Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

EE-569 HOMEWORK # 4

Issue Date: 11/03/13 Due Date: 12/01/13

Problem #1: FACE WARPING

Motivation: In this problem, we will be applying facial warping technique learnt and implemented in the HW#3 to morph the images of faces of people.

Task: To morph the image of the face of one person into another using spatial warping and cross dissolving. We are required to morph the images of Drew Barrymore now and the childhood image of drew Barrymore. Also we are required to morph the face of Bruce banner into the Hulk.

1. A - BACK TO BABY

Code Written in: C++ / Matrix Operation and Equation solving and evaluating co-efficient using MATLAB.

Approach and Implementation: Before applying the procedure of facial morphing, we need to align the features of the two images to be morphed, such that we do not see 4 eyes/2 noses/2 mouths etc in the morphed images which is obviously not present in the face. Hence proper alignment of salient and unique features of the images must be done before we morph the face to another face.As described in the homework itself [1]. If the co-ordinates of selected feature point in image 1 is (xm,ym) and that of the image 2 is (xt,yt), then that particular feature point is spatially warped to the common midpoint ((xm+xt/2),(ym+yt/2)) before morphing.

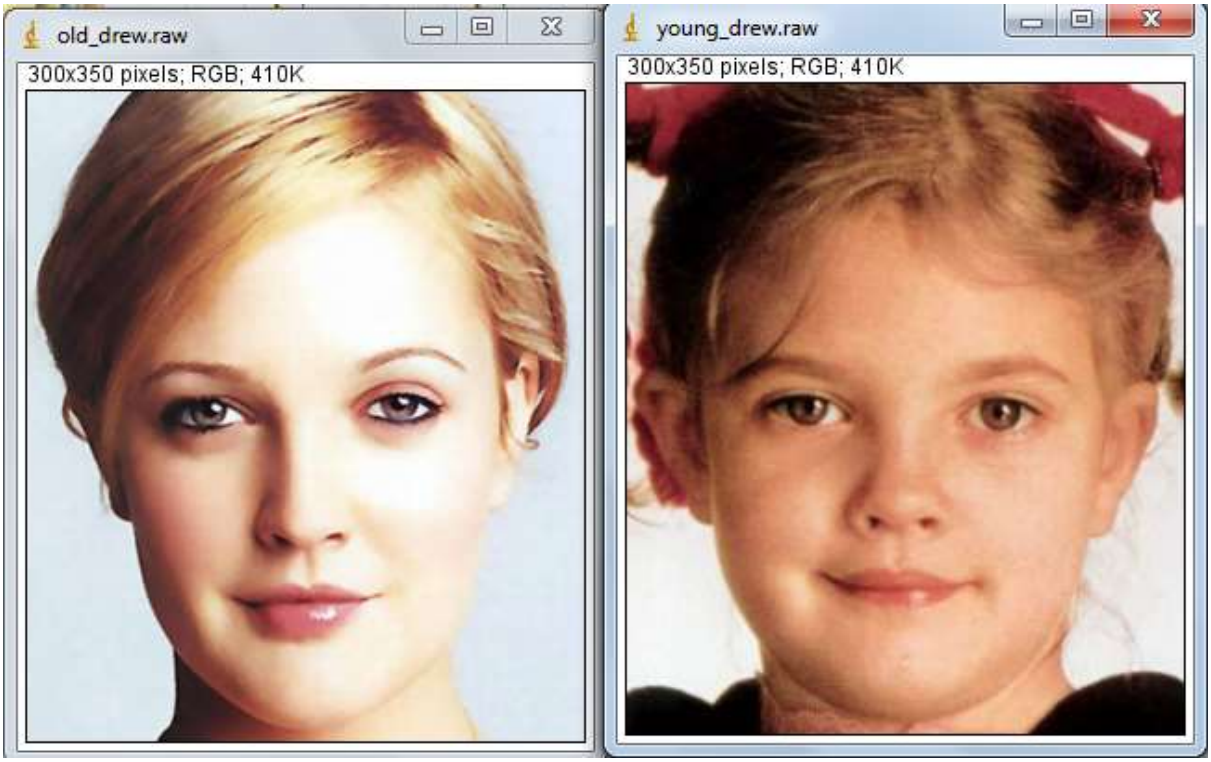


Figure 1.1: Drew Barrymore current image (Left) and young drew image (right).

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

Figure 1.2: Given Bruce Banner image (left) and Hulk Image (right).

List of control points selected for problem 1(a) – Warping Old and Young Drew. For both old drew and young drew images.

Control Points for midpoint warping of Drew Images

Face Feature	Old Drew Co-ordinates	Mid Point	Young Drew Co-ordinates
Left Eyeball Center	98,172	98,173	98,174
Right Eyeball Center	200,168	200,172	200,176
Eye Center Point (ECP)	148,170	147,172	146,174
Tip of Nose	150,246	148,240	146,234

Total # of Control Points = 4

Control Points for midpoint warping of Bruce Banner <-> Hulk Images

Face Feature	Bruce Banner Co-ordinates	Mid Point	Hulk Co-Ordinates.
Left Eyeball Center	134,164	131,168	128,172
Right Eyeball Center	252,160	260,166	268,172
Eye Center Point (ECP)	198,100	198,108	198,116
Tip of Nose	198,264	199,247	200,230

Total # of Control Points = 4

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

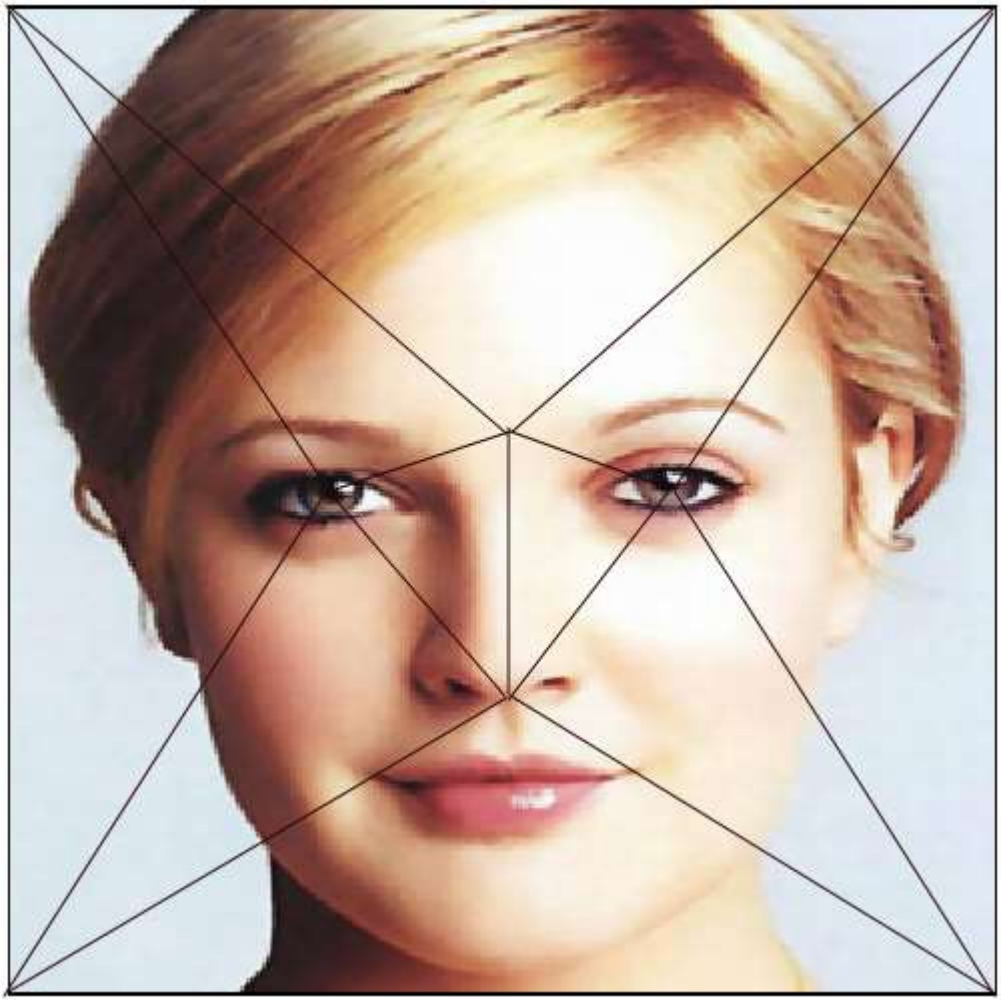


Figure 1.3. The control points and the triangles used to warp the Old Drew Image. Same was done for young drew image.

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

Figure 1.4. The control points and the triangles used to warp the Bruce Banner Image. Same was done for Hulk Image.

Spatial Warping:

Spatial Warping using triangles was done like in the warping in HW#3.

To avoid getting the fractional values of pixel co-ordinates in the output warped image, the reverse mapping was used. It was assumed that the output image pixel co-ordinates were only integral and the corresponding input image pixels location was found. If the pixel co-ordinate location turned out to be fractional, Bilinear Interpolation was used to approximate the mapping of input image pixels to warped image pixel intensity.

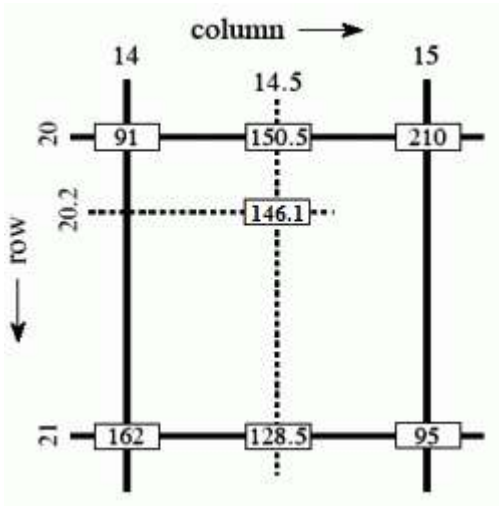


Figure 1.5 Bilinear Interpolation[3]:

For triangle method, barycentric co-ordinate system [2] were used to determine whether the pixel is within the triangle or outside it. Pixels with +ve eigen values are said to be within the triangle.

"*The eigen values of the any 3 arbitrary points (x1,y1),(x2,y2) and (x3,y3) that form a triangle are given as [2]* ;

$$\lambda_1 = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{\det(T)} = \frac{(y_2 - y_3)(x - x_3) + (x_3 - x_2)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)},$$

$$\lambda_2 = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{\det(T)} = \frac{(y_3 - y_1)(x - x_3) + (x_1 - x_3)(y - y_3)}{(y_2 - y_3)(x_1 - x_3) + (x_3 - x_2)(y_1 - y_3)},$$

$$\lambda_3 = 1 - \lambda_1 - \lambda_2 .$$

$$where\ ; \det(T) = (x1 - x3)(y2 - y3) - (y1 - y3)(x2 - x3)$$

If the eigen values are +ve then the point (x,y) is said to be inside the triangle with vertices (x1,y1),(x2,y2),(x3,y3).

If inside the then warp using co-efficients 'c0,c1,c2' and 'd1,d2 and d0' coefficients of the generalized linear transform matrix.

$$\begin{bmatrix} uq \\ vp \\ 1 \end{bmatrix} = \begin{bmatrix} C1 & C2 & C0 \\ D1 & D2 & D0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Xk \\ Yj \\ 1 \end{bmatrix}$$

Hence to convert the image to Cartesian co-ordinates the formula used was;

$$uq = q - 0.5\ ;\ and\ ;\ vp = 500 + 0.5 - p$$

Thus we have;

$$q = 0.5 + (C1 * Xk) + (C2 * Yj) + C0$$
$$p = 500 + 0.5 + (D1 * Xk) + (D2 * Yj) + D0$$

The same procedure was followed with the exact same feature points for warping the Bruce Banner and Hulk images to their midpoints.
The only thing different is the co-ordinates/location of the features in the Bruce and Hulk images.

Results and Discussion:

The following are the two images warped to the midpoints given in the table above.

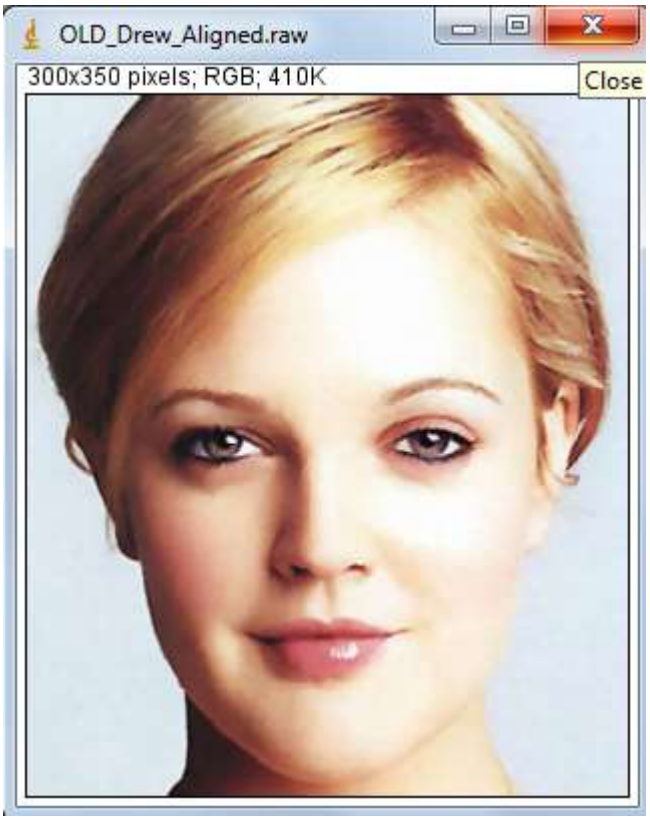


Figure 1.6: Old Drew Image Warped to Midpoint

Figure 1.6(b): Comparision between Old Drew and Old Drew Image Warped to Midpoint

Figure 1.7: Young Drew Image Warped to Midpoint

Since the differences in the original images itself is less, the warped to midpoint images need to be carefully observed for the correctness of the output. One such observation is the length of the nose.

Even though the homework problem 1(a) talks about only the ECP feature, I have tried 1(a) with more than 1 (ECP) feature point. This is because taking only 1 feature point will not produce a good output. Taking >1 feature point will produce a better output than taking only 1 feature point.

