UAS Pemrosesan Paralel

Image Stitching

Disusun Oleh :

Kelompok 3

Mutiah Andini	**09011182126027**
Zahra Hanifa	**09011182126025**
Keisyah Sabinatullah Qur'aini	**09011182126011**
Indah Gala Putri	**09011182126033**

Dosen Pengampu : Adi Hermansyah, M.T.

Jurusan Sistem Komputer

Fakultas Ilmu Komputer

Universitas Sriwijaya

2023

1. Import Library

In [27]:
```
import numpy as np
import pandas as pd

import os
for dirname, _, filenames in os.walk('C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image1.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image10.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image11.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image12.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image13.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image14.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image15.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image16.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image17.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image18.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image19.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image2.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image20.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image3.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image4.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image5.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image6.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image7.jpg
C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom\image8.jpg
```

In [28]:
```
!pip install opencv-contrib-python
```

```
Requirement already satisfied: opencv-contrib-python in c:\users\62812\anaconda3\envs\mutia\lib\site-packages (4.8.1.78)
Requirement already satisfied: numpy>=1.17.0 in c:\users\62812\anaconda3\envs\mutia\lib\site-packages (from opencv-contrib-pyth
on) (1.24.3)
```

In [29]:
```
pip install imutils
```

```
Requirement already satisfied: imutils in c:\users\62812\anaconda3\envs\mutia\lib\site-packages (0.5.4)
Note: you may need to restart the kernel to use updated packages.
```

In [30]:
```
from imutils import paths
import imutils
import cv2
import numpy as np
```

2. Pre-processing Image

In [31]:
```
class utils:
    def loadImages(path,resize):
        '''Load Images from path to array, @param path is the folder which containing images, @param resize is True
        if image is halved in size, otherwise is False'''
        image_path = list(paths.list_images(path))
        list_image = []
        for i,j in enumerate(image_path):
            image = cv2.imread(j)
            if resize==1:
                image=cv2.resize(image,(int(image.shape[1]/4),int(image.shape[0]/4)))
            list_image.append(image)
        return (list_image)

    def trim(frame):
        '''crop frame '''
        #crop top
        if not np.sum(frame[0]):
            return trim(frame[1:])
        #crop bottom
        elif not np.sum(frame[-1]):
            return trim(frame[:-2])
        #crop left
        elif not np.sum(frame[:,0]):
            return trim(frame[:,1:])
        #crop right
        elif not np.sum(frame[:,-1]):
            return trim(frame[:,:-2])
        return frame

    def padding(img,top,bottom,left,right):
        '''add padding to img'''
        border = cv2.copyMakeBorder(
            img,
            top=top,
            bottom=bottom,
            left=left,
            right=right,
            borderType=cv2.BORDER_CONSTANT
        )
        return border
```

Class utils merupakan sebuah kelas utilitas yang menyediakan beberapa metode untuk operasi umum pada gambar. Pertama, metode loadImages(path, resize) digunakan untuk memuat gambar dari suatu direktori ke dalam sebuah array. Parameter resize memungkinkan untuk mengubah ukuran gambar menjadi setengah dari ukuran aslinya jika diinginkan. Metode kedua, trim(frame), berfungsi untuk memotong (crop) gambar dengan

menghilangkan bagian kosong di sekitarnya. Sedangkan metode ketiga, padding(img, top, bottom, left, right), ditujukan untuk menambahkan padding ke gambar dengan menyediakan kontrol atas jumlah padding yang ditambahkan pada setiap sisi. Contoh penggunaan metode-metode ini termasuk dalam proses pengolahan gambar yang umum dilakukan dalam analisis visual.

