EE5175 - Lab 6

Shape from focus

Nikilesh B
EE17B112

In this assignment, we are given a set of images which are captured using different camera lens settings, thus resulting in different degrees of optical blur. We use these images to determine the depth of each pixel in the image using the shape from focus (SFF) algorithm discussed in class.

We know that the amount of blur due to optical defocussing depends on the distance between camera and scene. Hence it is possible to use blur as a cue to determine the depth of a particular point in the scene.

We will display the 3D structure of the scene using the calculated depth values.
We will use a sum-modified Laplacian (SML) operator as a focus measure and Δd = 50:50.
And we will also observe the output 3D structure for q = 0, 1 and 2 ;
where q is the size of the neighbourhood for the SML window as defined in class.

At first we load and read the given mat file and extract the images in it. The code for it is attached below:

```
#Loading .mat file
pics = loadmat("stack.mat")
#Loading only the images
imgs = []
for i in range(1,10) :
      img = pics["frame00%1d"%i]
      imgs.append(img)

for i in range(10,100) :
      img = pics["frame0%2d"%i]
      imgs.append(img)

img = pics["frame100"]
imgs.append(img)
imgs = np.array(imgs)
#print(shape(imgs))
```

The function block for sum-modified Laplacian (SML) is attached below :

```
def sml(src,Ngd,deld = 50.50) :
      #Obtain partial double derivates (hessian)
      kfxx = np.array([[0,0,0],[1,-2,1],[0,0,0]])
      kfyy = np.array([[0,1,0],[0,-2,0],[0,1,0]])

      kext = len(kfxx)//2
      Nimg,Nr,Nc = (src.shape)
      img = np.zeros((Nimg,int(Nr+2*kext),int(Nc+2*kext)))
      fin = np.zeros(src.shape)
      img[:,kext:Nr+kext,kext:Nc+kext] = np.copy(src)
      patch = np.zeros(kfxx.shape)
      #Go along rows and then along columns
      for i in range(kext,Nr+kext) :
      for j in range(kext,Nc+kext) :
            patch = img[:,i-kext:i+kext+1,j-kext:j+kext+1]
            patch1 = np.sum(patch*kfxx,axis = (1,2))
            patch2 = np.sum(patch*kfyy,axis = (1,2))
            #Modified by using abs
            patch = abs(patch1) + abs(patch2)
            fin[:,i-kext,j-kext] = patch
      if Ngd != 0 :
      #sum the Ngd
      ksum = np.ones((int(2*Ngd+1),int(2*Ngd+1)))/(2*Ngd+1)
      kext = len(ksum)//2
      Nimg,Nr,Nc = fin.shape
      img = np.zeros((Nimg,int(Nr+2*kext),int(Nc+2*kext)))
      sml = np.zeros(fin.shape)
      img[:,kext:Nr+kext,kext:Nc+kext] = fin
      patch = np.zeros(ksum.shape)
      #Go along rows and then along columns
      for i in range(kext,Nr+kext) :
            for j in range(kext,Nc+kext) :
                  patch = img[:,i-kext:i+kext+1,j-kext:j+kext+1]
                  patch1 = np.sum(patch*ksum,axis = (1,2))
                  sml[:,i-kext,j-kext] = patch1
      else :
      sml = fin
      depth = np.argmax(sml,axis = 0)
      depth = depth*deld
      return depth
```

The depth maps from different angles are attached below: