```
import numpy as np
import cv2
import scipy.io
import os
from numpy.linalg import norm
from matplotlib import pyplot as plt
from numpy.linalg import det
from numpy.linalg import inv
from scipy.linalg import rq
from numpy.linalg import svd
import matplotlib.pyplot as plt
import numpy as np
import math
import random
import sys
from scipy import ndimage, spatial
from tqdm.notebook import tqdm, trange

import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
from torch.autograd import Variable
import torchvision
from torchvision import datasets, models, transforms
from torch.utils.data import Dataset, DataLoader, ConcatDataset
from skimage import io, transform,data
from torchvision import transforms, utils
import numpy as np
import math
import glob
import matplotlib.pyplot as plt
import time
import os
import copy
import sklearn.svm
import cv2
from matplotlib import pyplot as plt
import numpy as np
from os.path import exists
import pandas as pd
import PIL
import random
from google.colab import drive
from sklearn.metrics.cluster import completeness_score
from sklearn.cluster import KMeans
from tqdm import tqdm, tqdm_notebook
from functools import partial
from torchsummary import summary
from torchvision.datasets import ImageFolder
from torch.utils.data.sampler import SubsetRandomSampler
import h5py as h5

#cuda_output = !ldconfig -p|grep cudart.so|sed -e 's/.*\.\([0-9]*\)\.\([0-9]*\)$/cu\1\2/'
#accelerator = cuda_output[0] if exists('/dev/nvidia0') else 'cpu'

#print("Accelerator type = ",accelerator)
#print("Pytorch verision: ", torch.__version__)
```

```
from google.colab import drive

# This will prompt for authorization.
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
#!cp "/content/drive/My Drive/cv2_gpu/cv2.cpython-37m-x86_64-linux-gnu.so" .
```

```
cv2.__version__
```

```
'4.5.3-pre'
```

```
#%%file mprun_demo31.py
import numpy as np
import cv2
import h5py as h5
import tqdm

def final_steps_left_union(len_H_left,xmax,xmin,ymax,ymin,t,h,w,Ht,scale_factor=16):


    for j in range(len_H_left):
      print(j)
      f=h5.File('drive/MyDrive/H_left_sift_220.h5','r')
      H = f['data'][j]
      f.close()
      if j==0:
        H_trans = Ht.dot(H)
      else:
        H_trans = H_trans.dot(H)

      f=h5.File('drive/MyDrive/all_images_bgr_sift_443.h5','r')
      input_img_orig = f['data'][j+1]
      f.close()
      del f
      input_img = cv2.resize(input_img_orig,None,fx=(1/scale_factor),fy = (1/scale_factor),interpolation = cv2.INTER_CUBIC)
      #input_img = cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)
      #print('input image accesssed')

      #input_img = images_left[j+1]
      result = np.zeros((ymax-ymin,xmax-xmin,3),dtype='uint8')
      #print('output init done')

      cv2.warpPerspective(src = np.uint8(input_img), M = H_trans, dsize = (xmax-xmin, ymax-ymin),dst=result)
      del input_img
      warp_img_init_curr = result

      if j==0:
        f=h5.File('drive/MyDrive/all_images_bgr_sift_443.h5','r')
        first_img_orig = f['data'][0]
        f.close()
        del f
```

```
        first_img = cv2.resize(first_img_orig,None,fx=(1/scale_factor),fy = (1/scale_factor),interpolation = cv2.INTER_CUBIC)
        #first_img = cv2.cvtColor(first_img, cv2.COLOR_BGR2GRAY)