3. Feature Extraction

In [33]:

```
class stitch:
    def blendingMask(height, width, barrier, smoothing_window, left_biased=True):
        assert barrier < width
        mask = np.zeros((height, width))

        offset = int(smoothing_window/2)
        try:
            if left_biased:
                mask[:,barrier-offset:barrier+offset+1]=np.tile(np.linspace(1,0,2*offset+1).T, (height, 1))
                mask[:,:barrier-offset] = 1
            else:
                mask[:,barrier-offset:barrier+offset+1]=np.tile(np.linspace(0,1,2*offset+1).T, (height, 1))
                mask[:,barrier+offset:] = 1
        except:
            if left_biased:
                mask[:,barrier-offset:barrier+offset+1]=np.tile(np.linspace(1,0,2*offset).T, (height, 1))
                mask[:,:barrier-offset] = 1
            else:
                mask[:,barrier-offset:barrier+offset+1]=np.tile(np.linspace(0,1,2*offset).T, (height, 1))
                mask[:,barrier+offset:] = 1

        return cv2.merge([mask, mask, mask])

    def panoramaBlending(dst_img_rz,src_img_warped,width_dst,side,showstep=False):
        '''Given two aligned images @dst_img and @src_img_warped, and the @width_dst is width of dst_img
        before resize, that indicates where there is the discontinuity between the images,
        this function produce a smoothed transient in the overlapping.
        @smoothing_window is a parameter that determines the width of the transient
        left_biased is a flag that determines whether it is masked the left image,
        or the right one'''

        h,w,_=dst_img_rz.shape
        smoothing_window=int(width_dst/8)
        barrier = width_dst -int(smoothing_window/2)
        mask1 = stitch.blendingMask(h, w, barrier, smoothing_window = smoothing_window, left_biased = True)
        mask2 = stitch.blendingMask(h, w, barrier, smoothing_window = smoothing_window, left_biased = False)

        if showstep:
            nonblend=src_img_warped+dst_img_rz
        else:
            nonblend=None
            leftside=None
```

```
    def warpTwoImages(src_img, dst_img,showstep=False):

        #generate Homography matrix
        H,_=features.generateHomography(src_img,dst_img)

        #get height and width of two images
        height_src,width_src = src_img.shape[:2]
        height_dst,width_dst = dst_img.shape[:2]

        #extract conners of two images: top-left, bottom-left, bottom-right, top-right
        pts1 = np.float32([[0,0],[0,height_src],[width_src,height_src],[width_src,0]]).reshape(-1,1,2)
        pts2 = np.float32([[0,0],[0,height_dst],[width_dst,height_dst],[width_dst,0]]).reshape(-1,1,2)
```

```
    def multiStitching(list_images):
        n=int(len(list_images)/2+0.5)
        left=list_images[:n]
        right=list_images[n-1:]
        right.reverse()
        while len(left)>1:
            dst_img=left.pop()
            src_img=left.pop()
            left_pano,_,_,_=stitch.warpTwoImages(src_img,dst_img)
            left_pano=left_pano.astype('uint8')
            left.append(left_pano)

        while len(right)>1:
            dst_img=right.pop()
            src_img=right.pop()
            right_pano,_,_,_=stitch.warpTwoImages(src_img,dst_img)
            right_pano=right_pano.astype('uint8')
            right.append(right_pano)

        #if width_right_pano > width_left_pano, Select right_pano as destination. Otherwise is left_pano
        if(right_pano.shape[1]>=left_pano.shape[1]):
            fullpano,_,_,_=stitch.warpTwoImages(left_pano,right_pano)
        else:
            fullpano,_,_,_=stitch.warpTwoImages(right_pano,left_pano)
        return fullpano

    def crop(panorama,h_dst,conners):
        #find max min of x,y coordinate
        [xmin, ymin] = np.int32(conners.min(axis=0).ravel() - 0.5)
        [xmax, ymax] = np.int32(conners.max(axis=0).ravel() + 0.5)
        t = [-xmin,-ymin]
        conners=conners.astype(int)

        #conners[0][0][0] is the X coordinate of top-left point of warped image
        #If it has value<0, warp image is merged to the left side of destination image
        #otherwise is merged to the right side of destination image
        if conners[0][0][0]<0:
            n=abs(-conners[1][0][0]+conners[0][0][0])
            panorama=panorama[t[1]:h_dst+t[1],n:,:]
        else:
            if(conners[2][0][0]<conners[3][0][0]):
                panorama=panorama[t[1]:h_dst+t[1],0:conners[2][0][0],:]
```

Class stitch merupakan kumpulan metode yang berkaitan dengan proses penggabungan (stitching) beberapa gambar menjadi satu panorama. Pertama, terdapat metode blendingMask(height, width, barrier, smoothing_window, left_biased=True) yang menghasilkan masker untuk blending panorama. Selanjutnya, metode panoramaBlending(dst_img_rz, src_img_warped, width_dst, side, showstep=False) melakukan blending dua gambar yang telah di-align untuk menciptakan transisi yang halus di daerah tumpang tindih. Terdapat juga metode warpTwoImages(src_img, dst_img, showstep=False) yang menghasilkan gambar yang sudah di-warped berdasarkan matriks homografi.

Kemudian, metode multiStitching(list_images) membagi daftar gambar menjadi dua bagian (kiri dan kanan) dan menggabungkan setiap bagian secara terpisah sebelum menyatukannya menjadi satu panorama penuh. Metode crop(panorama, h_dst, conners) digunakan untuk memotong panorama berdasarkan gambar destinasi.

Kelas ini menyediakan fungsi-fungsi esensial untuk proses penggabungan gambar menjadi panorama, termasuk proses warping, blending, dan penyusunan panorama multi-gambar.

4. Load Image

In [38]:
```
image_1 = mpimg.imread("C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom/image1.jpg")
image_2 = mpimg.imread("C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom/image2.jpg")
image_3 = mpimg.imread("C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom/image3.jpg")
image_4 = mpimg.imread("C:/Users/62812/Documents/Pempar_UAS/Pempar_UAS/Fasilkom/image4.jpg")
```

In [90]:
```
list_images= [image_1,image_2,image_3,image_4]
```

5. Feature Detection

In [91]:
```
# Ambil dua gambar dari list_images
img0 = list_images[2]
img1 = list_images[3]
```

In [42]:
```
sift = cv2.xfeatures2d.SIFT_create()
```

In [43]:
```
k0, d0 = sift.detectAndCompute(img0, None)
k1, d1 = sift.detectAndCompute(img1, None)
```

Dua baris kode ini bertujuan untuk menemukan dan mendeskripsikan fitur pada dua gambar yang diberikan menggunakan metode SIFT (Scale-Invariant Feature Transform). SIFT adalah algoritma deteksi dan deskripsi fitur yang dapat diandalkan terhadap perubahan skala dan rotasi dalam gambar.

6. Feature Mathcing

In [44]:
```
# Cocokkan fitur antara kedua gambar
bf = cv2.BFMatcher()
matches = bf.knnMatch(d0, d1, k=2)
```

In [45]:
```
# Terapkan ratio test untuk memilih good matches
good_matches = []
for m, n in matches:
    if m.distance < 0.75 * n.distance:
        good_matches.append(m)
```

In [46]:
```
# Ambil koordinat dari good matches
src_pts = np.float32([k0[m.queryIdx].pt for m in good_matches]).reshape(-1, 1, 2)
dst_pts = np.float32([k1[m.trainIdx].pt for m in good_matches]).reshape(-1, 1, 2)
```

7. Estimasi Homografi

In [47]:
```
# Temukan matriks transformasi homografi
H, _ = cv2.findHomography(src_pts, dst_pts, cv2.RANSAC, 5.0)
```

8. Transformasi persfektif

In [48]:
```
# Warp gambar pertama ke dalam koordinat gambar kedua
result_img = cv2.warpPerspective(img0, H, (img1.shape[1] + img0.shape[1], img1.shape[0]))
```

9. Penyusunan Gambar Hasil

```
In [49]: # Gabungkan kedua gambar
         result_img[:, :img1.shape[1]] = img1
```

10. Visualisasi Keypoint

```
In [50]: # Visualisasi keypoints
         img0_kp = cv2.drawKeypoints(img0, k0, None)
         img1_kp = cv2.drawKeypoints(img1, k1, None)
         plt_img = np.concatenate((img0_kp, img1_kp), axis=1)
```

11. Visualisasi Hasil Image Stitching

```
In [51]: # Tampilkan hasil
         plt.figure(figsize=(15, 15))
         plt.imshow(plt_img)
         plt.show()
```


```
In [52]: mat=features.matchFeatures(f0,f1,ratio=0.85,opt='BF')
```

Pada baris ini, Anda menggunakan fungsi matchFeatures dari kelas features untuk mencocokkan fitur antara dua set fitur, yaitu f0 dan f1.

```
In [53]: H,matMask=features.generateHomography(list_images[2],list_images[3])
```

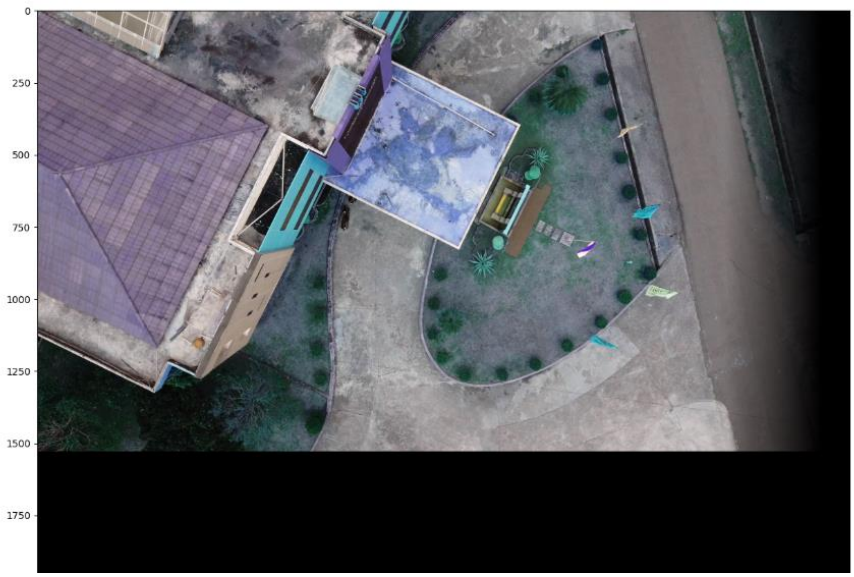
Baris kode tersebut menggunakan fungsi generateHomography dari kelas features untuk menghasilkan matriks homografi (H) dan mask (matMask).

```
In [54]: pano,non_blend,left_side,right_side=stitch.warpTwoImages(list_images[2],list_images[3],True)
```

baris kode ini menggunakan fungsi warpTwoImages dari kelas stitch untuk menggabungkan (warp) dua gambar, yaitu list_images[0] dan list_images[1]. Berikut adalah penjelasan variabel yang dihasilkan:

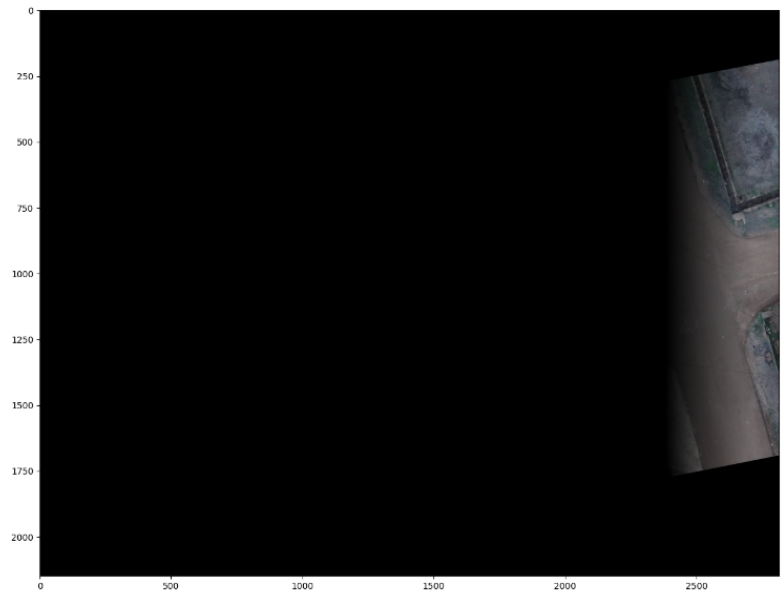
1. pano: Ini adalah gambar panorama akhir. Fungsi warpTwoImages melakukan warp pada salah satu gambar untuk menyelaraskan perspektif dengan menggunakan matriks homografi. Setelah itu, dilakukan blending dengan efek transisi di antara kedua gambar. Variabel pano menyimpan hasil akhir dari proses ini.
2. non_blend: Jika parameter showstep diatur sebagai True, ini akan berisi gambar yang merupakan hasil penjumlahan sederhana dari dua gambar sebelum efek transisi. Dengan kata lain, ini adalah gambar yang belum mengalami blending dan efek transisi.
3. left_side: Jika parameter showstep diatur sebagai True, variabel ini berisi gambar yang menunjukkan sisi kiri efek transisi. Sisi ini merupakan hasil dari warp pada gambar pertama sebelum blending dengan gambar kedua.
4. right_side: Jika parameter showstep diatur sebagai True, variabel ini berisi gambar yang menunjukkan sisi kanan efek transisi. Sisi ini merupakan hasil dari warp pada gambar kedua sebelum blending dengan gambar pertama.

```
In [55]: plt.figure(figsize=(15,15))
         plt.imshow(convertResult(left_side))
Out[55]: <matplotlib.image.AxesImage at 0x1af11fe2ca0>
```



