630
best score: 345.000000
i:  643 ; nearest_key_frame:  630
best score: 352.000000
i:  644 ; nearest_key_frame:  630
best score: 329.000000
i:  645 ; nearest_key_frame:  630
best score: 314.000000
i:  646 ; nearest_key_frame:  630
best score: 383.000000
i:  647 ; nearest_key_frame:  630
best score: 308.000000
i:  648 ; nearest_key_frame:  630
best score: 322.000000
i:  649 ; nearest_key_frame:  630
best score: 299.000000
i:  650 ; nearest_key_frame:  630
best score: 313.000000
i:  651 ; nearest_key_frame:  630
best score: 295.000000
i:  652 ; nearest_key_frame:  630
best score: 293.000000
i:  653 ; nearest_key_frame:  630
best score: 298.000000
i:  654 ; nearest_key_frame:  630
best score: 286.000000
i:  655 ; nearest_key_frame:  630
best score: 283.000000
i:  656 ; nearest_key_frame:  630
best score: 276.000000
i:  657 ; nearest_key_frame:  630
best score: 280.000000
i:  658 ; nearest_key_frame:  630
best score: 258.000000
i:  659 ; nearest_key_frame:  630
best score: 284.000000
i:  660 ; nearest_key_frame:  630
best score: 296.000000
i:  661 ; nearest_key_frame:  630
best score: 270.000000
i:  662 ; nearest_key_frame:  630
best score: 277.000000
i:  663 ; nearest_key_frame:  630
best score: 272.000000
```

```
i:  664 ; nearest_key_frame:  630
best score: 274.000000
i:  665 ; nearest_key_frame:  630
best score: 237.000000
i:  666 ; nearest_key_frame:  630
best score: 272.000000
i:  667 ; nearest_key_frame:  630
best score: 254.000000
i:  668 ; nearest_key_frame:  630
best score: 281.000000
i:  669 ; nearest_key_frame:  630
best score: 283.000000
i:  670 ; nearest_key_frame:  630
best score: 273.000000
i:  671 ; nearest_key_frame:  630
best score: 255.000000
i:  672 ; nearest_key_frame:  630
best score: 261.000000
i:  673 ; nearest_key_frame:  630
best score: 261.000000
i:  674 ; nearest_key_frame:  630
best score: 237.000000
i:  675 ; nearest_key_frame:  630
best score: 263.000000
i:  676 ; nearest_key_frame:  630
best score: 239.000000
i:  677 ; nearest_key_frame:  630
best score: 253.000000
i:  678 ; nearest_key_frame:  630
best score: 275.000000
i:  679 ; nearest_key_frame:  630
best score: 245.000000
i:  680 ; nearest_key_frame:  630
best score: 259.000000
i:  681 ; nearest_key_frame:  630
best score: 255.000000
i:  682 ; nearest_key_frame:  630
best score: 272.000000
i:  683 ; nearest_key_frame:  630
best score: 266.000000
i:  684 ; nearest_key_frame:  630
best score: 250.000000
i:  685 ; nearest_key_frame:  630
best score: 254.000000
i:  686 ; nearest_key_frame:  630
best score: 232.000000
i:  687 ; nearest_key_frame:  630
best score: 268.000000
i:  688 ; nearest_key_frame:  630
best score: 232.000000
i:  689 ; nearest_key_frame:  630
best score: 229.000000
i:  690 ; nearest_key_frame:  630
best score: 230.000000
i:  691 ; nearest_key_frame:  630
best score: 227.000000
i:  692 ; nearest_key_frame:  630
best score: 224.000000
i:  693 ; nearest_key_frame:  630
best score: 213.000000
i:  694 ; nearest_key_frame:  630
best score: 230.000000
i:  695 ; nearest_key_frame:  630
best score: 228.000000
i:  696 ; nearest_key_frame:  630
best score: 232.000000
```

```
i:  697 ; nearest_key_frame:  630
best score: 215.000000
i:  698 ; nearest_key_frame:  630
best score: 233.000000
i:  699 ; nearest_key_frame:  630
best score: 235.000000
i:  700 ; nearest_key_frame:  630
best score: 241.000000
i:  701 ; nearest_key_frame:  630
best score: 249.000000
i:  702 ; nearest_key_frame:  630
best score: 234.000000
i:  703 ; nearest_key_frame:  630
best score: 235.000000
i:  704 ; nearest_key_frame:  630
best score: 229.000000
i:  705 ; nearest_key_frame:  630
best score: 226.000000
i:  706 ; nearest_key_frame:  630
best score: 235.000000
i:  707 ; nearest_key_frame:  630
best score: 232.000000
i:  708 ; nearest_key_frame:  630
best score: 219.000000
i:  709 ; nearest_key_frame:  630
best score: 230.000000
i:  710 ; nearest_key_frame:  630
best score: 229.000000
i:  711 ; nearest_key_frame:  630
best score: 239.000000
i:  712 ; nearest_key_frame:  630
best score: 223.000000
i:  713 ; nearest_key_frame:  630
best score: 225.000000
i:  714 ; nearest_key_frame:  630
best score: 222.000000
i:  715 ; nearest_key_frame:  630
best score: 231.000000
i:  716 ; nearest_key_frame:  630
best score: 219.000000
i:  717 ; nearest_key_frame:  630
best score: 212.000000
i:  718 ; nearest_key_frame:  630
best score: 213.000000
i:  719 ; nearest_key_frame:  630
best score: 205.000000
i:  720 ; nearest_key_frame:  810
best score: 224.000000
i:  721 ; nearest_key_frame:  810
best score: 258.000000
i:  722 ; nearest_key_frame:  810
best score: 235.000000
i:  723 ; nearest_key_frame:  810
best score: 240.000000
i:  724 ; nearest_key_frame:  810
best score: 249.000000
i:  725 ; nearest_key_frame:  810
best score: 247.000000
i:  726 ; nearest_key_frame:  810
best score: 230.000000
i:  727 ; nearest_key_frame:  810
best score: 269.000000
i:  728 ; nearest_key_frame:  810
best score: 265.000000
i:  729 ; nearest_key_frame:  810
best score: 251.000000
```

