Dataflow

Painting taken for example

SubElement to be detected:

Our aim is to detect the center of this hotspot on the frame

Mobile Side

On the mobile side

This code starts when the painting is detected in a frame

```
point3 = [[0,0] , [h,0] , [h, w ] , [0, w]]  #'point3' is the coordinates of the image in painting(training data this data we get from canvas)
if found == 0 :    # if the image is detected in the frame
```

For example let's take the following image as the frame


```
    h_, w_,c = frame.shape
    kp, des = brisk.detectAndCompute(frame, None) #detect the interest points
```

From brisk we are able to get the keypoints and descriptors of the frame, which will be used to for matching the keypoints and descriptor obtained from the painting that is detected in the frame

Multiple Keypoints and descriptors are detected

Example of a key point, this is the keypoint for one point:

<KeyPoint 00000179F957E330>

Example of descriptor, this is a descriptor for one point:

```
[160 123 223  15   0 192  97  54 166 140  61 239 115 207 125 239  61 243
 200  33  14 243 176  79 182   1   0   0   1 111 124 227  59 255 255 255
 255 253 211  31 231 156  99  49   0   6   0 129 201 224 246 255 255 255
 255 247 249 124 118  54  27   0   0 128]
```

```
    # Matching the interest point in for the video to the training painting
    bf = cv2.BFMatcher()
    matches = bf.knnMatch(des1,des, k=2)
```

Here we have matched the the descriptors from the painting and the video frame

```
good = []
```
The array 'good' contains the matched points which are same

```
for i, (m, n) in enumerate(matches):
    if m.distance < 0.75*n.distance:
        good.append(m)


if len(good) > 10 :       # threshold for the image to be matched


        src_pts = np.float32([ kp1[m.queryIdx].pt for m in good ]).reshape(-1,1,2) #Contains
location of matched interest points on the painting
        dst_pts = np.float32([ kp[m.trainIdx].pt for m in good ]).reshape(-1,1,2) #Contains
location of matched interest points on the frame

        #changing the value in desired form
        s_ = [(int(b[0][0]),int(b[0][1])) for b in src_pts]   #interest points on the training side
        d_ = [(int(b[0][0]),int(b[0][1])) for b in dst_pts]  #interest point on the video side
```

Sample Location of same matched interest point on
Painting:
(51, 32)
Frame:
(36, 272)

```
        origin = [0,0]
        M, mask = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC,5.0) #Getting the
#perspective transformation
```
Perspective transformation helps to get the coordinates of the bounding box of the detected painting on the frame.

```
        matchesMask = mask.ravel().tolist()
        h,w,c = img.shape       # img == painting: training
        pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)
        dst = cv2.perspectiveTransform(pts,M)


        point1 = np.int32(dst)    # coordinates of the bounding box of the painting
```

The Coordinates of the painting detected on the frame
[[[-484 -181]]

 [[-504 395]]

[[408 293]]

[[404   11]]]

We have negative values because painting here is out of frame, only half the painting is in the frame.

In this example, since we have half the painting only two coordinates(Pink circles) are visible on the screen

```
#Using the coordinates of the detected painting to calculate the width and height of the painting
w1 = point1[2][0][0] - point1[1][0][0]     # width of side 1
w2 = point1[3][0][0] - point1[0][0][0]     # width of side 2
h1 = point1[1][0][1] - point1[0][0][1]     # height of 1st one
h2 = point1[2][0][1] - point1[3][0][1]     # height of 2nd one


#Calculating the hotspots
for i in range(totalNumberofSubElement): for each sub element
```

Point array contains the information about the coordinates of the hotspot on the painting and s_ is the list of the matches hotspot on the painting. . Here we are trying to figure out the closest matched interest point on the painting to the hotpot.

```
p = closest(point[i], s_)     # point[i] is the hotspot on the painting
```

```
                # s_ is the matched interest point on the painting
                'p' is the closest interest point to the hotspot on the painting. Now we need the
coordinated of the interest point that matches the point 'p'

                indx = s_.index(p)  # index of the closest point
                point2 = d_[indx]   # the closest point on the destination image: video

point2 is the coordinate that matches point 'p' on the video frame

Now our job is to find the approximate direction and distance at which we will have our hotspot
from point2
                # Detection of the hot spot
                # explanation of the logic is in the diagram in docs
'point3' contains the edge coordinates of the painting, this data we receive from canvas side
Here we are calculating the distance between the nearest interest point to the hotspot 'p' and
the sides of the painting, we are storing this in information in d1, d2 ,d3 , d4
                 d1 = pointLineDis(p,[point3[0], point3[1]])
                d2 = pointLineDis(p,[point3[1], point3[2]])
                d3 = pointLineDis(p,[point3[2], point3[3]])
                d4 = pointLineDis(p,[point3[3], point3[0]])
Here we are taking the weighted average of the h and w, this helps us to get a better localisation
when the user is standing at extreme left or right
                # taking the weighted average of the height to calculate the position of the hotspot
on the video frame

                h = ( d1*h2 + d2*h1 )/(d1 + d2)
                w = ( d3*w1 + d4*w2 )/(d3 + d4)

 h and  w here are the weighted height and weight
                x2 = int((point[i][0] - p[0])*(h/h3) + point2[0])
                y2 = int((point[i][1] - p[1])*(w/w3) + point2[1])
   x2, y2 is the coordinates of the hotpot on the video frame but how we have to calculate it wrt
to video frame size

                x3 = int(x2*(h_/frame.shape[0])) # Relative to the frame
                y3 = int(y2*(w_/frame.shape[1]))

               image = cv2.circle(image, (x3,y3), 6 , (255), -1)
      found = 1
       else:
          #Tracking the hotspot
```