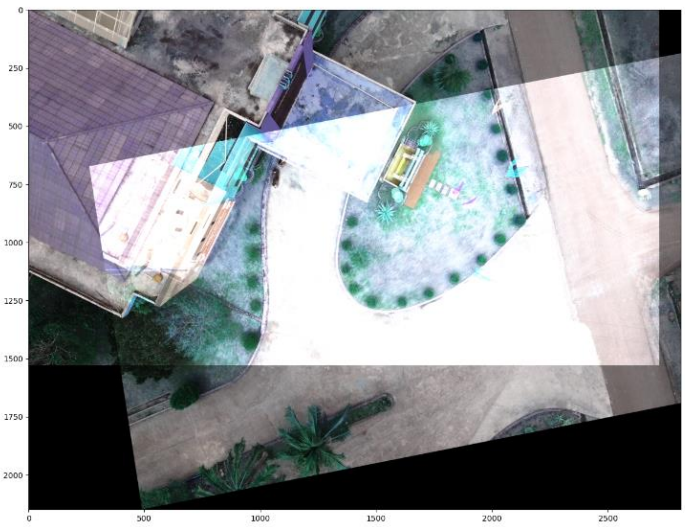
Baris kode ini menggunakan Matplotlib untuk membuat dan menampilkan gambar dari sisi kiri efek transisi (left_side).

```
In [56]: plt.figure(figsize=(15,15))
         plt.imshow(convertResult(right_side))
Out[56]: <matplotlib.image.AxesImage at 0x1af01dabd30>
```



Baris kode ini menggunakan Matplotlib untuk membuat dan menampilkan gambar dari sisi kanan efek transisi (right_side).

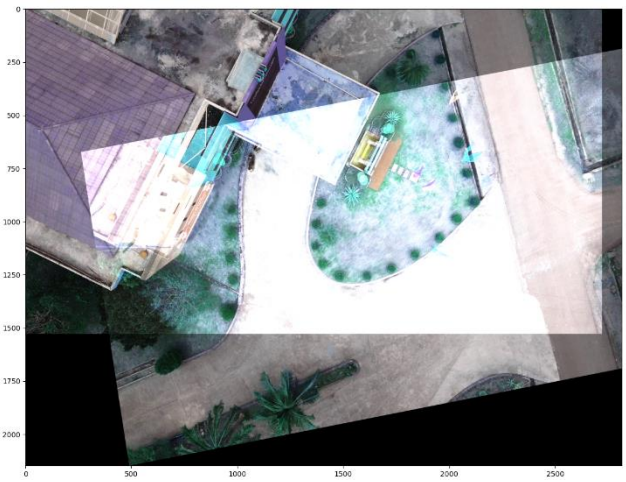
```
In [57]: plt.figure(figsize=(15,15))
         plt.imshow(convertResult(non_blend))

         Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

Out[57]: <matplotlib.image.AxesImage at 0x1af022ac490>
```



Baris kode ini menggunakan Matplotlib untuk membuat dan menampilkan gambar hasil penggabungan (blending) dua gambar, yaitu non_blend. Berikut penjelasan setiap bagian dari baris kode tersebut:

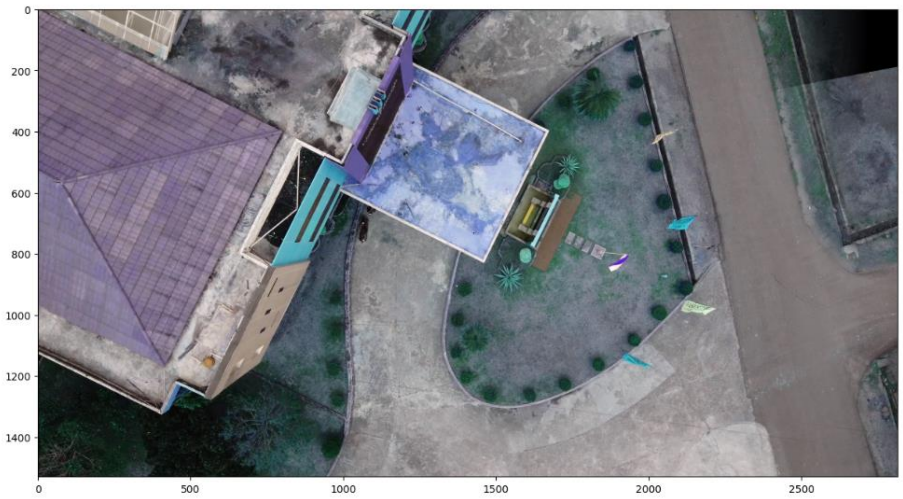
1. plt.figure(figsize=(15,15)): Membuat figur (gambar) dengan ukuran 15x15 inch. Fungsi plt.figure(figsize=(width, height)) digunakan untuk menentukan ukuran figur sebelum melakukan visualisasi.
2. plt.imshow(convertResult(non_blend)): Menampilkan gambar dengan menggunakan plt.imshow(). Fungsi ini digunakan untuk menampilkan representasi visual dari data gambar. Pada kasus ini, convertResult(non_blend) digunakan untuk mengonversi format gambar agar sesuai dengan tampilan yang diharapkan oleh Matplotlib (dari BGR ke RGB dan rentang nilai dari [0, 255] menjadi [0, 1]) sebelum ditampilkan.

```
In [58]: _,non_blend2,_,_=stitch.warpTwoImages(list_images[2],list_images[3],True)
         plt.figure(figsize=(15,15))
         plt.imshow(convertResult(non_blend2))

         Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

Out[58]: <matplotlib.image.AxesImage at 0x1af02324eb0>
```



Baris kode ini menggunakan fungsi warpTwoImages dari kelas stitch untuk menggabungkan (warp) dua gambar, yaitu list_images[0] dan list_images[1]. Selain itu, juga mengggunakan variabel _ (underscore) yang menunjukkan bahwa nilai tersebut tidak akan digunakan atau diambil.

```
In [78]: plt.figure(figsize=(15,15))
         plt.imshow(convertResult(pano))
Out[78]: <matplotlib.image.AxesImage at 0x1af02b10970>
```



Baris kode ini menggunakan Matplotlib untuk membuat figur (gambar) dengan ukuran 15x15 inci dan menampilkan citra panorama yang dihasilkan oleh proses stitching. Berikut adalah penjelasan untuk setiap bagian dari baris kode tersebut:

1. plt.figure(figsize=(15, 15)): Membuat figur (gambar) dengan ukuran 15x15 inci. Fungsi plt.figure(figsize=(width, height)) digunakan untuk menentukan ukuran figur sebelum melakukan visualisasi.
2. plt.imshow(convertResult(pano)): Menampilkan gambar dengan menggunakan plt.imshow(). Fungsi ini digunakan untuk menampilkan representasi visual dari data gambar. convertResult(pano) digunakan untuk mengonversi format gambar agar sesuai dengan tampilan yang diharapkan oleh Matplotlib (dari BGR ke RGB dan rentang nilai dari [0, 255] menjadi [0, 1]) sebelum ditampilkan.

Dengan menggunakan kedua fungsi ini, kami dapat melihat visualisasi dari citra panorama yang dihasilkan setelah proses stitching. Ukuran figur yang ditentukan memungkinkan untuk melihat citra panorama dengan jelas dan memberikan perspektif yang baik terhadap hasil akhir dari penggabungan beberapa gambar.

```
In [80]: #multi stitching
         panorama=stitch.multiStitching(list_images)
         plt.figure(figsize=(50,50))
         plt.imshow(convertResult(panorama))
Out[80]: <matplotlib.image.AxesImage at 0x1af02f4ca90>
```