```
i:  730 ; nearest_key_frame:  810
best score: 264.000000
i:  731 ; nearest_key_frame:  810
best score: 258.000000
i:  732 ; nearest_key_frame:  810
best score: 262.000000
i:  733 ; nearest_key_frame:  810
best score: 271.000000
i:  734 ; nearest_key_frame:  810
best score: 285.000000
i:  735 ; nearest_key_frame:  810
best score: 230.000000
i:  736 ; nearest_key_frame:  810
best score: 277.000000
i:  737 ; nearest_key_frame:  810
best score: 273.000000
i:  738 ; nearest_key_frame:  810
best score: 257.000000
i:  739 ; nearest_key_frame:  810
best score: 252.000000
i:  740 ; nearest_key_frame:  810
best score: 240.000000
i:  741 ; nearest_key_frame:  810
best score: 257.000000
i:  742 ; nearest_key_frame:  810
best score: 275.000000
i:  743 ; nearest_key_frame:  810
best score: 281.000000
i:  744 ; nearest_key_frame:  810
best score: 314.000000
i:  745 ; nearest_key_frame:  810
best score: 290.000000
i:  746 ; nearest_key_frame:  810
best score: 266.000000
i:  747 ; nearest_key_frame:  810
best score: 258.000000
i:  748 ; nearest_key_frame:  810
best score: 282.000000
i:  749 ; nearest_key_frame:  810
best score: 248.000000
i:  750 ; nearest_key_frame:  810
best score: 264.000000
i:  751 ; nearest_key_frame:  810
best score: 226.000000
i:  752 ; nearest_key_frame:  810
best score: 252.000000
i:  753 ; nearest_key_frame:  810
best score: 297.000000
i:  754 ; nearest_key_frame:  810
best score: 247.000000
i:  755 ; nearest_key_frame:  810
best score: 260.000000
i:  756 ; nearest_key_frame:  810
best score: 281.000000
i:  757 ; nearest_key_frame:  810
best score: 273.000000
i:  758 ; nearest_key_frame:  810
best score: 284.000000
i:  759 ; nearest_key_frame:  810
best score: 272.000000
i:  760 ; nearest_key_frame:  810
best score: 270.000000
i:  761 ; nearest_key_frame:  810
best score: 253.000000
i:  762 ; nearest_key_frame:  810
best score: 290.000000
```

```
i:  763 ; nearest_key_frame:  810
best score: 323.000000
i:  764 ; nearest_key_frame:  810
best score: 286.000000
i:  765 ; nearest_key_frame:  810
best score: 310.000000
i:  766 ; nearest_key_frame:  810
best score: 292.000000
i:  767 ; nearest_key_frame:  810
best score: 311.000000
i:  768 ; nearest_key_frame:  810
best score: 316.000000
i:  769 ; nearest_key_frame:  810
best score: 312.000000
i:  770 ; nearest_key_frame:  810
best score: 300.000000
i:  771 ; nearest_key_frame:  810
best score: 351.000000
i:  772 ; nearest_key_frame:  810
best score: 340.000000
i:  773 ; nearest_key_frame:  810
best score: 346.000000
i:  774 ; nearest_key_frame:  810
best score: 313.000000
i:  775 ; nearest_key_frame:  810
best score: 302.000000
i:  776 ; nearest_key_frame:  810
best score: 315.000000
i:  777 ; nearest_key_frame:  810
best score: 361.000000
i:  778 ; nearest_key_frame:  810
best score: 358.000000
i:  779 ; nearest_key_frame:  810
best score: 361.000000
i:  780 ; nearest_key_frame:  810
best score: 371.000000
i:  781 ; nearest_key_frame:  810
best score: 324.000000
i:  782 ; nearest_key_frame:  810
best score: 346.000000
i:  783 ; nearest_key_frame:  810
best score: 330.000000
i:  784 ; nearest_key_frame:  810
best score: 325.000000
i:  785 ; nearest_key_frame:  810
best score: 339.000000
i:  786 ; nearest_key_frame:  810
best score: 331.000000
i:  787 ; nearest_key_frame:  810
best score: 339.000000
i:  788 ; nearest_key_frame:  810
best score: 357.000000
i:  789 ; nearest_key_frame:  810
best score: 341.000000
i:  790 ; nearest_key_frame:  810
best score: 347.000000
i:  791 ; nearest_key_frame:  810
best score: 335.000000
i:  792 ; nearest_key_frame:  810
best score: 336.000000
i:  793 ; nearest_key_frame:  810
best score: 346.000000
i:  794 ; nearest_key_frame:  810
best score: 355.000000
i:  795 ; nearest_key_frame:  810
best score: 417.000000
```