        result[t[1]:h+t[1], t[0]:w+t[0]] = first_img
        warp_img_init_prev = result
        continue
      #inds = warp_img_init_prev[:, :] == 0
      del result

      inds = warp_img_init_prev[:, :, 0] == 0
      inds &= warp_img_init_prev[:, :, 1] == 0
      inds &= warp_img_init_prev[:, :, 2] == 0

      #black_pixels = np.where((warp_img_init_prev[:, :, 0] == 0) & (warp_img_init_prev[:, :, 1] == 0) & (warp_img_init_prev[:, :, 2] == 0))

      warp_img_init_prev[inds] = warp_img_init_curr[inds]

      del inds,warp_img_init_curr



    print('Step31:Done')

    return warp_img_init_prev
```

```
def warpnImages(len_H_left,len_H_right,scale_factor=16,offset=0):
    #img1-centre,img2-left,img3-right

    f=h5.File('drive/MyDrive/all_images_bgr_sift_443.h5','r')
    img = f['data'][0]
    f.close()
    h, w = img.shape[:2]
    h = round(h/scale_factor)
    w = round(w/scale_factor)

    pts_left = []
    pts_right = []

    pts_centre = np.float32([[0, 0], [0, h], [w, h], [w, 0]]).reshape(-1, 1, 2)

    for j in range(offset,len_H_left):
      pts = np.float32([[0, 0], [0, h], [w, h], [w, 0]]).reshape(-1, 1, 2)
      pts_left.append(pts)

    for j in range(offset,len_H_right):
      pts = np.float32([[0, 0], [0, h], [w, h], [w, 0]]).reshape(-1, 1, 2)
      pts_right.append(pts)

    pts_left_transformed=[]
    pts_right_transformed=[]


    for j,pts in enumerate(pts_left):
      if j==0:
        f=h5.File('drive/MyDrive/H_left_sift_220.h5','r')
        H_trans = f['data'][j+offset]
        f.close()
        #H_trans = H_left[j]
      else:
        f=h5.File('drive/MyDrive/H_left_sift_220.h5','r')
        H_trans = H_trans@f['data'][j+offset]
        f.close()
        #H_trans = H_trans@H_left[j]
      pts_ = cv2.perspectiveTransform(pts, H_trans)
      pts_left_transformed.append(pts_)

    for j,pts in enumerate(pts_right):
      if j==0:
        f=h5.File('drive/MyDrive/H_right_sift_222.h5','r')
        H_trans = f['data'][j+offset]
        f.close()
        #H_trans = H_right[j]
      else:
        f=h5.File('drive/MyDrive/H_right_sift_222.h5','r')
        H_trans = H_trans@f['data'][j+offset]
        f.close()
        #H_trans = H_trans@H_right[j]
      pts_ = cv2.perspectiveTransform(pts, H_trans)
      pts_right_transformed.append(pts_)


    print('Step1:Done')


    #pts = np.concatenate((pts1, pts2_), axis=0)

    pts_concat = np.concatenate((pts_centre,np.concatenate(np.array(pts_left_transformed),axis=0),np.concatenate(np.array(pts_right_transformed),axis=0)), axis=0)

    [xmin, ymin] = np.int32(pts_concat.min(axis=0).ravel() - 0.5)
    [xmax, ymax] = np.int32(pts_concat.max(axis=0).ravel() + 0.5)
    t = [-xmin, -ymin]
    Ht = np.array([[1, 0, t[0]], [0, 1, t[1]], [0, 0, 1]])  # translate

    print('Step2:Done')


    return xmax,xmin,ymax,ymin,t,h,w,Ht
```

```
#%%file mprun_demo31.py
import numpy as np
import cv2
import h5py as h5
import tqdm

def final_steps_left_union_gpu(len_H_left,xmax,xmin,ymax,ymin,t,h,w,Ht,warp_img_init_prev ,scale_factor=16,is_gray=True,offset=0,H_trans=np.eye(3)):
    from tqdm import tqdm
    tqdm = partial(tqdm, position=0, leave=True)

    for j in tqdm(range(offset,len_H_left)):
      #print(j)
      f=h5.File('drive/MyDrive/H_left_sift_220.h5','r')