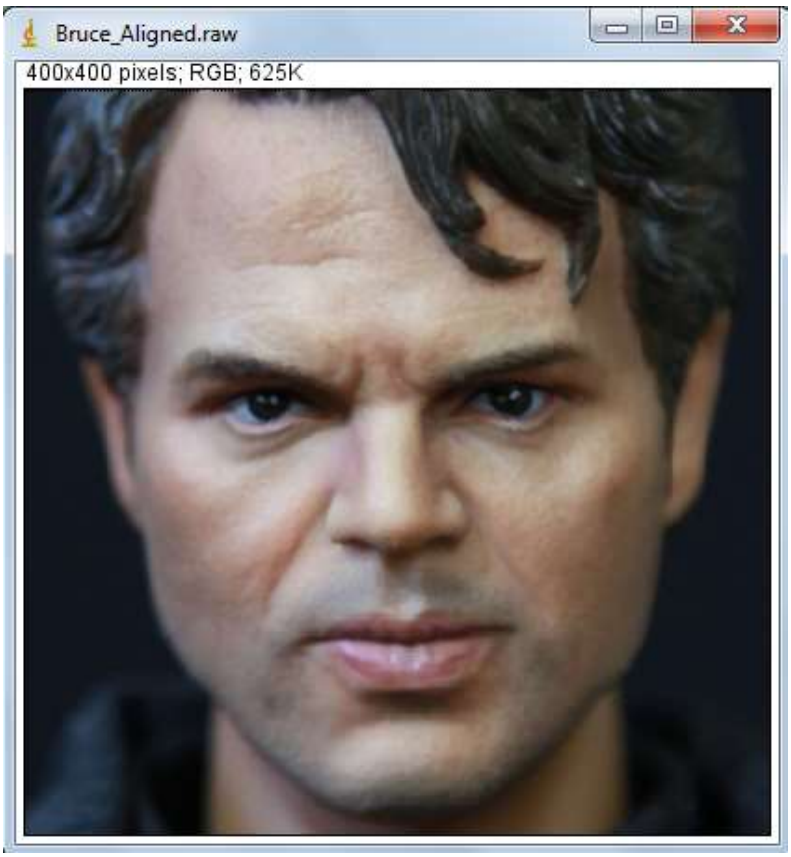


Figure 1.8: Bruce Banner Image Warped to midpoint.

Figure 1.9: Hulk Image warped to Midpoint.

In this case the warping is clearly visible.

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

I. B - FACE MORPHING

Task: To cross dissolve the two aligned images obtained from problem 1(a) to complete the face morphing procedure.

Code Written in: C++

Approach and Implementation: The following formula was used for cross-dissolving of the two images from initial to final image;

$$Output\ Image\ (i,j) = \ (1-\alpha)\ Initial\ Image(i,j) + \ \alpha\ Final\ Image(i,j)$$

In order to get a smooth transition from Initial Image to final Image, the value for the cross-dissolving parameter α was increased monotonically in the range [0, 1] in incremental steps.

The number of frames we require at the output determines the amount of incremental steps. Eg: The increment of 0.01 in value of α in [0, 1] will give us 100 output frames to later generate the video.

Results and Discussions:

The following results were obtained for Cross Dissolving from Old to Young Drew images which were aligned using the 4 control points discussed in Problem 1A;

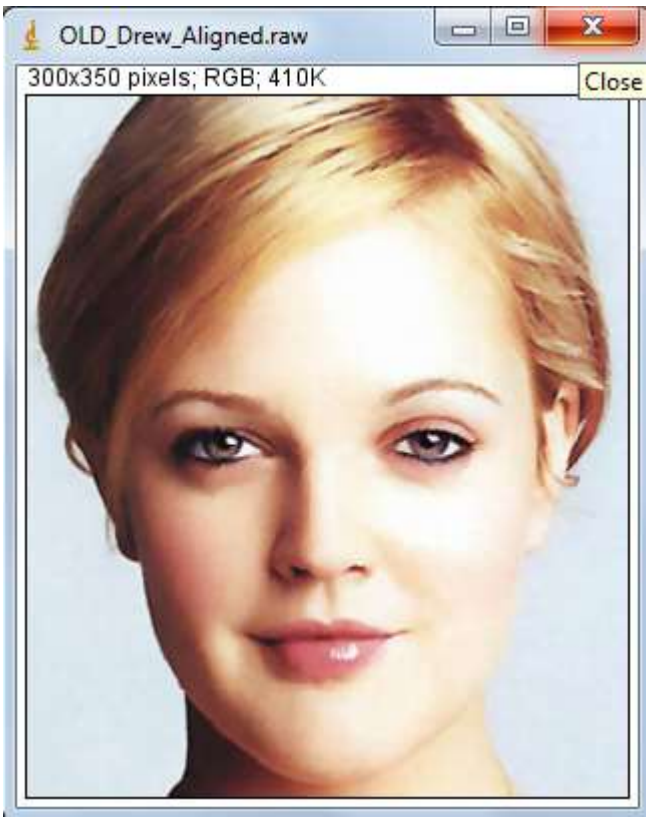


Figure 1.10: Old Drew aligned Image (Initial Image for Cross Dissolving).

Figure 1.11: Young Drew aligned Image (Final Image for Cross Dissolving)

Figure 1.12: Intermediate Image at α=0.2 for Cross Dissolving from Old to Young Drew image.

Figure 1.13: Intermediate Image at α=0.5 for Cross Dissolving from Old to Young Drew image.

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

Figure 1.14: Intermediate Image at α=0.8 for Cross Dissolving from Old to Young Drew image.

Figure 1.15: Some Intermediate Frames in the video from Old to Young Drew.

The following results were obtained for Cross Dissolving from Bruce Banner to Hulk images which were aligned using the 4 control points discussed in Problem 1A;

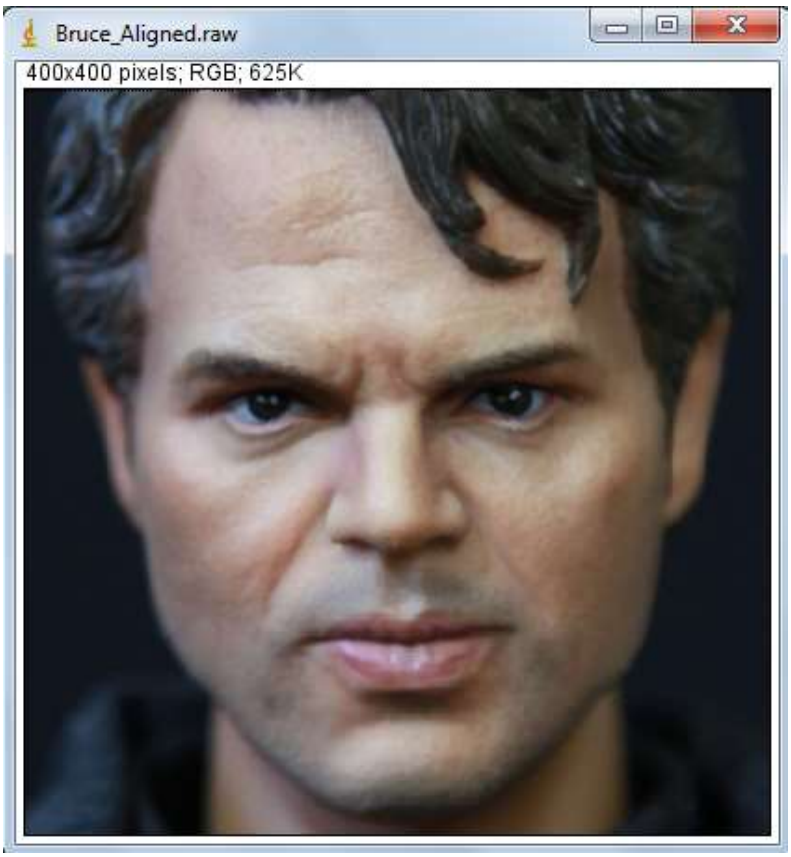


Figure 1.16: Bruce Banner aligned Image (Initial Image for Cross Dissolving).

Figure 1.17: Hulk aligned Image (Final Image for Cross Dissolving)

Figure 1.18: Intermediate Image at α=0.2 for Cross Dissolving from Bruce Banner to hulk image.

Figure 1.19: Intermediate Image at α=0.5 for Cross Dissolving from Bruce Banner to hulk image.

Figure 1.20: Intermediate Image at α=0.8 for Cross Dissolving from Bruce Banner to hulk image.

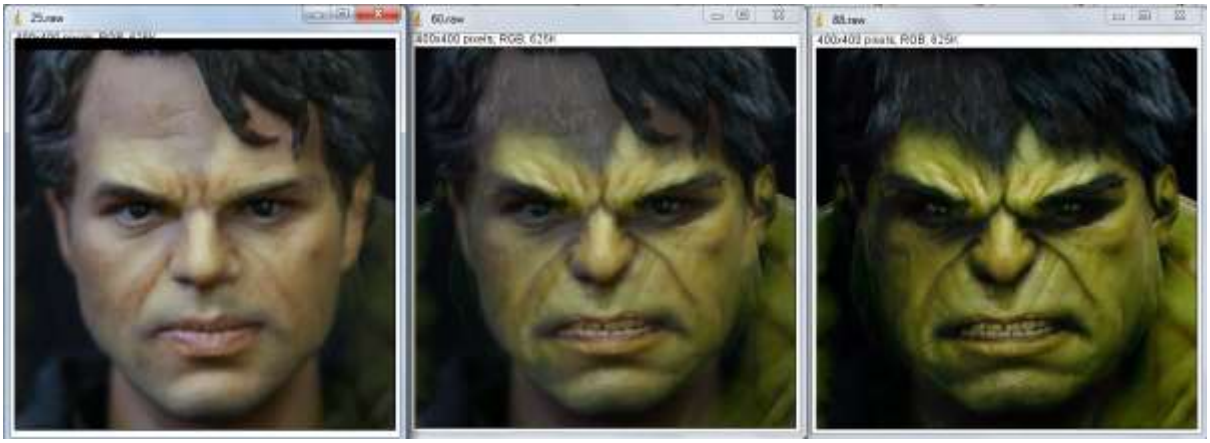


Figure 1.21: Some Intermediate Frames in the video from Bruce Banner to Hulk.

It can be easily observed, that although some of the image features like eyes are aligned near to ear other for both the sets of images. There is visible anomaly while cross dissolving. Most of the features that are visible to the naked eye are not aligned properly with one another and *thus we can easily see anomalies like 4 ears / 2 mouths / eyes not aligned properly / 4 eyebrows visible/ 4 cheeks and many more*. These anomalies are pointed out in the above figures for 1.20 and 1.21 both the sets of images. Thus it becomes necessary to improve the morphing process rather than just

choosing few feature points and applying cross-dissolving like what we did in Problem 1(a) and problem 1(b).

I. C - ADVANCED MORPHING

Task: To improve the results obtained by implementing face morphing as described in problem#19a0 and problem#1(b).

Code Written in: C++ / Matrix Operation and Equation solving and evaluating co-efficient using MATLAB.

Approach and Implementation: The proposed solution for improving the morphing from Old-Young Drew and Bruce Banner – Hulk involves two steps;

- **More feature points** aligned so that we do not get anomalies while cross dissolving. More feature points aligned means more triangles and there will be subtle variations in the aligned image from the original image as opposed to warping using less triangles of more area. More feature points means we can align more salient features of the face that are unique and the cross dissolving will be better as compared to that observed previously.

- **Simultaneous Warping and Morphing:** Rather than going from the aligned initial image -> final aligned image, we must go from initial image ->aligned initial image -> aligned final image -> final image. The more elegant way to this step is to simultaneously warp and morph the initial and final images. While going from Initial image to Final Image we must select more intermediate points to be aligned rather than going to just the middle point. In this implementation, I warped to $1/4^{th}$ common point, ½ common point and $3/4^{th}$ common point as opposed to just aligning to midpoint. This means that when the initial image is warped to $1/4^{th}$ amount, the final image is warped to $3/4^{th}$ amount. Midway through, the aligning is ½ as discussed in the problem 1(a). And when the initial point is warped to $3/4^{th}$ common point the final image is aligned to $1/4^{th}$ common point. Simultaneous morphing ensures that the image that been warped more is seen less during the cross dissolving operation otherwise there will be no "improvement" in the proposed method.

List of control points selected for problem 1(c) –Old and Young Drew. For both old drew and young drew images.

Control Points warping of Bruce Banner – Hulk Images

Feature Point	**Old Drew Co-Ordinates**	**Young Drew Co-Ordinates**
Left Ear Upper Tip	*30,144*	*24,124*
Left Ear Lower Tip	*56,224*	*32,208*
Left Eyebrow Upper End	*86,148*	*68,140*
Left Eyebrow Lower End	*156,152*	*152,168*
Left Eyeball Center	*134,166*	*130,172*
Eye Center Point (ECP)	*194,266*	*196,172*
Right Eyebrow Lower End	*222,152*	*242,168*
Right Eyebrow Upper End	*296,142*	*322,144*
Right Ear Upper Tip	*374,118*	*376,128*
Right Ear Lower Tip	*350,226*	*368,206*
Nose Tip	*200,256*	*200,224*
Left Cheek	*132,274*	*120,250*
Right Cheek	*262,174*	*288,250*
Top End of Lips	*204,296*	*202,284*
Left tip of Lips	*150,308*	*130,290*
Bottom End of Lips	*204,328*	*200,310*
Right Tip of Lips	*256,304*	*276,286*
Tip of Chin	*204,390*	*206,384*
Left side of Chin	*120,356*	*118,356*
Right side of Chin	*282,356*	*284,356*
Left Jaw	*82,308*	*58,308*
Right Jaw	*326,204*	*346,304*

Control Points warping of Old-Young Drew Images

Feature Point	**Old Drew Co-Ordinates**	**Young Drew Co-Ordinates**
Left Ear Upper Tip	*200,164*	*8,158*
Left Ear Lower Tip	*48,230*	*40,236*
Left Eyebrow Upper End	*66,156*	*56,162*
Left Eyebrow Lower End	*120,154*	*118,160*
Left Eyeball Center	*98,172*	*98,174*
Eye Center Point (ECP)	*150,150,*	*150,154*
Right Eyebrow Lower End	*178,152*	*170,160*
Right Eyebrow Upper End	*236,150*	*238,162*
Right Ear Upper Tip	*274,142*	*284,164*
Right Ear Lower Tip	*256,224*	*252,246*
Nose Tip	*150,244*	*148,236*
Left Nostril	*124,228*	*122,224*
Right Nostril	*172,228*	*174,224*
Top End of Lips	*150,266*	*148,262*
Left tip of Lips	*110,272,*	*106,264*
Bottom End of Lips	*150,292*	*148,282*
Right Tip of Lips	*188,272*	*188,270*
Tip of Chin	*150,344*	*150,340*
Left Jaw	*64,272*	*52,264*
Right Jaw	*236,272*	*244,264*

# of Control Points in both cases for improved result : 23 and Warping was done to quarter, mid-point and three-quarter points for simultaneous, warping and morphing.