```
i:  796 ; nearest_key_frame:  810
best score: 415.000000
i:  797 ; nearest_key_frame:  810
best score: 347.000000
i:  798 ; nearest_key_frame:  810
best score: 374.000000
i:  799 ; nearest_key_frame:  810
best score: 368.000000
i:  800 ; nearest_key_frame:  810
best score: 389.000000
i:  801 ; nearest_key_frame:  810
best score: 390.000000
i:  802 ; nearest_key_frame:  810
best score: 412.000000
i:  803 ; nearest_key_frame:  810
best score: 431.000000
i:  804 ; nearest_key_frame:  810
best score: 439.000000
i:  805 ; nearest_key_frame:  810
best score: 528.000000
i:  806 ; nearest_key_frame:  810
best score: 622.000000
i:  807 ; nearest_key_frame:  810
best score: 802.000000
i:  808 ; nearest_key_frame:  810
best score: 1048.000000
i:  810 ; nearest_key_frame:  810
best score: 1079.000000
i:  811 ; nearest_key_frame:  810
best score: 600.000000
i:  812 ; nearest_key_frame:  810
best score: 547.000000
i:  813 ; nearest_key_frame:  810
best score: 465.000000
i:  814 ; nearest_key_frame:  810
best score: 439.000000
i:  815 ; nearest_key_frame:  810
best score: 399.000000
i:  816 ; nearest_key_frame:  810
best score: 365.000000
i:  817 ; nearest_key_frame:  810
best score: 348.000000
i:  818 ; nearest_key_frame:  810
best score: 357.000000
i:  819 ; nearest_key_frame:  810
best score: 328.000000
i:  820 ; nearest_key_frame:  810
best score: 321.000000
i:  821 ; nearest_key_frame:  810
best score: 333.000000
i:  822 ; nearest_key_frame:  810
best score: 319.000000
i:  823 ; nearest_key_frame:  810
best score: 324.000000
i:  824 ; nearest_key_frame:  810
best score: 309.000000
i:  825 ; nearest_key_frame:  810
best score: 305.000000
i:  826 ; nearest_key_frame:  810
best score: 328.000000
i:  827 ; nearest_key_frame:  810
best score: 333.000000
i:  828 ; nearest_key_frame:  810
best score: 334.000000
i:  829 ; nearest_key_frame:  810
best score: 306.000000
```

```
i:  830 ; nearest_key_frame:  810
best score: 296.000000
i:  831 ; nearest_key_frame:  810
best score: 294.000000
i:  832 ; nearest_key_frame:  810
best score: 284.000000
i:  833 ; nearest_key_frame:  810
best score: 286.000000
i:  834 ; nearest_key_frame:  810
best score: 312.000000
i:  835 ; nearest_key_frame:  810
best score: 304.000000
i:  836 ; nearest_key_frame:  810
best score: 304.000000
i:  837 ; nearest_key_frame:  810
best score: 274.000000
i:  838 ; nearest_key_frame:  810
best score: 269.000000
i:  839 ; nearest_key_frame:  810
best score: 278.000000
i:  840 ; nearest_key_frame:  810
best score: 284.000000
i:  841 ; nearest_key_frame:  810
best score: 266.000000
i:  842 ; nearest_key_frame:  810
best score: 263.000000
i:  843 ; nearest_key_frame:  810
best score: 250.000000
i:  844 ; nearest_key_frame:  810
best score: 261.000000
i:  845 ; nearest_key_frame:  810
best score: 269.000000
i:  846 ; nearest_key_frame:  810
best score: 293.000000
i:  847 ; nearest_key_frame:  810
best score: 278.000000
i:  848 ; nearest_key_frame:  810
best score: 271.000000
i:  849 ; nearest_key_frame:  810
best score: 261.000000
i:  850 ; nearest_key_frame:  810
best score: 269.000000
i:  851 ; nearest_key_frame:  810
best score: 258.000000
i:  852 ; nearest_key_frame:  810
best score: 264.000000
i:  853 ; nearest_key_frame:  810
best score: 253.000000
i:  854 ; nearest_key_frame:  810
best score: 254.000000
i:  855 ; nearest_key_frame:  810
best score: 252.000000
i:  856 ; nearest_key_frame:  810
best score: 262.000000
i:  857 ; nearest_key_frame:  810
best score: 274.000000
i:  858 ; nearest_key_frame:  810
best score: 248.000000
i:  859 ; nearest_key_frame:  810
best score: 264.000000
i:  860 ; nearest_key_frame:  810
best score: 255.000000
i:  861 ; nearest_key_frame:  810
best score: 262.000000
i:  862 ; nearest_key_frame:  810
best score: 247.000000
```

```
i:  863 ; nearest_key_frame:  810
best score: 255.000000
i:  864 ; nearest_key_frame:  810
best score: 277.000000
i:  865 ; nearest_key_frame:  810
best score: 268.000000
i:  866 ; nearest_key_frame:  810
best score: 264.000000
i:  867 ; nearest_key_frame:  810
best score: 272.000000
i:  868 ; nearest_key_frame:  810
best score: 270.000000
i:  869 ; nearest_key_frame:  810
best score: 271.000000
i:  870 ; nearest_key_frame:  810
best score: 272.000000
i:  871 ; nearest_key_frame:  810
best score: 235.000000
i:  872 ; nearest_key_frame:  810
best score: 250.000000
i:  873 ; nearest_key_frame:  810
best score: 263.000000
i:  874 ; nearest_key_frame:  810
best score: 251.000000
i:  875 ; nearest_key_frame:  810
best score: 263.000000
i:  876 ; nearest_key_frame:  810
best score: 271.000000
i:  877 ; nearest_key_frame:  810
best score: 272.000000
i:  878 ; nearest_key_frame:  810
best score: 263.000000
i:  879 ; nearest_key_frame:  810
best score: 256.000000
i:  880 ; nearest_key_frame:  810
best score: 249.000000
i:  881 ; nearest_key_frame:  810
best score: 268.000000
i:  882 ; nearest_key_frame:  810
best score: 240.000000
i:  883 ; nearest_key_frame:  810
best score: 245.000000
i:  884 ; nearest_key_frame:  810
best score: 253.000000
i:  885 ; nearest_key_frame:  810
best score: 213.000000
i:  886 ; nearest_key_frame:  810
best score: 220.000000
i:  887 ; nearest_key_frame:  810
best score: 221.000000
i:  888 ; nearest_key_frame:  810
best score: 231.000000
i:  889 ; nearest_key_frame:  810
best score: 228.000000
i:  890 ; nearest_key_frame:  810
best score: 214.000000
i:  891 ; nearest_key_frame:  810
best score: 232.000000
i:  892 ; nearest_key_frame:  810
best score: 222.000000
i:  893 ; nearest_key_frame:  810
best score: 223.000000
i:  894 ; nearest_key_frame:  810
best score: 250.000000
i:  895 ; nearest_key_frame:  810
best score: 234.000000
```

```
i:  896 ; nearest_key_frame:  810
best score: 237.000000
i:  897 ; nearest_key_frame:  810
best score: 240.000000
i:  898 ; nearest_key_frame:  810
best score: 205.000000
i:  899 ; nearest_key_frame:  810
best score: 220.000000
```

In [15]:

```
# warp the frames and save to output
projectedWidth = 2000
projectedHeight = 800
Tr = np.array([[1, 0, 700], [0, 1, 120], [0, 0, 1]], dtype=np.float32)

output_dir = 'images/output/frames'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

for i in range(frameCount):
    warped_frame = cv2.warpPerspective(frames[i], np.dot(Tr, all_homographies[i]), (proj
    cv2.imwrite(os.path.join(output_dir, 'f{num}.jpg'.format(num=str(i+1).zfill(4))), wa
```

In [17]:

```
# make it video
(
    ffmpeg
    .input('images/output/frames/f%04d.jpg', framerate=30)
    .output('panorama.mp4')
    .run()
)
```