      H = f['data'][j]
      f.close()
      if j==0:
        H_trans = Ht.dot(H)
      else:
```

```
      else:
        H_trans = H_trans.dot(H)

      f=h5.File('drive/MyDrive/all_images_bgr_sift_443.h5','r')
      input_img_orig = f['data'][j+1]
      f.close()
      del f
      src = cv2.cuda_GpuMat()
      src.upload( np.uint8(input_img_orig))
      if scale_factor>1:
        dst = cv2.cuda.resize(src,None,fx=(1/scale_factor),fy = (1/scale_factor),interpolation = cv2.INTER_CUBIC)
      else:
        dst = src
      #input_img = dst.download()

      if is_gray==True:
        dst = cv2.cuda.cvtColor(dst, cv2.COLOR_BGR2GRAY)
      #print('input image accesssed')
      input_img = dst.download()

      #input_img = images_left[j+1]
      #result = np.zeros((ymax-ymin,xmax-xmin,3),dtype='uint8')
      #print('output init done')
      src = cv2.cuda_GpuMat()
      src.upload( np.uint8(input_img))



      #print('Step 42: Done')
      #if is_gray==False:
      #  result = np.zeros((ymax-ymin,xmax-xmin,3),dtype='uint8')
      #else:
      #  result = np.zeros((ymax-ymin,xmax-xmin),dtype='uint8')

      #dst = cv2.cuda_GpuMat()
      #dst.upload(result)


      dst = cv2.cuda.warpPerspective(src, M = H_trans, dsize = (xmax-xmin, ymax-ymin) )




      #cv2.warpPerspective(src = np.uint8(input_img), M = H_trans, dsize = (xmax-xmin, ymax-ymin),dst=result)
      del input_img
      result = dst.download()
      warp_img_init_curr = result
      #print('Step 43: Done')

      if j==0:
        f=h5.File('drive/MyDrive/all_images_bgr_sift_443.h5','r')
        first_img_orig = f['data'][0]
        f.close()
        del f
        src = cv2.cuda_GpuMat()
        src.upload(np.uint8(first_img_orig))
        dst = cv2.cuda.resize(src,None,fx=(1/scale_factor),fy = (1/scale_factor),interpolation = cv2.INTER_CUBIC)
        #first_img = dst.download()
        #first_img = cv2.resize(first_img_orig,None,fx=(1/scale_factor),fy = (1/scale_factor),interpolation = cv2.INTER_CUBIC)
        if is_gray==True:
          dst = cv2.cuda.cvtColor(dst, cv2.COLOR_BGR2GRAY)
        first_img = dst.download()
        result[t[1]:h+t[1], t[0]:w+t[0]] = first_img
        warp_img_init_prev = result
        continue
      del result
      #print('Step 44: Done')

      if is_gray==True:
        inds = warp_img_init_prev[:, :] == 0
      else:
        inds = warp_img_init_prev[:, :, 0] == 0
        inds &= warp_img_init_prev[:, :, 1] == 0
        inds &= warp_img_init_prev[:, :, 2] == 0
      #print('Step 45: Done')

      #black_pixels = np.where((warp_img_init_prev[:, :, 0] == 0) & (warp_img_init_prev[:, :, 1] == 0) & (warp_img_init_prev[:, :, 2] == 0))
      plt.clf()
      plt.imshow(warp_img_init_prev,cmap='gray')
      plt.show()
      plt.imshow(warp_img_init_curr,cmap='gray')
      plt.show()

      warp_img_init_prev[inds] = warp_img_init_curr[inds]
      #print('Step 46: Done')

      del inds,warp_img_init_curr



    print('Step31:Done')

    return warp_img_init_prev, H_trans
```

```
def final_steps_right_union_gpu(warp_img_init_prev,len_H_right,xmax,xmin,ymax,ymin,t,h,w,Ht,scale_factor=16,is_gray=True):

    for j in tqdm(range(len_H_right)):
      #print(j)
      f=h5.File('drive/MyDrive/H_right_sift_222.h5','r')