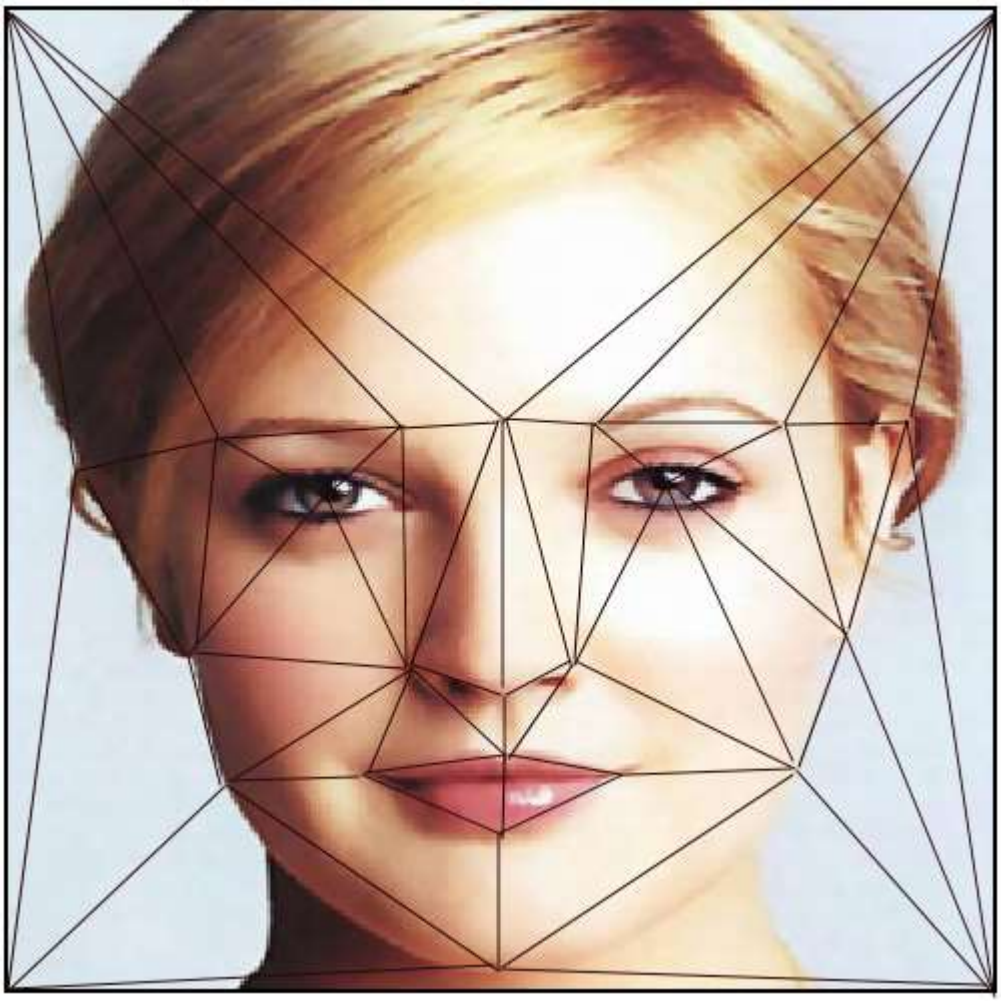


Figure 1.22: Control Points and triangles for Drew.

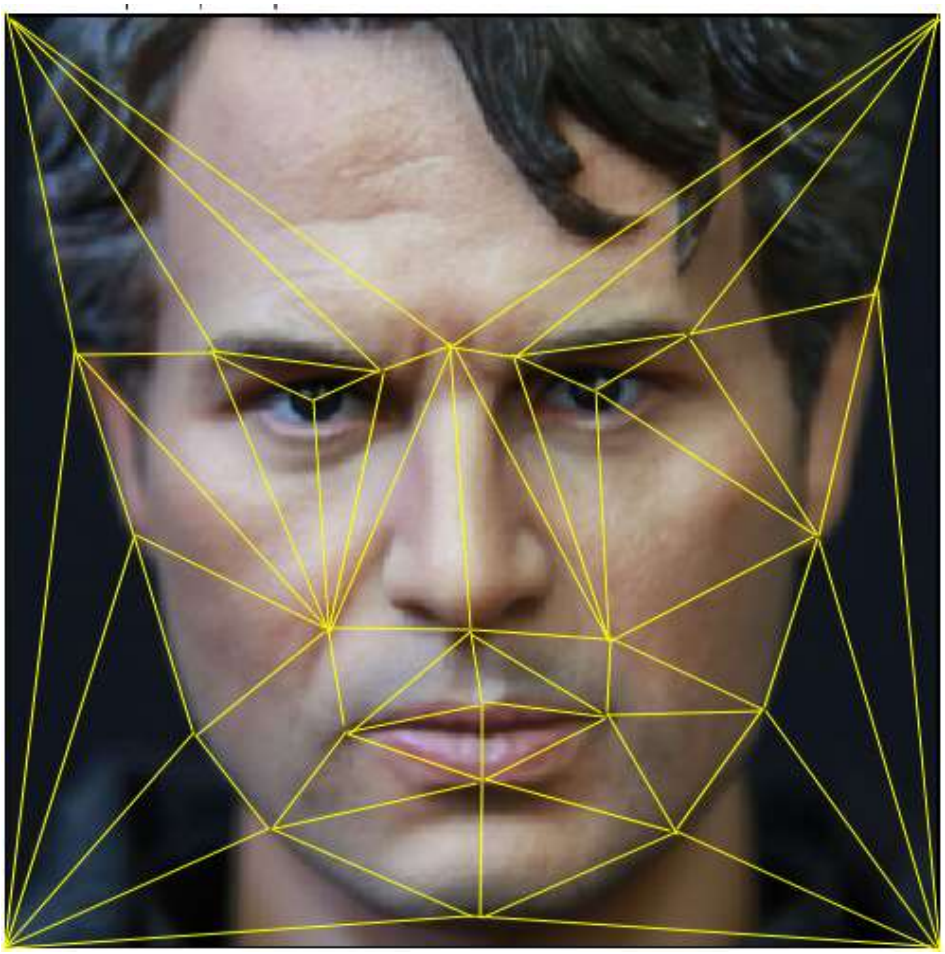


Figure 1.23: Control Points and triangles for Bruce Banner.

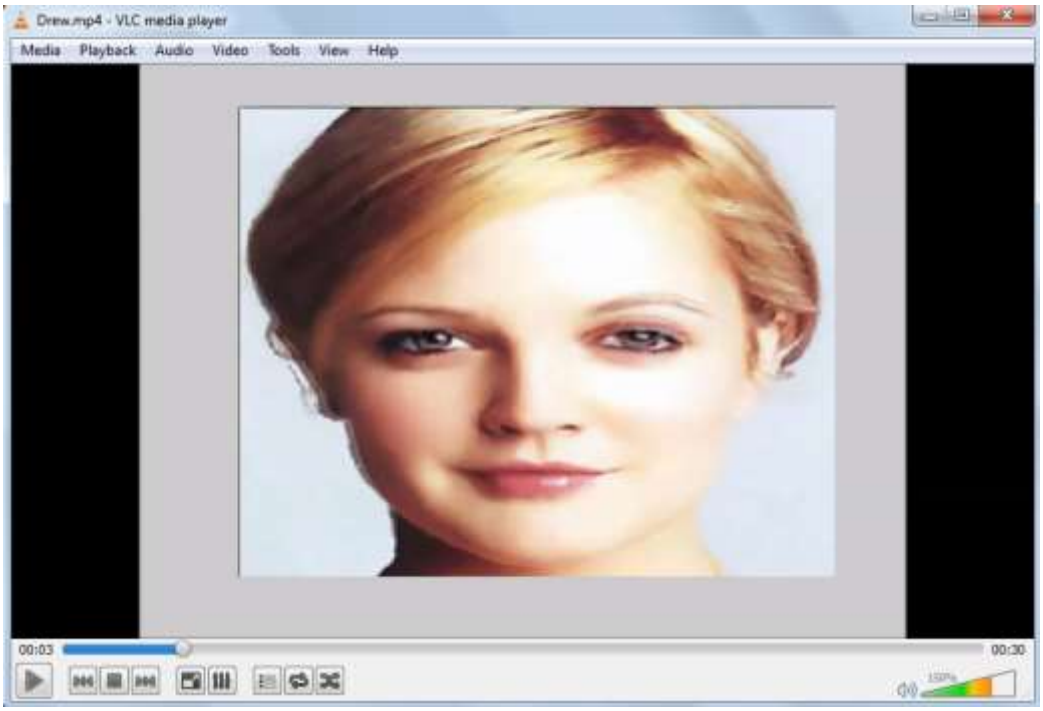


Figure 1.24: Intermediate Frames in the video from Old Drew to Young Drew.

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

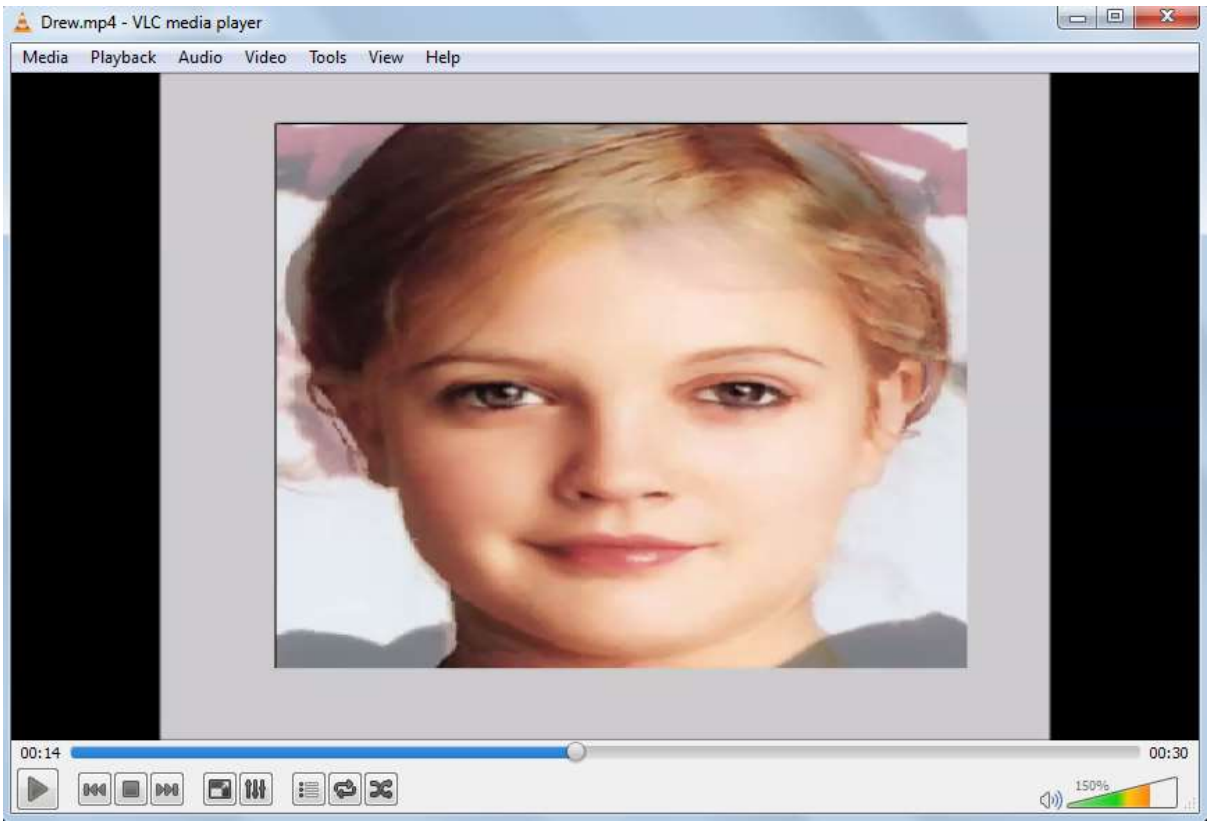


Figure 1.25: Intermediate Frames in the video from Old Drew to Young Drew.

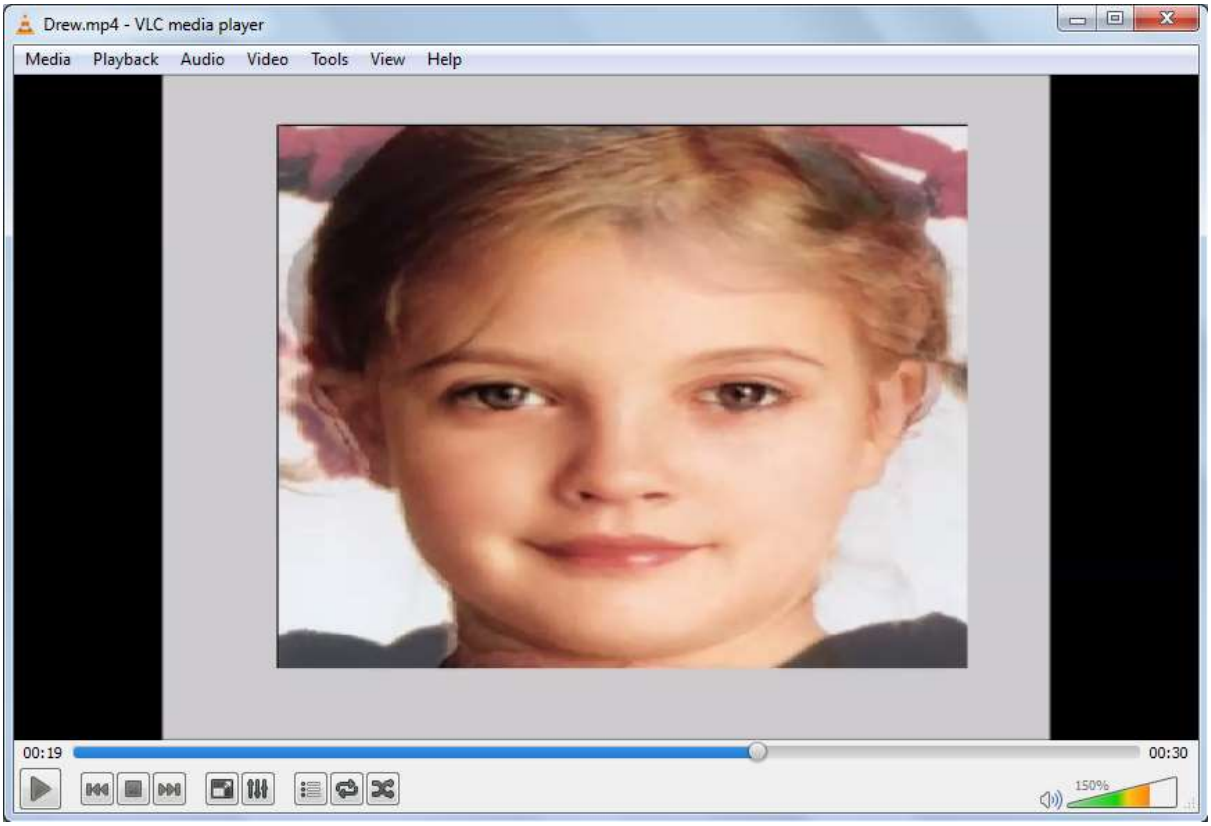


Figure 1.26: Intermediate Frames in the video from Old Drew to Young Drew.