```
ffmpeg version 4.2.2 Copyright (c) 2000-2019 the FFmpeg developers
  built with gcc 7.3.0 (crosstool-NG 1.23.0.449-a04d0)
  configuration: --prefix=/home/baoyu/anaconda3/envs/dinov2 --cc=/tmp/build/80754af9/ffm
peg_1587154242452/_build_env/bin/x86_64-conda_cos6-linux-gnu-cc --disable-doc --enable-a
vresample --enable-gmp --enable-hardcoded-tables --enable-libfreetype --enable-libvpx --
enable-pthreads --enable-libopus --enable-postproc --enable-pic --enable-pthreads --enab
le-shared --enable-static --enable-version3 --enable-zlib --enable-libmp3lame --disable-
nonfree --enable-gpl --enable-gnutls --disable-openssl --enable-libopenh264 --enable-lib
x264
  libavutil      56. 31.100 / 56. 31.100
  libavcodec     58. 54.100 / 58. 54.100
  libavformat    58. 29.100 / 58. 29.100
  libavdevice    58.  8.100 / 58.  8.100
  libavfilter     7. 57.100 /  7. 57.100
  libavresample   4.  0.  0 /  4.  0.  0
  libswscale      5.  5.100 /  5.  5.100
  libswresample   3.  5.100 /  3.  5.100
  libpostproc    55.  5.100 / 55.  5.100
Input #0, image2, from 'images/output/frames/f%04d.jpg':
  Duration: 00:00:30.00, start: 0.000000, bitrate: N/A
    Stream #0:0: Video: mjpeg (Baseline), yuvj420p(pc, bt470bg/unknown/unknown), 2000x80
0 [SAR 1:1 DAR 5:2], 30 fps, 30 tbr, 30 tbn, 30 tbc
Stream mapping:
  Stream #0:0 -> #0:0 (mjpeg (native) -> h264 (libx264))
Press [q] to stop, [?] for help
[libx264 @ 0x55fcf8aa8300] using SAR=1/1
[libx264 @ 0x55fcf8aa8300] using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX FMA3 B
MI2 AVX2
[libx264 @ 0x55fcf8aa8300] profile High, level 4.0, 4:2:0, 8-bit
[libx264 @ 0x55fcf8aa8300] 264 - core 157 - H.264/MPEG-4 AVC codec - Copyleft 2003-2018
- http://www.videolan.org/x264.html - options: cabac=1 ref=3 deblock=1:0:0 analyse=0x3:0
x113 me=hex subme=7 psy=1 psy_rd=1.00:0.00 mixed_ref=1 me_range=16 chroma_me=1 trellis=1
8x8dct=1 cqm=0 deadzone=21,11 fast_pskip=1 chroma_qp_offset=-2 threads=18 lookahead_thre
ads=3 sliced_threads=0 nr=0 decimate=1 interlaced=0 bluray_compat=0 constrained_intra=0
bframes=3 b_pyramid=2 b_adapt=1 b_bias=0 direct=1 weightb=1 open_gop=0 weightp=2 keyint=
250 keyint_min=25 scenecut=40 intra_refresh=0 rc_lookahead=40 rc=crf mbtree=1 crf=23.0 q
comp=0.60 qpmin=0 qpmax=69 qpstep=4 ip_ratio=1.40 aq=1:1.00
Output #0, mp4, to 'panorama.mp4':
  Metadata:
    encoder         : Lavf58.29.100
    Stream #0:0: Video: h264 (libx264) (avc1 / 0x31637661), yuvj420p(pc), 2000x800 [SAR
1:1 DAR 5:2], q=-1--1, 30 fps, 15360 tbn, 30 tbc
    Metadata:
      encoder         : Lavc58.54.100 libx264
    Side data:
      cpb: bitrate max/min/avg: 0/0/0 buffer size: 0 vbv_delay: -1
frame=  900 fps=227 q=-1.0 Lsize=    9256kB time=00:00:29.90 bitrate=2536.0kbits/s speed
=7.55x
video:9245kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhea
d: 0.124479%
[libx264 @ 0x55fcf8aa8300] frame I:4     Avg QP:18.89  size: 36522
[libx264 @ 0x55fcf8aa8300] frame P:227   Avg QP:24.47  size: 18752
[libx264 @ 0x55fcf8aa8300] frame B:669   Avg QP:28.67  size:  7568
[libx264 @ 0x55fcf8aa8300] consecutive B-frames:  0.9%  0.0%  0.0% 99.1%
[libx264 @ 0x55fcf8aa8300] mb I  I16..4: 33.7% 60.7%  5.6%
[libx264 @ 0x55fcf8aa8300] mb P  I16..4:  0.2%  1.4%  0.6%  P16..4:  5.2%  5.5%  2.7%
0.0%  0.0%    skip:84.4%
[libx264 @ 0x55fcf8aa8300] mb B  I16..4:  0.1%  0.1%  0.0%  B16..8:  7.8%  3.3%  0.8%  d
irect: 1.9%  skip:86.1%  L0:38.9% L1:34.5% BI:26.7%
[libx264 @ 0x55fcf8aa8300] 8x8 transform intra:61.6% inter:54.9%
[libx264 @ 0x55fcf8aa8300] coded y,uvDC,uvAC intra: 41.0% 36.3% 17.2% inter: 8.2% 4.5%
0.3%
[libx264 @ 0x55fcf8aa8300] i16 v,h,dc,p: 72% 19%  6%  3%
[libx264 @ 0x55fcf8aa8300] i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 25% 19% 35%  2%  4%  3%  6%
2%  5%
[libx264 @ 0x55fcf8aa8300] i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 20% 28% 15%  3%  6%  6%  9%
5%  7%
[libx264 @ 0x55fcf8aa8300] i8c dc,h,v,p: 73% 16%  9%  2%
[libx264 @ 0x55fcf8aa8300] Weighted P-Frames: Y:4.8% UV:1.3%
[libx264 @ 0x55fcf8aa8300] ref P L0: 41.3% 12.4% 33.0% 12.6%  0.7%
[libx264 @ 0x55fcf8aa8300] ref B L0: 80.7% 15.4%  3.9%
[libx264 @ 0x55fcf8aa8300] ref B L1: 95.0%  5.0%
[libx264 @ 0x55fcf8aa8300] kb/s:2524.26
```

Out[17]:

```
(None, None)
```

Part 4: Create background panorama

Create a background panorama based on the result from Part 3.

In [47]:

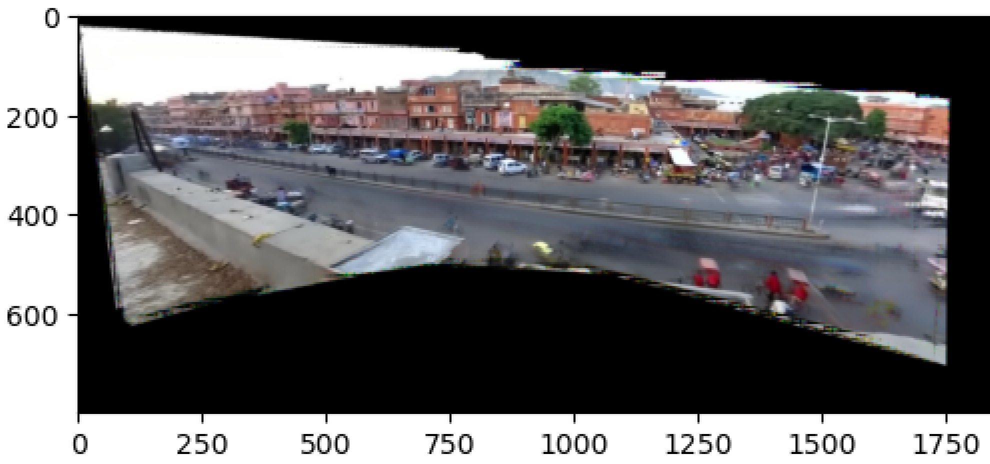
```
# TO DO part 4
# read all the images
import os
transformed_dir_frames = 'images/output/frames'
transformed_filenames = []
transformed_filesinfo = os.scandir(transformed_dir_frames)

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

transformed_frameCount = len(transformed_filenames)
transformed_frameHeight, transformed_frameWidth, transformed_frameChannels = cv2.imread(
transformed_frames = np.zeros((transformed_frameCount, transformed_frameHeight, transfor

for idx, file_i in enumerate(transformed_filenames):
  transformed_frames[idx] = cv2.imread(file_i)
```

In [48]:

```
# construct the background panorama
bg_panorama = np.zeros((transformed_frameHeight, transformed_frameWidth, 3), dtype='uint
for i in range(transformed_frameHeight):
    for j in range(transformed_frameWidth):
        pixel_values = transformed_frames[:,i,j,:].astype(np.int64)
        for c in range(3):
            # find the non-zero pixel values
            non_zero_pixel_values = pixel_values[:,c][pixel_values[:,c] > 0]
            # take the median of non-zero pixel values as bg pixel value
            if len(non_zero_pixel_values) > 0:
                bg_panorama[i,j,c] = np.median(non_zero_pixel_values)
```

In [49]:

```
plt.figure()
plt.imshow(bg_panorama[:,:,[2,1,0]])
plt.show()
```



Part 5: Create background movie

Generate a movie that looks like the input movie but shows only background pixels. For each frame of the movie, you need to estimate a projection from the panorama to that frame. Your solution can use the background image you created in Part 4 and the per-frame homographies you created in Part 3.

In [50]:

```
# TO DO part 5
# load the homographies in Part 3
all_homographies = np.load('all_homographies.npy')
# create the background frames dir
output_dir = "images/output/bg_frames"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# generate the bg frames
video_frameHeight, video_frameWidth, _ = cv2.imread("./images/input/frames/f0450.jpg").s
for i in range(transformed_frameCount):
    H_inv = np.linalg.inv(np.dot(Tr,all_homographies[i]))
    warped_frame = cv2.warpPerspective(bg_panorama, H_inv, (video_frameWidth, video_fram
    cv2.imwrite(os.path.join(output_dir, f"f{str(i+1).zfill(4)}.jpg"), warped_frame)
```

In [37]:

```
# make it video
(
    ffmpeg
    .input('images/output/bg_frames/f%04d.jpg', framerate=30)
    .output('bg_panorama.mp4')
    .run()
)
```

```
ffmpeg version 4.2.2 Copyright (c) 2000-2019 the FFmpeg developers
  built with clang version 12.0.0
  configuration: --prefix=/Users/ktietz/demo/mc3/conda-bld/ffmpeg_1628925491858/_h_env_p
lacehold_placehold_placehold_placehold_placehold_placehold_placehold_placehold_placehold
_placehold_placehold_placehold_placehold_placehold_placehold_placehold_placehold_placeho
ld_placehold_plac --cc=arm64-apple-darwin20.0.0-clang --disable-doc --enable-avresample
--enable-gmp --enable-hardcoded-tables --enable-libfreetype --enable-libvpx --enable-pth
reads --enable-libopus --enable-postproc --enable-pic --enable-pthreads --enable-shared
--enable-static --enable-version3 --enable-zlib --enable-libmp3lame --disable-nonfree --
enable-gpl --enable-gnutls --disable-openssl --enable-libopenh264 --enable-libx264
  libavutil      56. 31.100 / 56. 31.100
  libavcodec     58. 54.100 / 58. 54.100
  libavformat    58. 29.100 / 58. 29.100
  libavdevice    58.  8.100 / 58.  8.100
  libavfilter     7. 57.100 /  7. 57.100
  libavresample   4.  0.  0 /  4.  0.  0
  libswscale      5.  5.100 /  5.  5.100
  libswresample   3.  5.100 /  3.  5.100
  libpostproc    55.  5.100 / 55.  5.100
Input #0, image2, from 'images/output/bg_frames/f%04d.jpg':
  Duration: 00:00:30.00, start: 0.000000, bitrate: N/A
    Stream #0:0: Video: mjpeg (Baseline), yuvj420p(pc, bt470bg/unknown/unknown), 480x360
[SAR 1:1 DAR 4:3], 30 fps, 30 tbr, 30 tbn, 30 tbc
Stream mapping:
  Stream #0:0 -> #0:0 (mjpeg (native) -> h264 (libx264))
Press [q] to stop, [?] for help
[libx264 @ 0x12680e400] using SAR=1/1
[libx264 @ 0x12680e400] using cpu capabilities: ARMv8 NEON
[libx264 @ 0x12680e400] profile High, level 3.0
[libx264 @ 0x12680e400] 264 - core 152 - H.264/MPEG-4 AVC codec - Copyleft 2003-2017 - h
ttp://www.videolan.org/x264.html - options: cabac=1 ref=3 deblock=1:0:0 analyse=0x3:0x11
3 me=hex subme=7 psy=1 psy_rd=1.00:0.00 mixed_ref=1 me_range=16 chroma_me=1 trellis=1 8x
8dct=1 cqm=0 deadzone=21,11 fast_pskip=1 chroma_qp_offset=-2 threads=11 lookahead_thread
s=1 sliced_threads=0 nr=0 decimate=1 interlaced=0 bluray_compat=0 constrained_intra=0 bf
rames=3 b_pyramid=2 b_adapt=1 b_bias=0 direct=1 weightb=1 open_gop=0 weightp=2 keyint=25
0 keyint_min=25 scenecut=40 intra_refresh=0 rc_lookahead=40 rc=crf mbtree=1 crf=23.0 qco
mp=0.60 qpmin=0 qpmax=69 qpstep=4 ip_ratio=1.40 aq=1:1.00
Output #0, mp4, to 'bg_panorama.mp4':
  Metadata:
    encoder         : Lavf58.29.100
    Stream #0:0: Video: h264 (libx264) (avc1 / 0x31637661), yuvj420p(pc), 480x360 [SAR
1:1 DAR 4:3], q=-1--1, 30 fps, 15360 tbn, 30 tbc
    Metadata:
      encoder         : Lavc58.54.100 libx264
    Side data:
      cpb: bitrate max/min/avg: 0/0/0 buffer size: 0 vbv_delay: -1
frame=  900 fps=827 q=-1.0 Lsize=     966kB time=00:00:29.90 bitrate= 264.7kbits/s speed
=27.5x
video:955kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead:
1.191504%
[libx264 @ 0x12680e400] frame I:4     Avg QP:20.69  size: 20816
[libx264 @ 0x12680e400] frame P:227   Avg QP:22.99  size:  2327
[libx264 @ 0x12680e400] frame B:669   Avg QP:28.84  size:   546
[libx264 @ 0x12680e400] consecutive B-frames:  0.9%  0.0%  0.0% 99.1%
[libx264 @ 0x12680e400] mb I  I16..4:  2.6% 79.1% 18.2%
[libx264 @ 0x12680e400] mb P  I16..4:  0.2%  0.9%  0.3%  P16..4: 44.6% 15.8% 10.2%  0.0%
  0.0%    skip:28.0%
[libx264 @ 0x12680e400] mb B  I16..4:  0.0%  0.1%  0.0%  B16..8: 48.9%  1.8%  0.3%  dire
ct: 0.3%  skip:48.6%  L0:50.5% L1:48.2% BI: 1.3%
[libx264 @ 0x12680e400] 8x8 transform intra:71.2% inter:79.8%
[libx264 @ 0x12680e400] coded y,uvDC,uvAC intra: 76.1% 74.1% 43.9% inter: 6.3% 5.3% 0.5%
[libx264 @ 0x12680e400] i16 v,h,dc,p: 19% 55% 13% 14%
[libx264 @ 0x12680e400] i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 22% 32% 15%  2%  4%  3% 10%  3%
6%
[libx264 @ 0x12680e400] i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 28% 34% 12%  2%  4%  3%  9%  2%
7%
[libx264 @ 0x12680e400] i8c dc,h,v,p: 49% 31% 13%  7%
[libx264 @ 0x12680e400] Weighted P-Frames: Y:0.0% UV:0.0%
```

```
[libx264 @ 0x12680e400] ref P L0: 59.3% 15.8% 12.4% 12.6%
[libx264 @ 0x12680e400] ref B L0: 82.0% 14.0%  4.1%
[libx264 @ 0x12680e400] ref B L1: 89.8% 10.2%
[libx264 @ 0x12680e400] kb/s:260.53
```

Out[37]:

```
(None, None)
```

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 emphasizes or includes only foreground pixels.

In [58]:

```
360*480
```

Out[58]:

```
172800
```

In [82]:

```
# TO DO part 6
# difference threshold
diff_threshold = 60

# create the foreground frames dir
output_dir = "images/output/fg_frames"
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# generate the fg frames
for i in range(transformed_frameCount):
    # obtain the bg and transformed frames
    H_inv = np.linalg.inv(np.dot(Tr,all_homographies[i]))
    bg = cv2.warpPerspective(bg_panorama, H_inv, (video_frameWidth, video_frameHeight)).
    img = cv2.warpPerspective(transformed_frames[i], H_inv, (video_frameWidth, video_fra

    ## compute the foreground pixels that are different from the background
    # find the zero values in img
    zero_img_idx = np.argwhere(img == 0)
    # compute the difference between bg and img for non-zero pixels
    diff = np.sqrt(((bg - img) ** 2).sum(axis=-1))
    # remove the difference for zero pixels
    diff[zero_img_idx[:,0],zero_img_idx[:,1]]=0

    fg_idx = np.argwhere(diff > diff_threshold)
    # print("fg_idx: ", fg_idx.shape)
    # generate the mask
    mask = np.zeros((video_frameHeight, video_frameWidth,3))
    mask[fg_idx[:,0],fg_idx[:,1],:]=1
    mask = cv2.GaussianBlur(mask, (3,3), cv2.BORDER_DEFAULT)

    foreground = img * mask
    cv2.imwrite(os.path.join(output_dir, f"f{str(i+1).zfill(4)}.jpg"), foreground)
```

In [83]:

```
# make it video
(
    ffmpeg
    .input('images/output/fg_frames/f%04d.jpg', framerate=30)
    .output('fg_panorama.mp4')
    .run()
)
```

```
ffmpeg version 4.2.2 Copyright (c) 2000-2019 the FFmpeg developers
  built with clang version 12.0.0
  configuration: --prefix=/Users/ktietz/demo/mc3/conda-bld/ffmpeg_1628925491858/_h_env_p
lacehold_placehold_placehold_placehold_placehold_placehold_placehold_placehold_placehold
```