      H = f['data'][j]
      f.close()
      if j==0:
        H_trans = Ht@H
      else:
        H_trans = H_trans@H

      f=h5.File('drive/MyDrive/all_images_bgr_sift_443.h5','r')
      input_img_orig = f['data'][(len_H_right+1)+j+1]
      f.close()
      del f
      src = cv2.cuda_GpuMat()
      src.upload( np.uint8(input_img_orig))
      if scale_factor>1:
        dst = cv2.cuda.resize(src,None,fx=(1/scale_factor),fy = (1/scale_factor),interpolation = cv2.INTER_CUBIC)
      else:
```

```
      dst = src
    #input_img = dst.download()
    if is_gray==True:
      dst = cv2.cuda.cvtColor(dst, cv2.COLOR_BGR2GRAY)
    #print('input image accesssed')
    input_img = dst.download()
    #input_img = images_right[j+1]
    #result = np.zeros((ymax-ymin,xmax-xmin,3),dtype='uint8')

    src = cv2.cuda_GpuMat()
    src.upload( np.uint8(input_img))

    #dst = cv2.cuda_GpuMat()
    #dst.upload(result)

    #print('Step 42: Done')


    dst = cv2.cuda.warpPerspective(src, M = H_trans, dsize = (xmax-xmin, ymax-ymin) )



    #cv2.warpPerspective(src = np.uint8(input_img), M = H_trans, dsize = (xmax-xmin, ymax-ymin),dst=result)
    del input_img
    result = dst.download()

    warp_img_init_curr = result

    del result
    #print('Step 44: Done')

    if is_gray==True:
      inds = warp_img_init_prev[:, :] == 0
    else:
      inds = warp_img_init_prev[:, :, 0] == 0
      inds &= warp_img_init_prev[:, :, 1] == 0
      inds &= warp_img_init_prev[:, :, 2] == 0
    #print('Step 45: Done')


    warp_img_init_prev[inds] = warp_img_init_curr[inds]
    #print('Step 46: Done')

    del inds,warp_img_init_curr

  return warp_img_init_prev
```

```
from tqdm import tqdm
tqdm = partial(tqdm, position=0, leave=True)
```

```
xmax,xmin,ymax,ymin,t,h,w,Ht = warpnImages(10,10,scale_factor=1,offset=00)
```

```
    Step1:Done
    Step2:Done
```

```
print(xmax-xmin, ymax-ymin)
```

```
    5557 7060
```

```
print(xmax-xmin, ymax-ymin)
```

```
    5557 7060
```

```
warp_imgs_left,H_trans = final_steps_left_union_gpu(10,xmax,xmin,ymax,ymin,t,h,w,Ht,scale_factor=1,is_gray=True,offset=0)
```

```
    100%|██████████| 10/10 [00:01<00:00,  9.21it/s]Step31:Done
```

```
fig,ax =plt.subplots()
fig.set_size_inches(20,20)
ax.imshow(warp_imgs_left,cmap='gray')
ax.set_title('300-Images Mosaic-SIFT-Modified2')
```

```
print(H_trans)
```

```
    [[ 1.65387354e+00 -1.79429296e-01  2.83591525e+03]
     [ 3.33226951e-01  1.46379734e+00  4.52182521e+03]
     [ 7.59172052e-05 -1.26960152e-04  1.10845051e+00]]
```

```
xmax,xmin,ymax,ymin,t,h,w,Ht = warpnImages(20,20,scale_factor=1,offset=11)
```

```
    Step1:Done
    Step2:Done
```

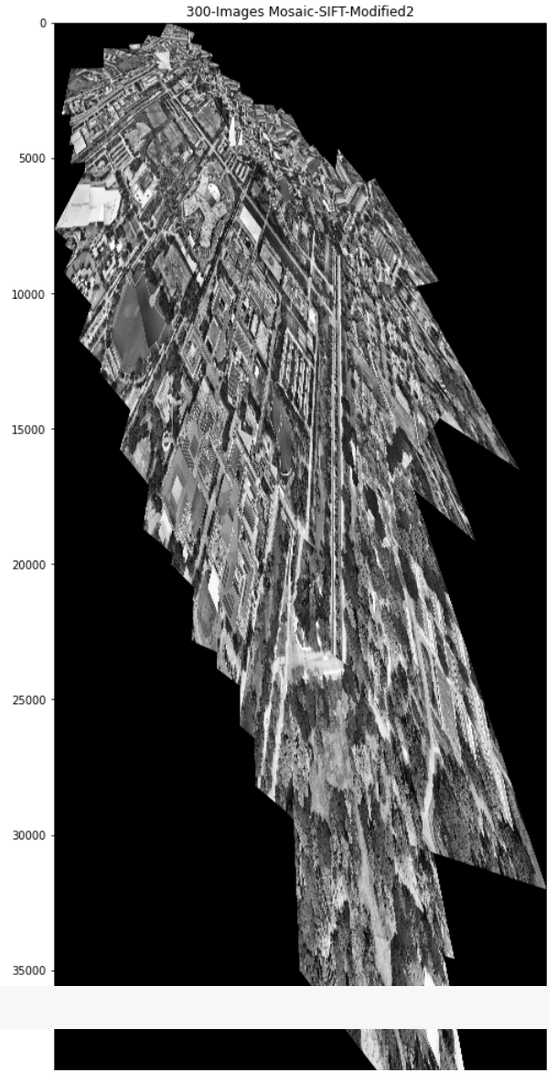
```
warp_imgs_left2,H_trans2 = final_steps_left_union_gpu(20,xmax,xmin,ymax,ymin,t,h,w,Ht,warp_imgs_left,scale_factor=1,is_gray=True,offset=11,H_trans=H_trans)
```

```
warp_imgs_all = final_steps_right_union_gpu(warp_imgs_left,10,xmax,xmin,ymax,ymin,t,h,w,Ht,scale_factor=1,is_gray=True)
```

```
    100%|██████████| 100/100 [03:30<00:00,  2.10s/it]
```

```
fig,ax =plt.subplots()
fig.set_size_inches(20,20)
ax.imshow(warp_imgs_all,cmap='gray')
ax.set_title('300-Images Mosaic-SIFT-Modified2')
```

```
Text(0.5, 1.0, '300-Images Mosaic-SIFT-Modified2')
```



1s completed at 4:43 PM