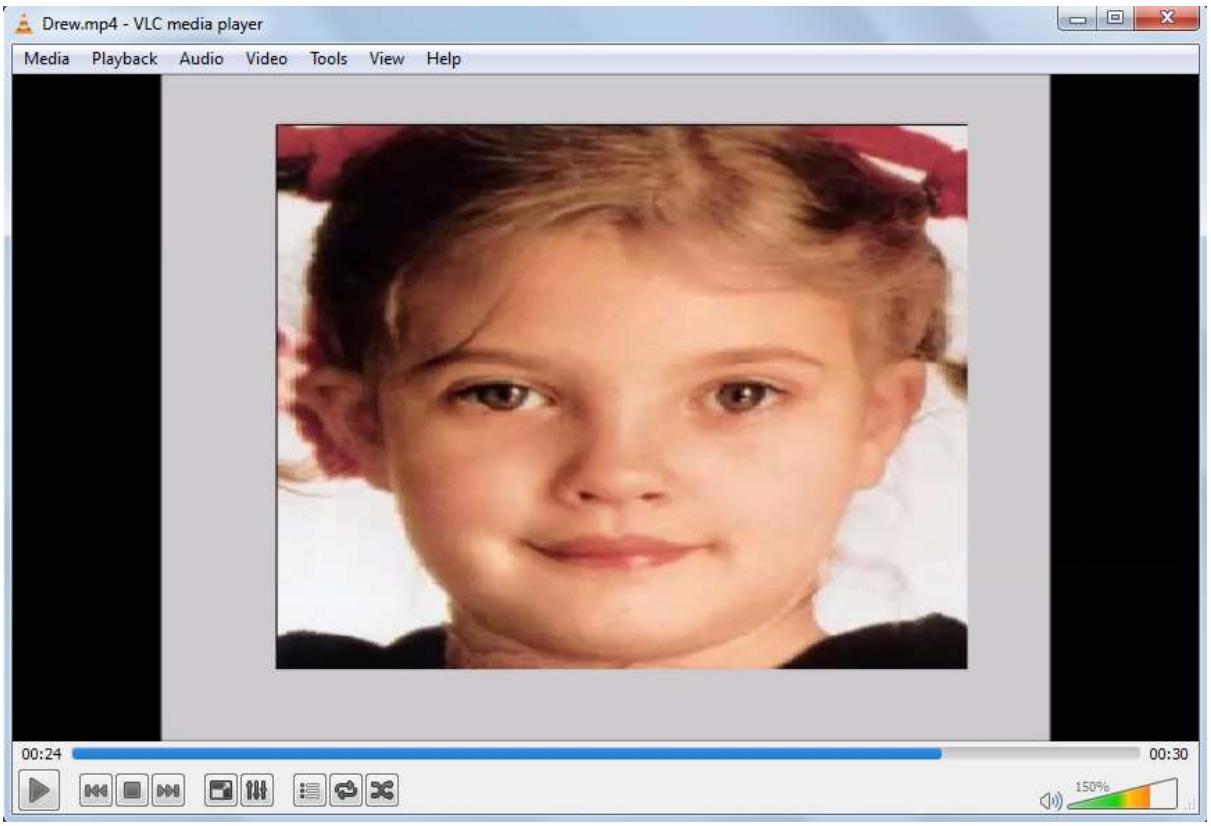


Figure 1.27: Intermediate Frames in the video from Old Drew to Young Drew.

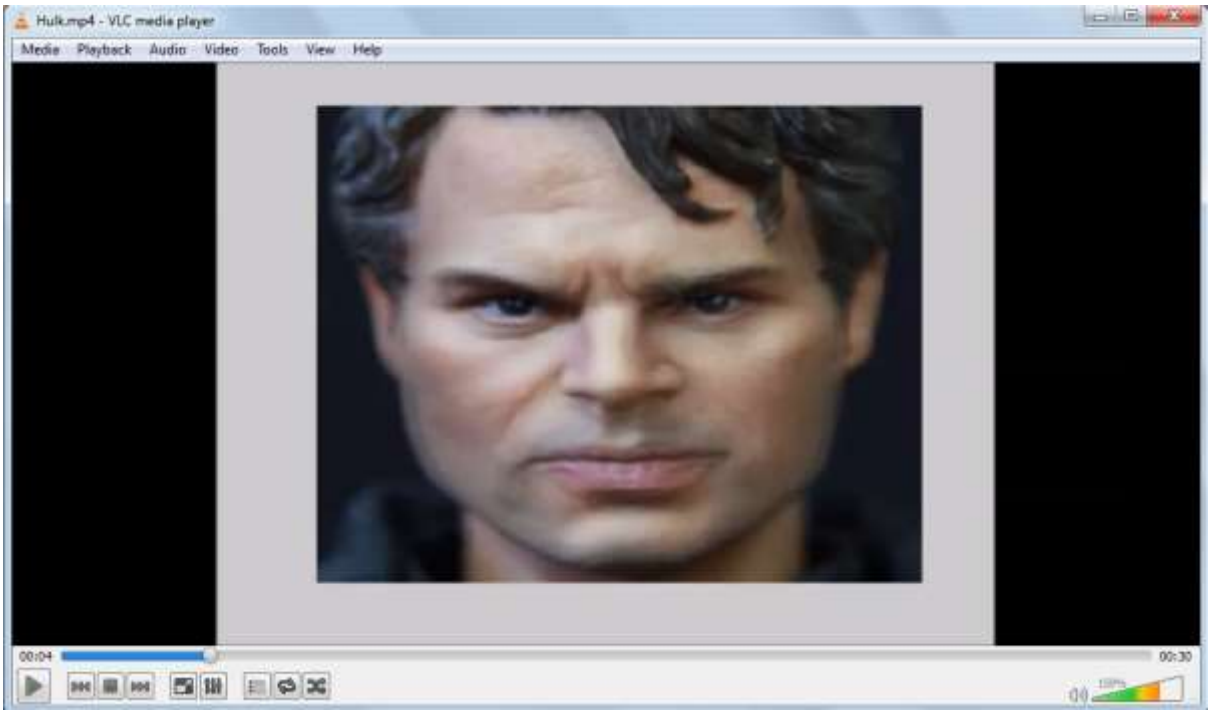


Figure 1.28: Intermediate Frames in the video from Bruce Banner to Hulk.

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

Figure 1.29: Intermediate Frames in the video from Bruce Banner to Hulk.

Figure 1.30: Intermediate Frames in the video from Bruce Banner to Hulk.

Figure 1.31: Intermediate Frames in the video from Bruce Banner to Hulk.

Thus we see that the unique salient features of the images are aligned with each other, and there are minimum anomalies while cross-dissolving the two images to get the morphing output.

Double features like 4 eyes, 2 mouths, 2 noses, 4 eyebrows etc.. are removed which were visible in previous case and the cross-dissolving and morphing from one face to another is also quite smooth as compared to the previous case.

Thus it can be concluded, that this method implemented is better than the one implemented in Problem 1(a) and problem 1(b).

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

Problem # 2: OPTICAL CHARACTER RECOGNITION (OCR)

Motivation: OCR is fast becoming of the most sought after and innovative tools in the field of image processing and computer vision. If it has to be done properly, then it is not just limited to only concept of image processing but encompasses all of the image processing concept clubbed together in one application and hence it becomes important to study it and implement it.

Code Written in: C++

2. A: OCR SEGMENTATION AND TRAINING.

Task: Automatically segment the training characters from the given image and extract properties of all the characters present in the training data. To make a decision tree based on the unique features extracted for each character.

Approach and Implementation:

- **Pre Processing:** This was done as the first step of the whole process. Firstly the RGB images were converted to grey and then to binary images Using appropriate threshold for each of the three images, these 3 images were binarized. If the images were noisy, an appropriate threshold for binarization was selected to remove the majority of the noise. If the noise still remained, then I used the isolated pixel removal concept in which I check for the 8 connected neighbours for a particular pixel. If all the eight connected neighbours have a different intensity level then the middle pixel/pixel of interest was changed to that around it. This all the noise was removed by this process.
- **Threshold for Binarization of Training image = 200**
- **Threshold for Binarization of TestBill1 image = 159**
- **Threshold for Binarization of TestBill2 image = 163**
- **Threshold for Binarization of Restraunt image = 172**

- **Automatic segmentation:** In order to individually segment out the characters from the training image, I used the **'connected component labelling' algorithm.** The following is the algorithm for connected component labelling;

"On the 1st pass:

- *Iterate through each element of the data by column, then by row (Raster Scanning).*
- *If the pixel value is not the background.*
- *Get the neighboring non-background label of the current pixel.*
- *If there are no such neighbor, uniquely label the current pixel and continue.*
- *Otherwise, find the neighbor with the smallest label, and assign that label to current pixel.*

- *Store the equivalence between neighboring labels.*

On the 2nd pass:

- *Iterate through each element of the data by column, then by row.*
- *If the pixel value is not the background: Re-label the element with the lowest equivalent label." [5]*

After, connected components were labelled; I assigned a particular grey level equal to that of the label for that connected component. Thus going forward, I can easily segment out the image using only that particular grey level connected component that I am interested in and perform further procedure. This was done for all the 29 training characters. Moreover, connected component labelling was also used in testing set, as we also need to segment the testing images in order to extract their features and then use the decision tree to compare and match the characters.

Problem with thinning: *Only few of the characters that have very similar properties like 2,5 and 6,9 and 0,o,O thinning was applied to distinguish the characters. Otherwise thinning created problems in segmentation because characters like 't' broke up into 2 components especially in testing images.*

- **Feature Extraction :**
  Once we have all the individual components segemented through connected component labelling, we now extract their features to make the decision tree.From the lectures, the bit Quads were used to extract many features. The following are the features that were extracted in my implementation. In all 19 features were extracted for each character during training and testing purposes.

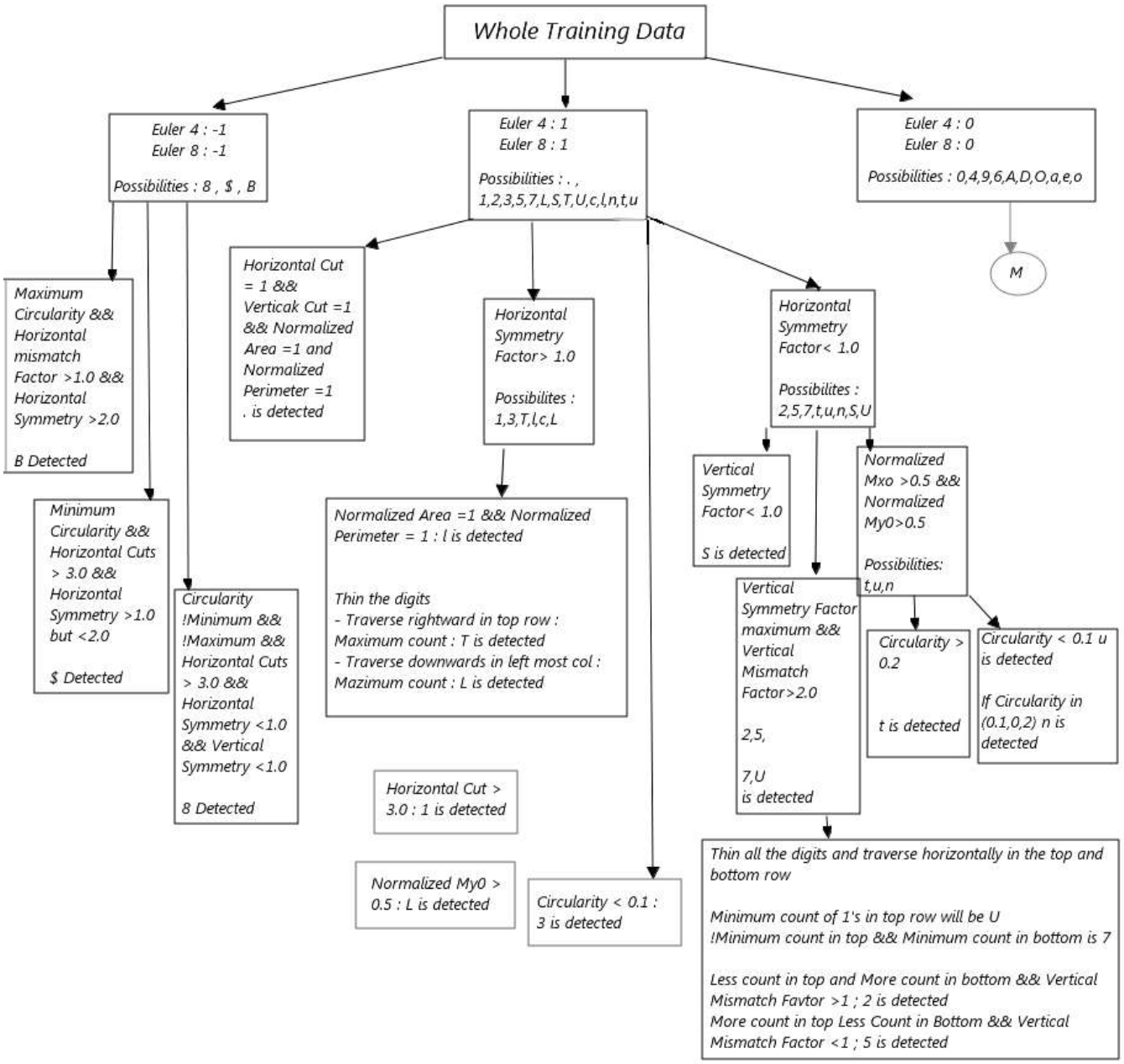
  - **Area :** Numbers of 1's present in the character : Implemented by the formula of Bit Quad counts given in lecture notes
  - **Perimeter :** Perimenter of the characted encapsulated by 1's : Implemented by the formula of Bit Quad counts given in lecture notes
  - **Euler # (4-connectivity) :** Euler number considering 4-connected pixel : Implemented by the formula of Bit Quad counts given in lecture notes
  - **Euler # (8-connectivity) :** Euler number considering 8-connected pixel Implemented by the formula of Bit Quad counts given in lecture notes
  - **Circularity :** 4*pi*Area/(Perimeter*Perimeter)
  - **Bounding Box Area :** Self Expalanatory
  - **Aspect Ratio :** Width:Height of Bounding Box
  - **Vertical Mismatch Factor :** No of 1s in the upper half of bounding box / Number of 1s in the lower half of Bounding Box
  - **Horizontal Mismatch Factor :** No of 1s in the left half of bounding box / Number of 1s in the right half ofBounding Box
  - **First Order Moment in X direction (Mx0) :** Self Expalanatory
  - **First Order Moment in Y direction (My0) :** Self Expalanatory
  - **Horizontal Cuts:** Number of 0 to 1 transitions in the horizontal direction
  - **Vertical Cuts :** Number of 0 to 1 transitions in the vertical direction

- **Normalized Area :** Area normalized by area of bounding box
- **Normalized Perimeter :** Perimeter normalized by Perimeter of bounding box
- **Normalized Mx0 :** Mx0 Normalized by the width of the bounding box
- **Normalized My0 :** My0 Normalized by the height of the bounding box
- **Vertical Symmetry:** Count of 1's in Upper half - Count of 1's in Lower half normalized by the height of the bounding box**.**
- **Horizontal Symmetry:** Count of 1's in Left half - Count of 1's in Right half normalized by the width of the boundingbox**.**

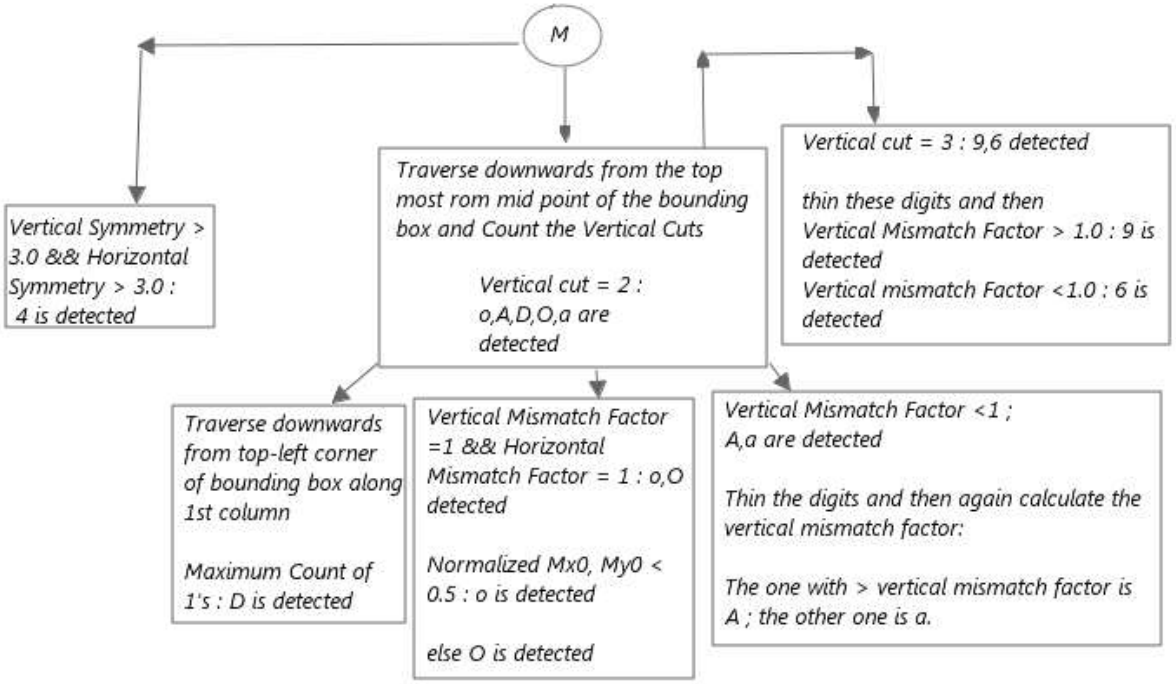
As far as possible, I tried to take the Normalized Features so that the size of the Character doest not affect classification.