```
_placehold_placehold_placehold_placehold_placehold_placehold_placehold_placehold_placeho
ld_placehold_plac --cc=arm64-apple-darwin20.0.0-clang --disable-doc --enable-avresample
--enable-gmp --enable-hardcoded-tables --enable-libfreetype --enable-libvpx --enable-pth
reads --enable-libopus --enable-postproc --enable-pic --enable-pthreads --enable-shared
--enable-static --enable-version3 --enable-zlib --enable-libmp3lame --disable-nonfree --
enable-gpl --enable-gnutls --disable-openssl --enable-libopenh264 --enable-libx264
  libavutil      56. 31.100 / 56. 31.100
  libavcodec     58. 54.100 / 58. 54.100
  libavformat    58. 29.100 / 58. 29.100
  libavdevice    58.  8.100 / 58.  8.100
  libavfilter     7. 57.100 /  7. 57.100
  libavresample   4.  0.  0 /  4.  0.  0
  libswscale      5.  5.100 /  5.  5.100
  libswresample   3.  5.100 /  3.  5.100
  libpostproc    55.  5.100 / 55.  5.100
Input #0, image2, from 'images/output/fg_frames/f%04d.jpg':
  Duration: 00:00:30.00, start: 0.000000, bitrate: N/A
    Stream #0:0: Video: mjpeg (Baseline), yuvj420p(pc, bt470bg/unknown/unknown), 480x360
[SAR 1:1 DAR 4:3], 30 fps, 30 tbr, 30 tbn, 30 tbc
Stream mapping:
  Stream #0:0 -> #0:0 (mjpeg (native) -> h264 (libx264))
Press [q] to stop, [?] for help
[libx264 @ 0x14c022400] using SAR=1/1
[libx264 @ 0x14c022400] using cpu capabilities: ARMv8 NEON
[libx264 @ 0x14c022400] profile High, level 3.0
[libx264 @ 0x14c022400] 264 - core 152 - H.264/MPEG-4 AVC codec - Copyleft 2003-2017 - h
ttp://www.videolan.org/x264.html - options: cabac=1 ref=3 deblock=1:0:0 analyse=0x3:0x11
3 me=hex subme=7 psy=1 psy_rd=1.00:0.00 mixed_ref=1 me_range=16 chroma_me=1 trellis=1 8x
8dct=1 cqm=0 deadzone=21,11 fast_pskip=1 chroma_qp_offset=-2 threads=11 lookahead_thread
s=1 sliced_threads=0 nr=0 decimate=1 interlaced=0 bluray_compat=0 constrained_intra=0 bf
rames=3 b_pyramid=2 b_adapt=1 b_bias=0 direct=1 weightb=1 open_gop=0 weightp=2 keyint=25
0 keyint_min=25 scenecut=40 intra_refresh=0 rc_lookahead=40 rc=crf mbtree=1 crf=23.0 qco
mp=0.60 qpmin=0 qpmax=69 qpstep=4 ip_ratio=1.40 aq=1:1.00
Output #0, mp4, to 'fg_panorama.mp4':
  Metadata:
    encoder         : Lavf58.29.100
    Stream #0:0: Video: h264 (libx264) (avc1 / 0x31637661), yuvj420p(pc), 480x360 [SAR
1:1 DAR 4:3], q=-1--1, 30 fps, 15360 tbn, 30 tbc
    Metadata:
      encoder         : Lavc58.54.100 libx264
    Side data:
      cpb: bitrate max/min/avg: 0/0/0 buffer size: 0 vbv_delay: -1
frame=  900 fps=528 q=-1.0 Lsize=    5345kB time=00:00:29.90 bitrate=1464.3kbits/s speed
=17.5x
video:5333kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhea
d: 0.214406%
[libx264 @ 0x14c022400] frame I:5     Avg QP:24.25  size: 12976
[libx264 @ 0x14c022400] frame P:226   Avg QP:27.80  size:  8307
[libx264 @ 0x14c022400] frame B:669   Avg QP:30.02  size:  5259
[libx264 @ 0x14c022400] consecutive B-frames:  0.7%  0.2%  1.3% 97.8%
[libx264 @ 0x14c022400] mb I  I16..4:  7.5% 46.3% 46.1%
[libx264 @ 0x14c022400] mb P  I16..4:  2.1% 13.7% 11.7%  P16..4: 20.3% 19.8% 10.7%  0.0%
  0.0%    skip:21.8%
[libx264 @ 0x14c022400] mb B  I16..4:  0.8%  4.0%  3.2%  B16..8: 32.9% 22.2%  8.4%  dire
ct: 7.3%  skip:21.3%  L0:47.3% L1:41.9% BI:10.7%
[libx264 @ 0x14c022400] 8x8 transform intra:50.0% inter:34.2%
[libx264 @ 0x14c022400] coded y,uvDC,uvAC intra: 41.6% 13.3% 6.4% inter: 33.9% 6.1% 0.8%
[libx264 @ 0x14c022400] i16 v,h,dc,p: 50% 40%  9%  2%
[libx264 @ 0x14c022400] i8 v,h,dc,ddl,ddr,vr,hd,vl,hu: 18% 13% 58%  2%  2%  2%  2%  1%
2%
[libx264 @ 0x14c022400] i4 v,h,dc,ddl,ddr,vr,hd,vl,hu: 23% 25% 28%  3%  5%  3%  5%  3%
4%
[libx264 @ 0x14c022400] i8c dc,h,v,p: 86%  7%  6%  0%
[libx264 @ 0x14c022400] Weighted P-Frames: Y:23.9% UV:8.4%
[libx264 @ 0x14c022400] ref P L0: 36.0% 20.9% 23.0% 16.8%  3.3%
[libx264 @ 0x14c022400] ref B L0: 73.8% 20.2%  6.0%
```

```
[libx264 @ 0x14c022400] ref B L1: 88.3% 11.7%
[libx264 @ 0x14c022400] kb/s:1456.12
```

```
Out[83]: (None, None)
```

Bells and whistles

```
In [ ]:
```