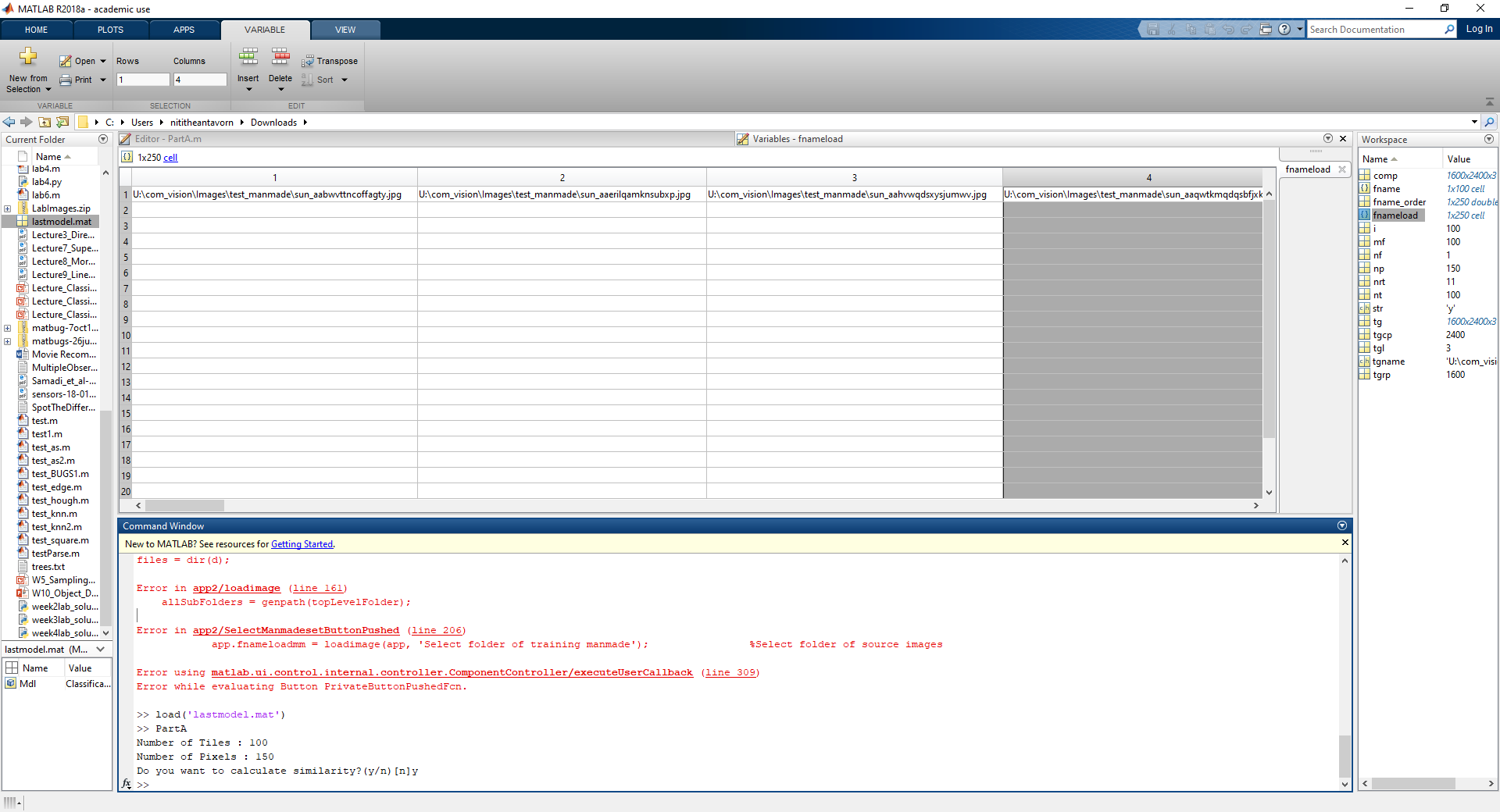
Part A

Load target image

Load folder of source images and keep file directory of all images in that folder on structure “fnameload”



Random the order of “fnameload”

Input number of tiles and pixels

Choose images from “fnameload” equal number of tiles

Calculate number tiles in each row for target composite image

**Composite Image**

There are two ways to composite image easy and hard.

**Easy method**

Put the source images in order as the table below.

**Number of tiles : 7**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 |

**Hard method**

Calculate the similarity of histogram for each tile and the crop target image by chi-square distance and shuffle source images with the high similarity in the same colour as the table below.

**Number of tiles: 7**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 2 | 3 | 7 | 5 | 6 | 1 | 1 | 2 | 4 |
| 6 | 3 | 7 | 6 | 5 | 6 | 3 | 4 | 2 | 1 |
| 7 | 7 | 2 | 3 | 1 | 5 | 4 | 6 | 1 | 3 |
| 5 | 4 | 6 | 2 | 7 | 5 | 4 | 3 | 7 | 2 |
| 6 | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Both method, the source images will be resize and crop in square equal the number of pixels by Gaussian filter and resize twice and crop after that. The composite of source images and the target image will be merge together with the weight “w = 0.3” which means that the target image has 70% and the composite of source images 30%.

Part B

The variables that use for K-nearest neighbour classifier are “R”, “G”, “B” and “linenum”. “R”, “G” and “B” are the mean value of percent of red, green and blue value for all pixels of image.

Red

|  |  |  |
| --- | --- | --- |
| 62 | 65 | 125 |
| 201 | 3 | 95 |
| 65 | 44 | 141 |

|  |  |  |
| --- | --- | --- |
| 62/  255 | 65/  255 | 125/  255 |
| 201/  255 | 3/  255 | 95/  255 |
| 65/  255 | 44/  255 | 141/  255 |

R = (62/255+65/255+125/255+201/255+3/255+95/255+65/255+44/255+141/255) / 9

R = 0.3490

“linennum” is the number lines that can detect by hough transform and filter unwanted line. First, the image will be convert to grayscale image and detect edges by Canny method. Canny, J. (1986, p. 679) mention that “The optimal detector has a simple approximate implementation in which edges are marked at maxima in gradient magnitude of a Gaussian-smoothed image”. The strength of this method may be the weak edges connected with the strong edges can be detected (MatLab Help). Second, hough transform will be applied for line detection. Hough transform…………………………………………………………………………………….

The parameters for detecting line will be adjusted to detect more line like this :

[H,T,R] = hough(BW,'RhoResolution',0.5,'Theta',-90:0.35:89);

P = houghpeaks(H,300,'Threshold',0.3\*max(H(:)),'NHoodSize',[3,3]);

With these parameters, abandon of lines can be detected including unwanted line