- **Decision Tree: Based on the feature data obtained for the training data set of characters I built up the following decision tree.**

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu



Decision Tree Contd...


M
Vertical Symmetry > 3.0 && Horizontal Symmetry > 3.0 : 4 is detected
Traverse downwards from the top most rom mid point of the bounding box and Count the Vertical Cuts
Vertical cut = 2 : o,A,D,O,a are detected
Vertical cut = 3 : 9,6 detected
thin these digits and then Vertical Mismatch Factor > 1.0 : 9 is detected
Vertical mismatch Factor <1.0 : 6 is detected
Traverse downwards from top-left corner of bounding box along 1st column
Maximum Count of 1's : D is detected
Vertical Mismatch Factor =1 && Horizontal Mismatch Factor = 1 : o,O detected
Normalized Mx0, My0 < 0.5 : o is detected
else O is detected
Vertical Mismatch Factor <1 ; A,a are detected
Thin the digits and then again calculate the vertical mismatch factor:
The one with > vertical mismatch factor is A ; the other one is a.

Results and Discussion:

The outputs for the segmentation and connected component are given in the results and discussion section of problem 2(b).

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

The following csv file was generated as the consolidated features of all the characters within the training data;

	Area	Euler4	Euler8	Circularity	AspectRatio	BoundingArea	V-MismatchFactor	H-MismatchFactor	MxO	MyO	V-Cut	H-Cut	Normalized Area	Normalized Perimeter	MxO-Norm	MyO-Norm	Vertical Symmetry	Horizontal Symmetry
1	323	1	1	0.213135	0.369565	782	1.284722	0.146853	12.19195	20.90093	0.369565	3.058824	0.413043	1.095238	0.717174	0.454368	2.411765	5.304348
2	505	1	1	0.095337	0.652174	1380	0.774306	0.806228	15.45347	25.19604	1.630435	1.8	0.365942	1.697368	0.515116	0.54774	2.166667	1.217391
3	451	1	1	0.082563	0.638298	1410	1.063063	0.581633	16.65854	23.05543	1.468085	2.066667	0.319858	1.701299	0.555285	0.490541	0.466667	2.617021
4	547	0	0	0.152942	0.711111	1440	0.693939	0.516216	17.48812	23.43144	1	1.90625	0.379861	1.376623	0.546504	0.520699	3.15625	3.977778
5	524	1	1	0.093063	0.652174	1380	1.361233	0.925	14.87023	20.79008	1.565217	2.033333	0.37971	1.75	0.495674	0.451958	2.733333	0.456522
6	553	0	0	0.073255	0.659574	1457	1.069343	1.046931	14.45208	23.11573	1.553191	2.612903	0.379547	1.974359	0.466196	0.491824	0.612903	0.276596
7	365	1	1	0.135478	0.666667	1350	2.189655	1.010471	14.63288	14.44111	1.044444	1.5	0.27037	1.226667	0.487763	0.320913	4.6	0.044444
8	573	-1	-1	0.074928	0.638298	1410	1.023891	0.96	14.43456	23.16405	1.617021	2.633333	0.406383	2.012987	0.481152	0.492852	0.233333	0.255319
9	551	0	0	0.074923	0.638298	1410	0.941581	0.92517	14.94918	22.88929	1.510638	2.7	0.39078	1.974026	0.498306	0.487006	0.566667	0.468085
0	535	0	0	0.096462	0.787234	1739	0.99635	0.792079	21.49907	23.05794	1.021277	2.27027	0.307648	1.571429	0.581056	0.490595	0.027027	1.340426
.	36	1	1	0.7854	11.5	414	0.75	0	65.5	2.5	1	0.086957	0.086957	0.16	0.949275	0.416667	0.086957	6
T	426	1	1	0.199037	0.783609	1656	2	0.822394	17.5	15.28169	0.782609	1.277778	0.257246	1	0.486111	0.332211	4	1
o	431	0	0	0.111903	0.833333	1080	0.951542	0.951327	14.49188	17.49188	1.333333	2.066667	0.399074	1.666667	0.483063	0.485886	0.366667	0.305556
t	319	1	1	0.230067	0.326087	690	1.138158	1.005525	7.097179	22.32915	0.434783	3.066667	0.462319	1.081967	0.473145	0.485416	1.4	0.021739
a	492	0	0	0.102166	0.805556	1044	0.807018	0.775439	15.51626	18.1687	1.805556	2	0.471264	1.892308	0.535043	0.504686	1.896552	1.777778
l	276	1	1	0.320666	0.130435	276	0.958333	0.75	2.5	22.5	0.130435	7.666667	1	1	0.416667	0.48913	1	1
D	680	0	0	0.10022	0.826087	1748	0.988506	1.429577	15.34118	22.31029	1.391304	2.157895	0.389016	1.738095	0.403715	0.485006	0.105263	2.652174
u	455	1	1	0.148837	0.8	980	0.860558	0.944915	13.93846	17.95824	0.8	2.5	0.464286	1.555556	0.497802	0.513093	1.25	0.371429
e	490	0	0	0.100116	0.861111	1116	1.189076	1.121849	14.32041	17.30204	2.055556	1.612903	0.439068	1.850746	0.461949	0.480612	1.451613	0.805556
B	764	-1	-1	0.090338	0.76087	1610	0.933985	1.411765	14.21335	22.64267	1.869565	2.2	0.474534	2.012346	0.406096	0.492232	0.771429	2.891304
n	457	1	1	0.149491	0.8	980	1.171296	1.02193	13.07878	15.97812	0.8	2.5	0.466327	1.555556	0.467099	0.456518	1.321429	0.142857
c	356	1	1	0.116452	1.944444	2520	0.915344	0	53.98315	17.75281	1.361111	0.7	0.14127	0.924528	0.771188	0.493134	0.228571	9.888889
S	597	1	1	0.071465	0.770833	1776	0.955556	0.964744	18.39196	23.76214	1.958333	1.837838	0.336149	1.905882	0.49708	0.495045	0.378378	0.229167
U	603	1	1	0.123204	0.765957	1692	0.880734	0.977273	17.50415	24.35821	0.765957	2.444444	0.356383	1.493976	0.486226	0.51826	1.083333	0.148936
B	764	-1	-1	0.090338	0.76087	1610	0.933985	1.411765	14.21335	22.64267	1.869565	2.2	0.474534	2.012346	0.406096	0.492232	0.771429	2.891304
O	596	0	0	0.071346	0.916667	2112	0.96129	0.954839	21.61074	23.45638	1.583333	1.954545	0.282197	1.76087	0.491153	0.488674	0.272727	0.291667
A	534	0	0	0.083198	0.934783	1978	0.59882	1	21	24.55993	1.304348	1.906977	0.26997	1.595506	0.488372	0.533911	3.162791	0
L	391	1	1	0.218376	0.630435	1334	0.532819	4.28	6.764706	28.52941	0.630435	1.586207	0.293103	1	0.233266	0.620205	4.172414	5.347826
$	621	-1	-1	0.068308	0.545455	1650	1.085526	0.742268	14.71498	25.91787	1.236364	3.366667	0.376364	1.988235	0.490499	0.471234	0.866667	1.818182

2. B: OCR TESTING.

Task: Use the decision tree discussed above to find the total amount in the given two bills.

Approach and implementation: The same above procedure was repeated for the test bills and depending upon the decision tree the system would be detecting the appropriate digit.

Detecting Characters: In order to detect the characters, I first searched for TOTAL. Once I get at least 2 to 3 correct characters corresponding to TOTAL, I travel rightwards to find the amount.

Assumption: Even if it is apparent in the bills given, the total amount is assumed to be always on the RHS of the word 'TOTAL'.

Displaying Output: I firstly, copied the input image to the output image. Once, I get a match with the character, I also have the bounding box co-ordinates corresponding to that. Hence I passed, a unique combination of RGB value to trace out the bounding box in the output image. Since the colours of the tracing line of boundary box is known, we can also see clearly whether mis-classification happened or not.

This method was adopted for the test bill 1 and test bill 2 detection. However, this method became complicated to implement in part 2(C) due to presence of non English characters and many misclassification.

The following coloured boxes were used to display the bounding boxes for that particular character;

Manan Vyas
USC ID: 7483-8632-00
Email: mvyas@usc.edu

Character	Colour -> correct match	R	G	B
1		192	0	0
2		255	0	0
3		255	192	0
4		255	255	0
5		146	208	80
6		0	176	80
7		0	176	240
8		0	112	192
9		0	32	96
0		99	37	35
.		154	54	52
T		218	150	148
o		230	184	183
t		64	49	81
a		96	73	182
l		177	160	199
D		204	192	198
u		151	71	6
e		226	107	0
B		250	191	143
n		252	213	180
c		79	98	40
S		118	147	80
U		196	215	193
B		216	228	188
O		128	128	128
A		166	166	166
L		191	191	191
$		217	217	217

Manan Vyas
USC ID: 7483-8632-00
Email: mvyas@usc.edu

Results and Discussion:

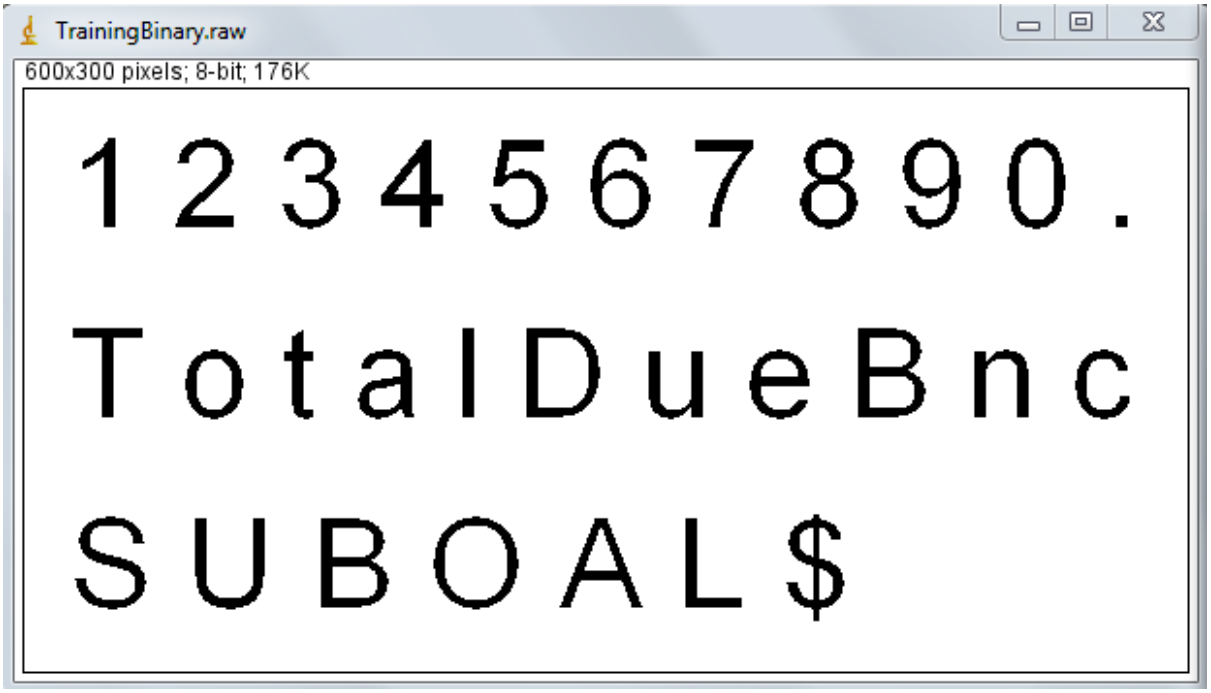


Figure 2.1 Training Image Binarized.

Figure 2.2 Connected Components Labelled for Training Data

Figure 2.3 Connected Components Labelled for Test Bill 1

Figure 2.4 Connected Components Labelled for Test Bill 1

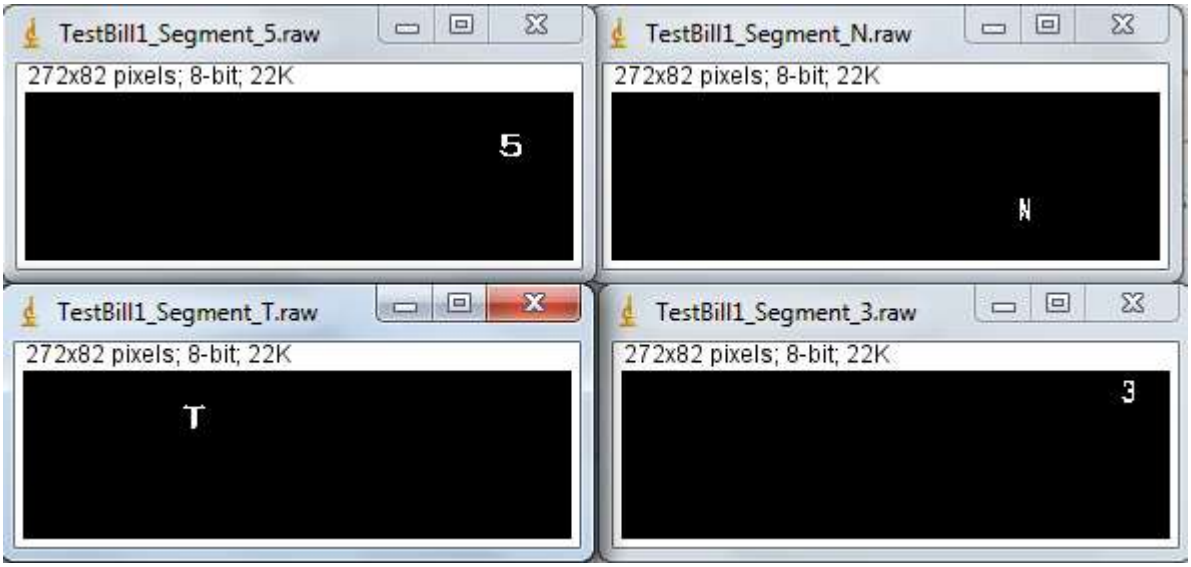


Figure 2.5 : Some of the segmented charcters are shown above after connected component labelling

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

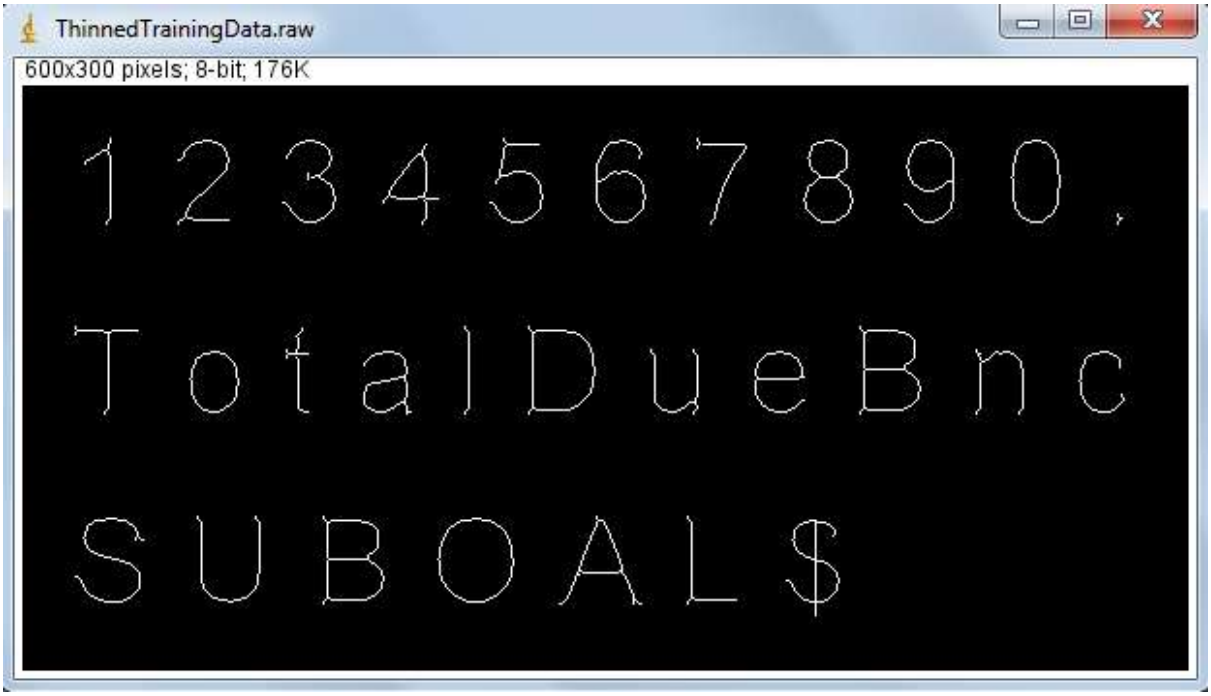


Figure 2.6: Some of the times in training and decision tree thinned segments are required. The above are the thinned characters .As given in the decision tree, thinning is quite useful for detecting characters that are very close to each other.

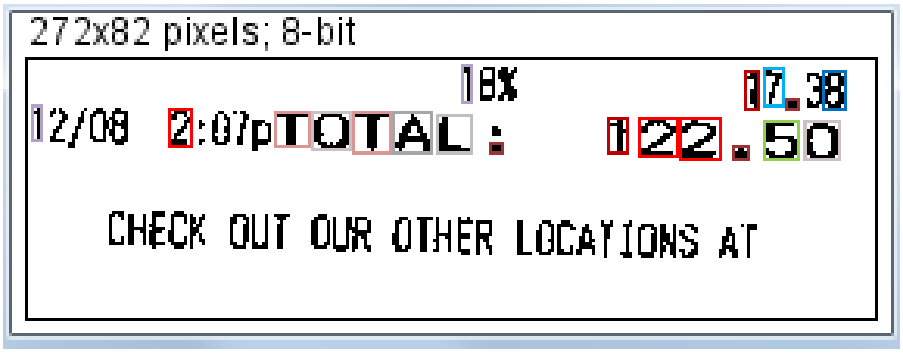


Figure 2.7: Output for Test OCR Bill 1

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

Figure 2.8: Output for Test OCR Bill 2

- It is observed in test ocr bill1 that two 1 are mis-classified as l and two 1's are classified correctly.
- The 'O' in total is misclassified as 'o' and the 0 in 122.50 is classified as 'O'.
- Thus, it is inferred that the the are many mis-classification among similar looking characters like o,0,O.
- Both the '2' are recognized correctly.
- The decimal point is recognized correctly if it is a perfect square.
- 5 and 7 is correctly classified, 3 is not classified at all at the top.
- Once I encounter the TOTAL: 122.50 I stopped further testing as we are interested only in the total amount.

In OCR BILL 2 I tried to recognize the total amount and other characters as well.

- The 'A' in total was grossly misclassified as 4. The amount 41.45 was correctly recognized.
- The 2 was correctly classified correctly, and one '1' is wrongly classified as 2.
- The character 'C' and 'G' gave the same output.
- T were recognized correctly.
- '0' and '.' Were also classified correctly.

- Using mostly the normalized parameters helped because of the non-uniformity of the training and testing data set.
- Even though the characters in TestOCRBill1 and TestOCRBill2 have different size and shape as compared to the training data, their inherent and unique properties still remain close enough to be detected with some degree of respectable accuracy by this approach.
- However, there is still room for improvements as this implemented is highly specialized to this problem only and might not work with other characters or other testing images.

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

2 .C: OCR TESTING ON RESTAURANT BILL:

Task: To recognize the individual amounts on the printed bill and also the total amount printed on a real-life restaurant bill.

Approach and Implementation:

The same approach as discussed in the part 2(a) and part 2(b) was implemented.

The restaurant bill was first binarized with a threshold of 172 and then isolated 8-connected pixels which were noisy pixels were removed using the 8-connectivity isolated pixel removal as discussed in the approach in 2(a).

The decision tree depicted in problem 2(a) was used for learning and as a blue print for classification.

Even though the restraint bill was tilted, the testing was carried out on that only, this approach does not deals with pre-processing and correcting the image before applying OCR testing on it, although, it is highly recommended.

Results and Discussion:

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

Figure 2.9: Given Restaurant Bill.

Figure 2.10: Binarized Restaurant Bill.

For simplicity I assumed that all the amounts are on the right hans side of the bill and hardcoded the RHS starting point and applied connected component labelling only on the RHS of the bill where the amounts are present.

Figure 2.10: Connected Component Restaurant Bill.

Following is the recognition output obtained;

Figure 2.10: Output for Restaurant Bill

- No characters were recognized.
- One of the reasons being that the image is highly titled and the properties of these highly tilted may not be close enough to the training data.
- Noise also affects the recognizing ability in this implementation.
- This problem certainly provides the motivation to improve the implemented OCR program.

Manan Vyas
USC ID: 7483-8632-00 Email: mvyas@usc.edu

Problem # 3: IMAGE SEGMENTATION.

Motivation: Image segmentation is one of the most important problems in Computer Vision domain. Texture segmentation, which was implemented in previous homework # 3 dealt with the ability to classify and identify different textures. However, in image segmentation is different. It is the first step in object recognition in computer vision and image processing domain. Thus it is important to study and implement image segmentation.

Task: To perform Image segmentation using different techniques as mentioned in the homework, on the three given image.

3. A – Image Segmentation Using K-Means

Task: (1) To perform image segmentation using k-means using the [R(i) G(i) B(i)]' as the feature vector for the whole image while giving the number of clusters intuitively.

(2) Propose and implement a new feature for the given image and segment the image using that new feature [The only constraint being that we are not allowed to use the spatial location of the pixel within the image as its unique feature] and compare the output with the part (1).

Code implemented in: All Processing in C++/ Only Kmeans output shown in Matlab.

Approach and Implementation:

(1) The algorithm to implement the image segmentation using clustering using RGB clustering is as follows;

Step 1: Read the image.
Step 2: Read each of the RGB values and write it into a separate column of a '.csv' file. Thus we have a (height*width)x3 dimension '.csv' file containing all the RGB intensity values present in that particular image.
Step 3: Read the '.csv' file in MATLAB and give it as an input parameter to the kmeans function of the Matlab along with the intuitive choice of # of clusters.
Step 4: Construct an output image containing RGB values corresponding to the intensity values of the cluster centroids.
Step 5: Write the output image into a '.raw' file.

Thus we get the segmented image corresponding to clustering and segmentation based on the intensity values of the RGB components present in the image.

(2) For the alternative feature, I used the HS colour space as one of the HSI (Hue,Saturation and Intensity) parameters(Hue) is non-linearly related to RGB. The formula for conversion is given by[6];

A) Normalize RGB values;
Rnorm = R/(R+G+B);
Gnorm = G/(R+G+B);
Bnorm = B/(R+G+B);

B) $h =$
$(0.5(Rnorm - Gnorm) * 0 or\ 2Pi +$
$cos^{-1} \frac{Rnorm - Bnorm}{(Rnorm - Gnorm)^2 + (Rnorm - Gnorm)(Gnorm - Bnorm))^{\frac{1}{2}}} for\ Bnorm < Gnorm$

$$s = 1 - 3(\min(Rnorm, Gnorm, Bnorm)) and\ \ i = \frac{R + G + B}{255} * 3\ and\ then$$
$$Normalize\ ; h = h * \frac{180}{pi}; s = s * 100\ and\ i = i * 255$$

Then the same procedure explained in part (a) was performed.

Results and Discussion:

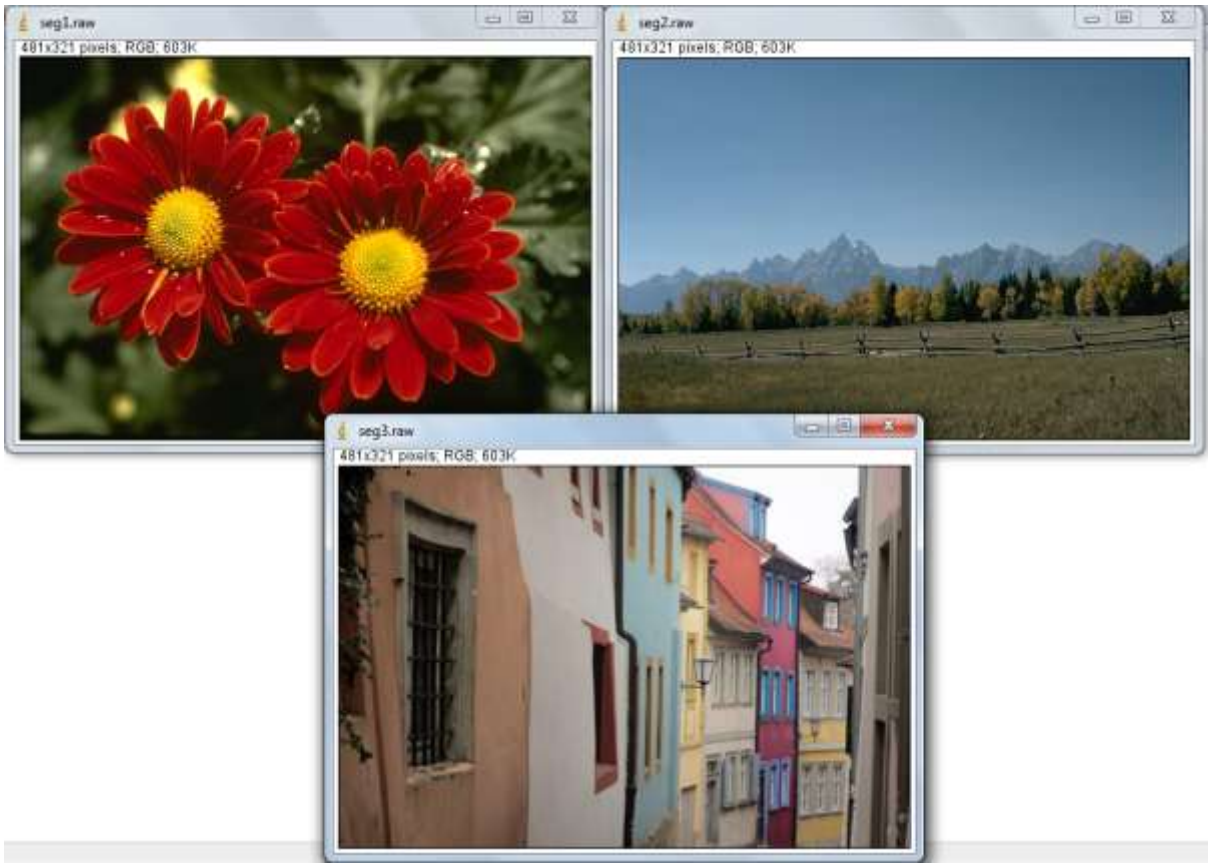


Figure 3.1: Given Images : seg1.raw (Top Left), Seg2.raw(To Right) and Seg3.raw(Bottom).

First all the three images were segmented using the same number of clusters even though intuitively, we can observe that that is not the case. Thus, initially the three images were segmented using the # of clusters = 3.

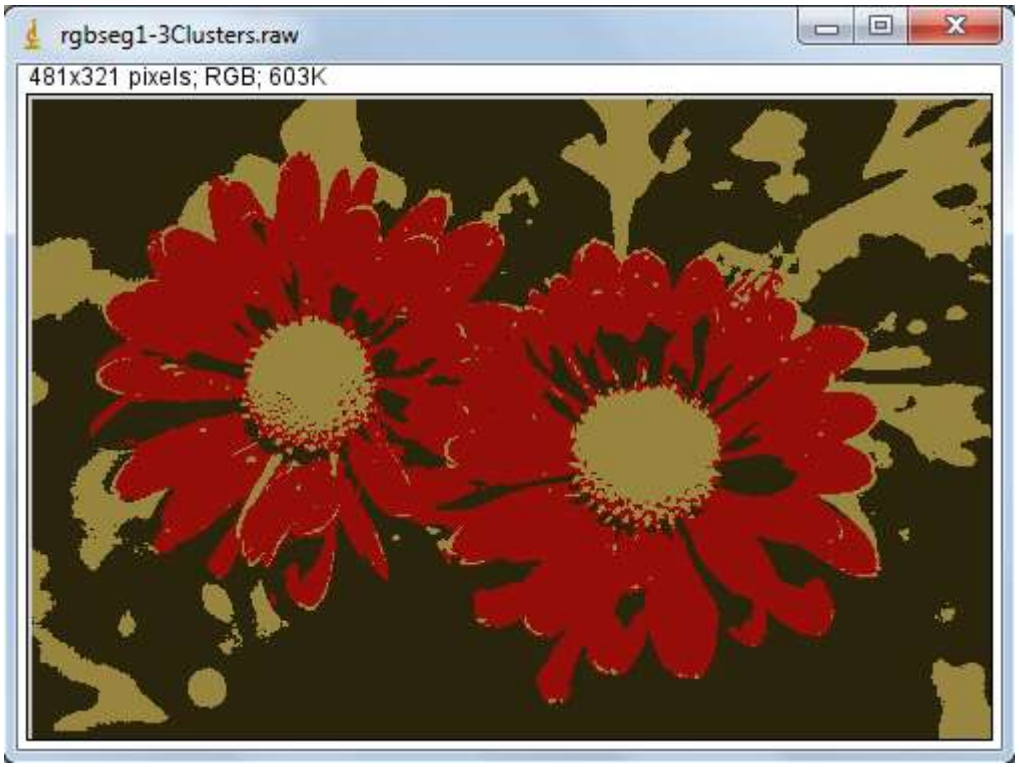


Figure 3.2: Output of Image segmentation using RGB on Seg1 image with 3 clusters

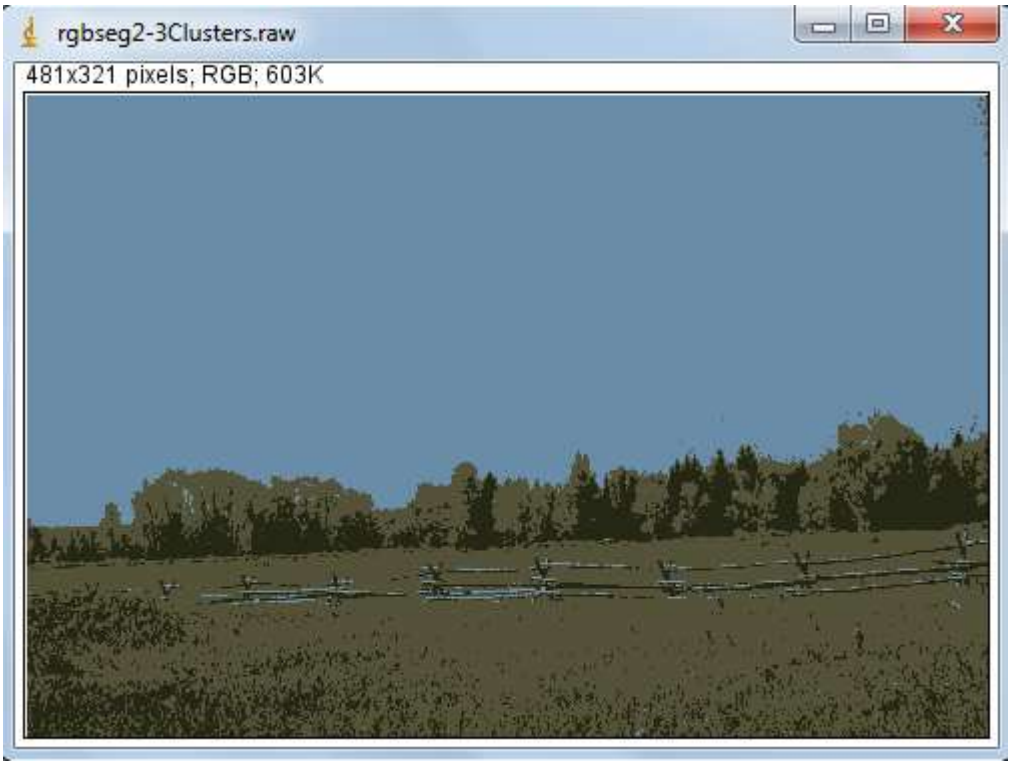


Figure 3.3: Output of Image segmentation using RGB on Seg2 image with 3 clusters.

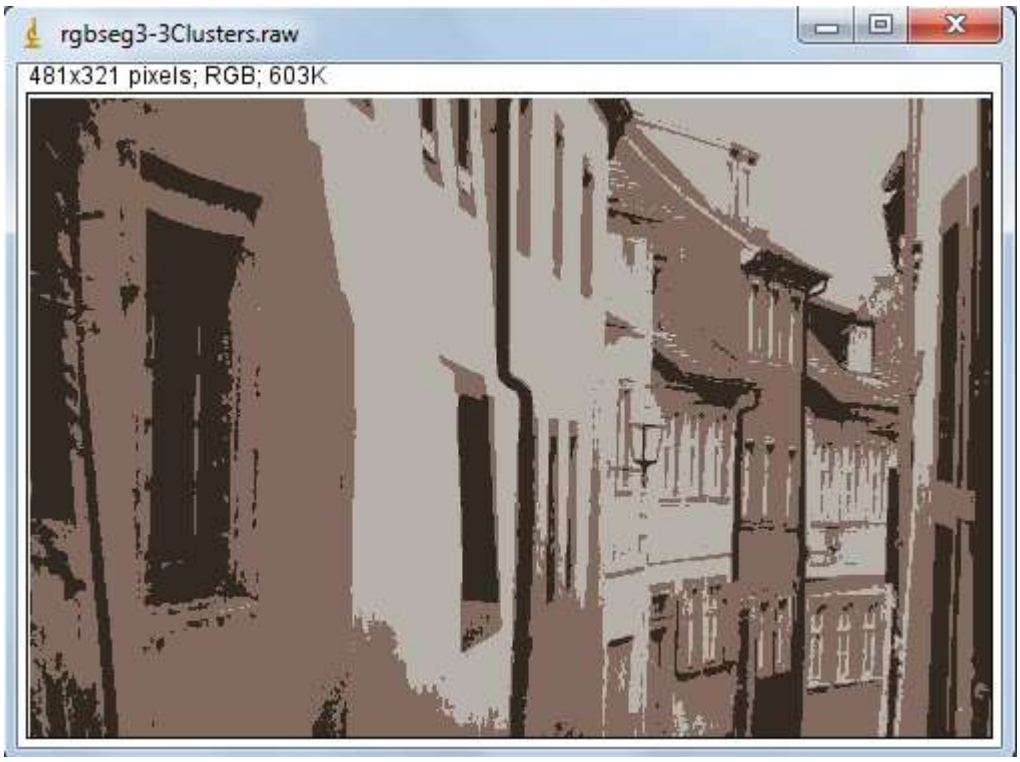


Figure 3.4: Output of Image segmentation using RGB on Seg2 image with 3 clusters

- It is thus observed, that the image seg1 is segmented quite well since that image can be said to contain three regions namely, Flower petals(Red Colour), Flower Centers (Yellow Colour) and Background.

- However, 3 clusters are far too less in the other two images where in Seg2 the mountains are lost and are classified as the sky , trees are lost and are classified as grassland. Also in the seg3 image, only the walls, windows and sky are clustered separately and all the information about the different buildings are lost. Hence, 3 clusters are optimum for seg1 image but fails for seg2 and seg3 image.

- Hence rather than clustering using the same number of clusters in all three images it is a better idea to intuitively observe the image and decide on the number of clusters. The only disadvantage in this method being that it is very subjective in nature but I proceed nevertheless. As discussed earlier, 3 clusters are optimum for seg1 image.

- Now observing the image seg2, it seems that there are 4 RGB/Segments in the image namely, Sky,Mountains, Trees and Grasslands and hence in order to segment this image, I used 4 clusters for Kmeans.

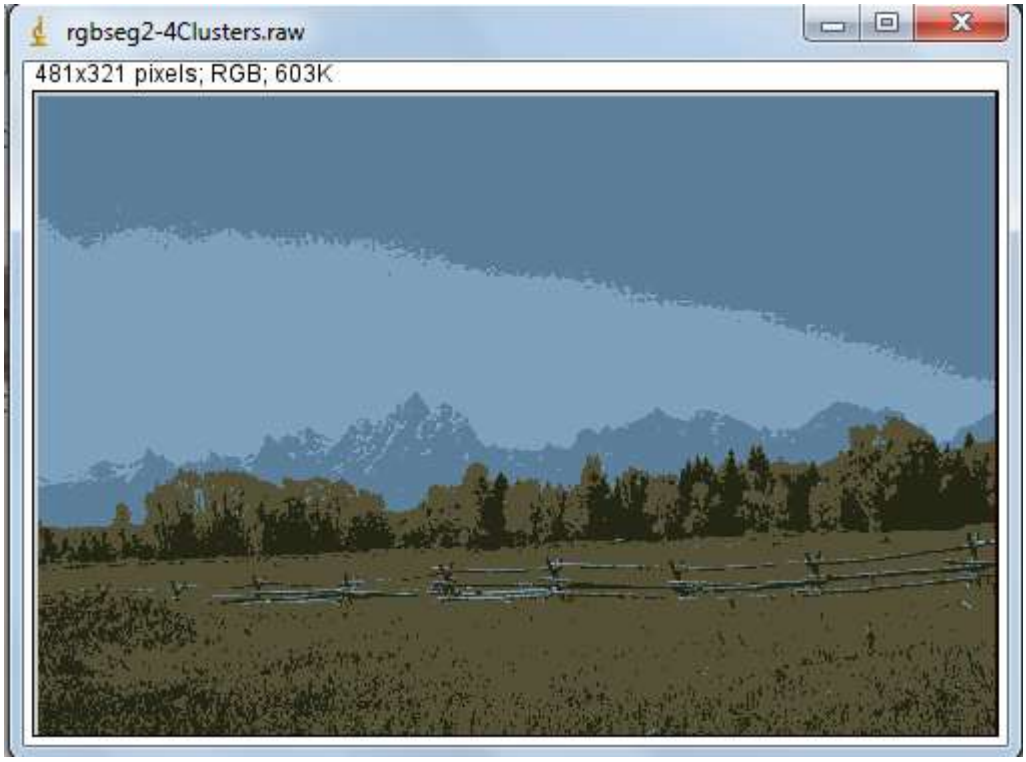


Figure 3.5(a): Output of Image segmentation using RGB on Seg2 image with 4 clusters

Figure 3.5(b): Output of Image segmentation using RGB on Seg1 image with 5 clusters (5 clusters is redundant for this image and not needed)

Now observing the image seg3, it seems that there are about 7 to 9 RGB/Segments in the image namely, amongst the various buildings and sky and hence in order to segment this image, I used 7 and 10 clusters for Kmeans and following are the results obtained.

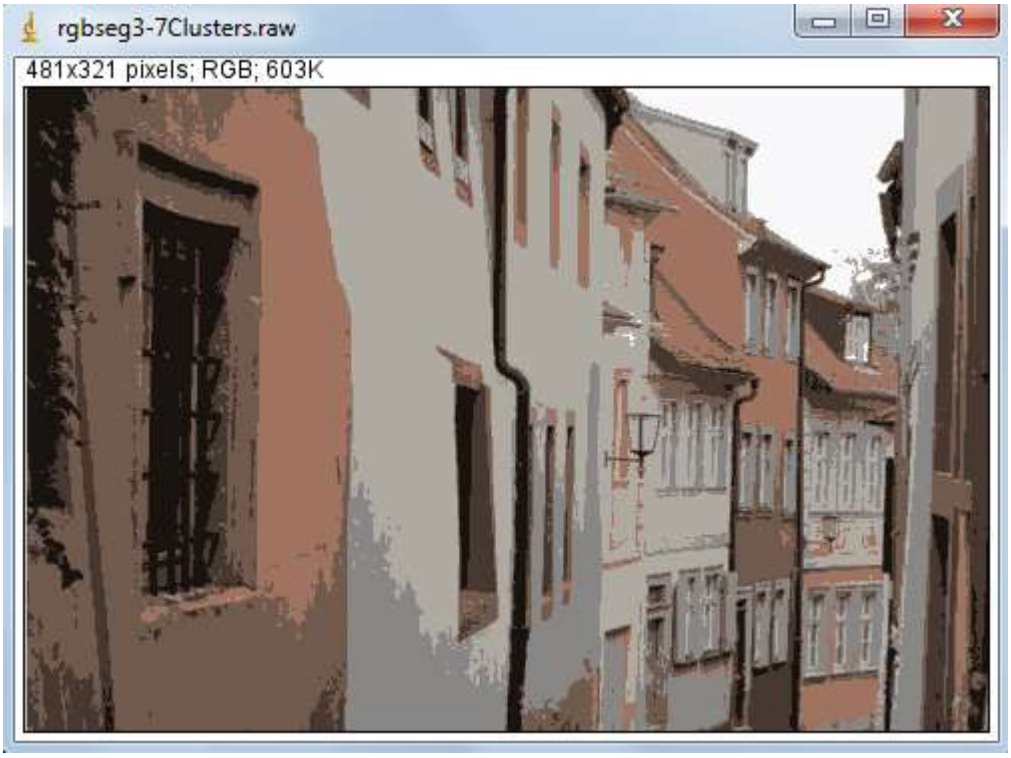


Figure 3.6: Output of Image segmentation using RGB on Seg3 image with 7 clusters

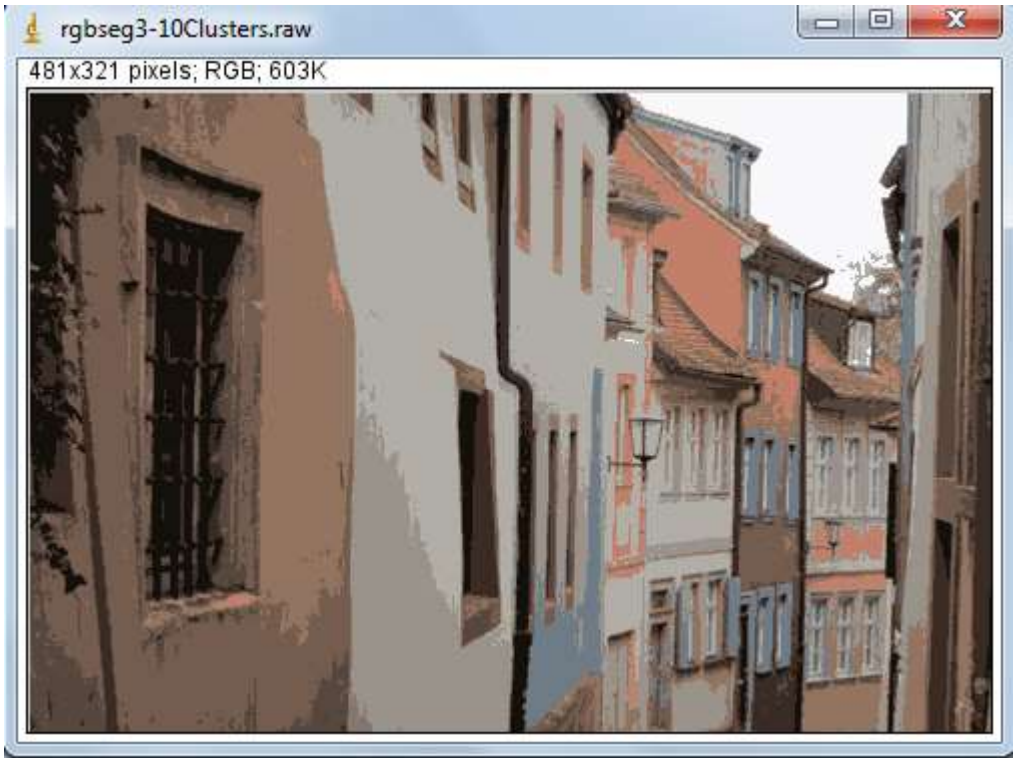


Figure 3.7: Output of Image segmentation using RGB on Seg3 image with 10 clusters

(b): The following results were obtained for the new feature.

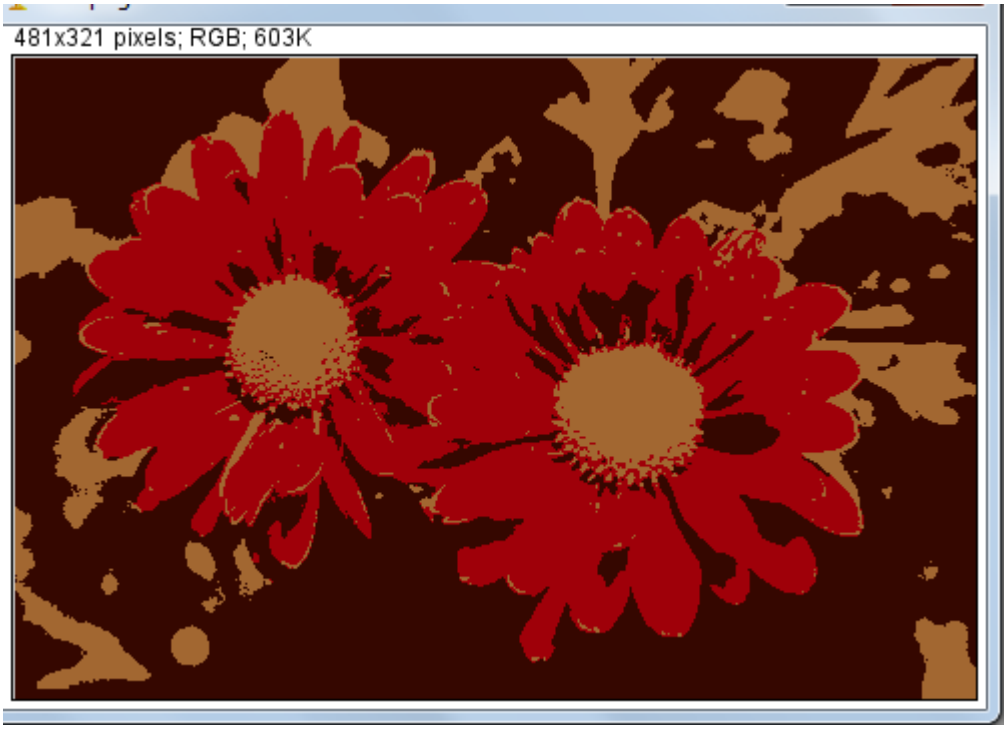


Figure 3.8: Output of Image segmentation using HSV on Seg1 image with 3 clusters.

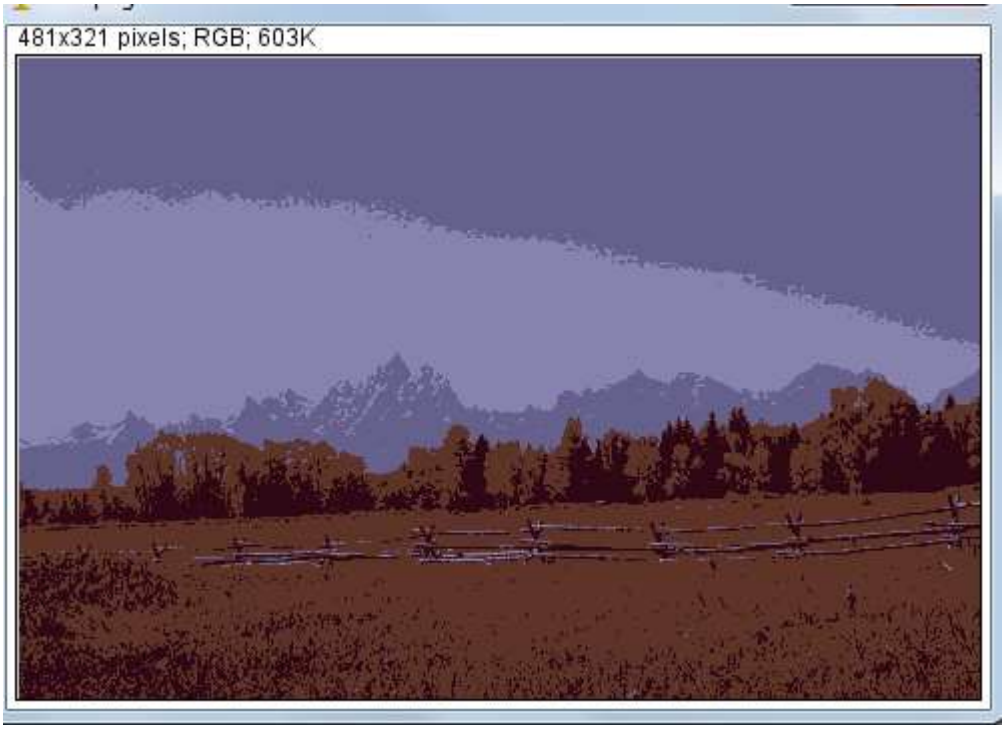


Figure 3.9: Output of Image segmentation using HSV on Seg2 image with 4 clusters

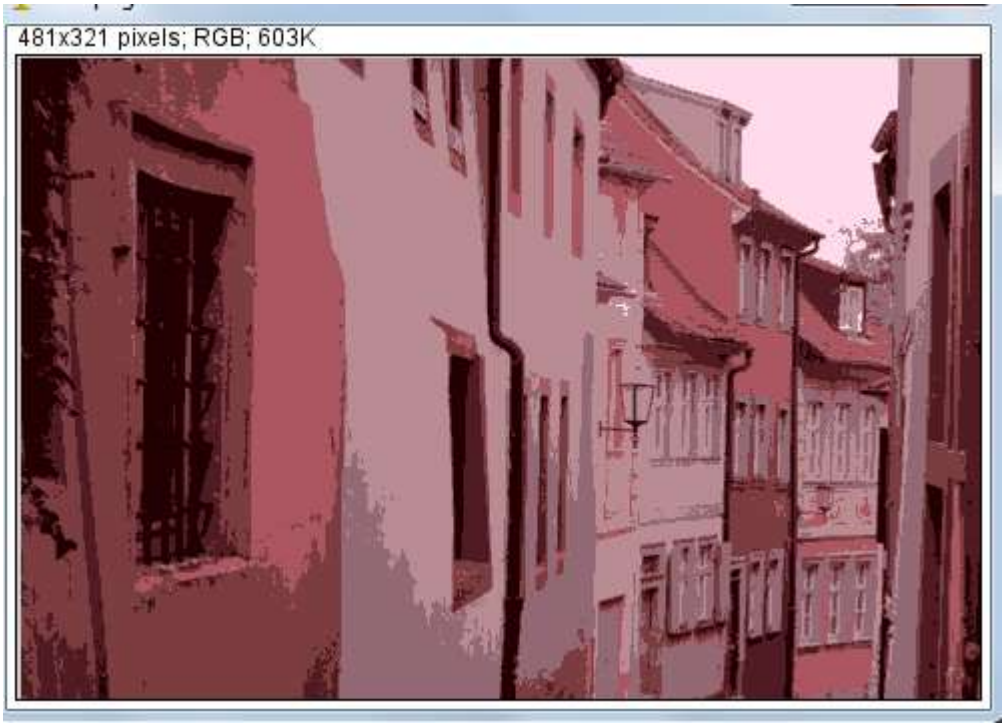


Figure 3.10: Output of Image segmentation using HSV on Seg3 image with 7 clusters

3. B: MEAN SHIFT FILTERING:

Task: To segment the given three images using Mean Shift Filtering as proposed in [4].

Code implemented in: C++.

Approach and Implementation: As proposed in the paper[4], the mean shift filter uses a density gradient to characterize the distribution of pixels in the image. The algorithm searches along the gradients within the images to search for local maxima and find these local maxima points in the colour space of the image. The algorithm implemented is as follows;

Step 1: Initialization – Select an initial starting point xi = (Ri,Gi,Bi)

Step 2: Iteration – Search along the maximum positive gradient direction (steepest ascend direction)

This can be done by the following formula for j=1..2..3.. Height;

$$y_{j+1} = \frac{\sum_{i=1}^{N} xi\, W(yik-1, j)}{\sum_{i=1}^{N} W(yik-1, j)} \;;$$

Where I is in the window region aroung the pixel of interest. The window is generally of the size 6*sigmaS+1 which is used in the implementatiom.

Step 3: Convergence Check condition

$$\frac{||\Delta \boldsymbol{yj}+\mathbf{1}||}{\boldsymbol{yj}+\mathbf{1}} < \boldsymbol{threshold\ then\ Stop\ iteration}\ ;\ \Delta \boldsymbol{yj}+\mathbf{1} = (\boldsymbol{yj}+\mathbf{1}) - \boldsymbol{yj}$$

Results and Discussion:

The following results were obtained for seg 1 image,

Figure 3.11 : Mean Shift Filtering output for seg 1 image with convergence after 10 iterations , sigmaS 5, sigmaR 10, i.e: 15x15 window

Figure 3.12: Mean Shift Filtering output for seg 1 image with convergence after 15 iterations , sigmaS 15, sigmaR 10, i.e: 45x45 window

Figure 3.13: Mean Shift Filtering output for seg 1 image with convergence after 25 iterations , sigmaS 10, sigmaR 20, i.e: 30x30 window

The following results were obtained for seg 2 image,

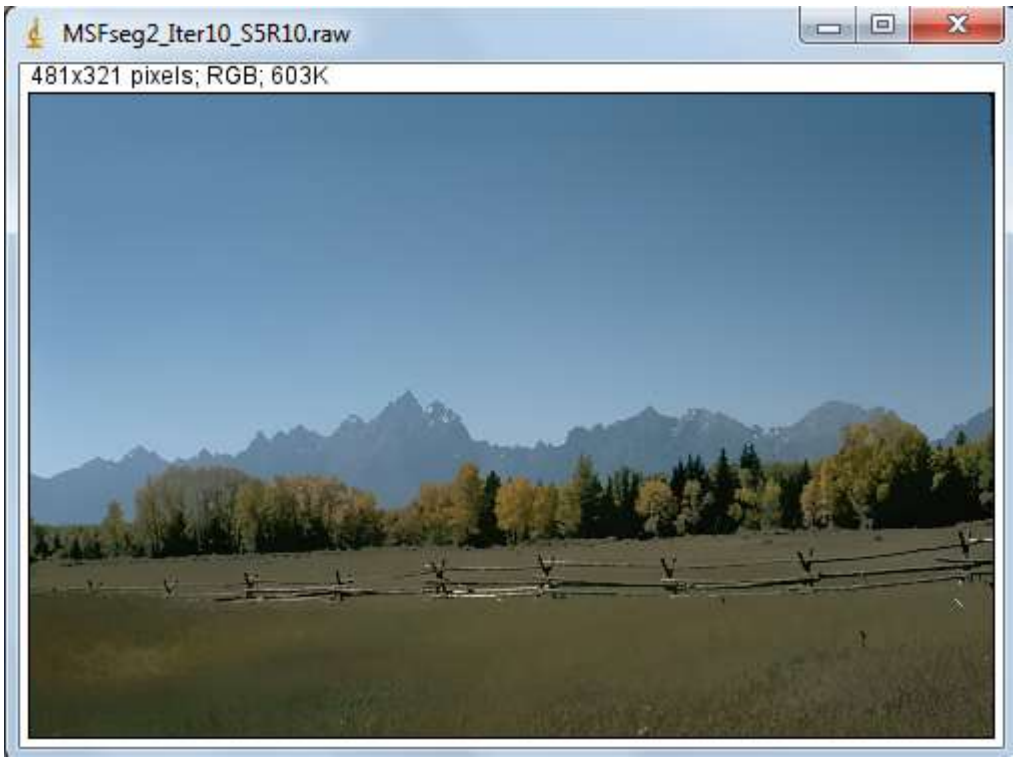


Figure 3.14 : Mean Shift Filtering output for seg 2 image with convergence after 10 iterations , sigmaS 5, sigmaR 10, i.e: 15x15 window

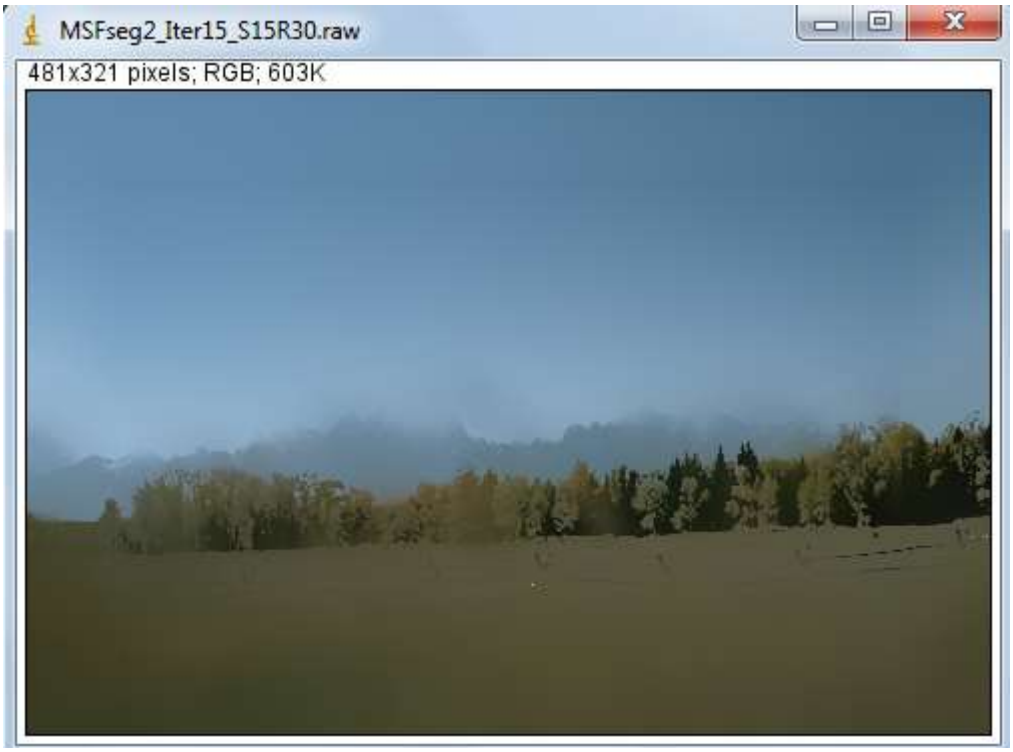


Figure 3.15: Mean Shift Filtering output for seg 2 image with convergence after 15 iterations , sigmaS 15, sigmaR 10, i.e: 45x45 window

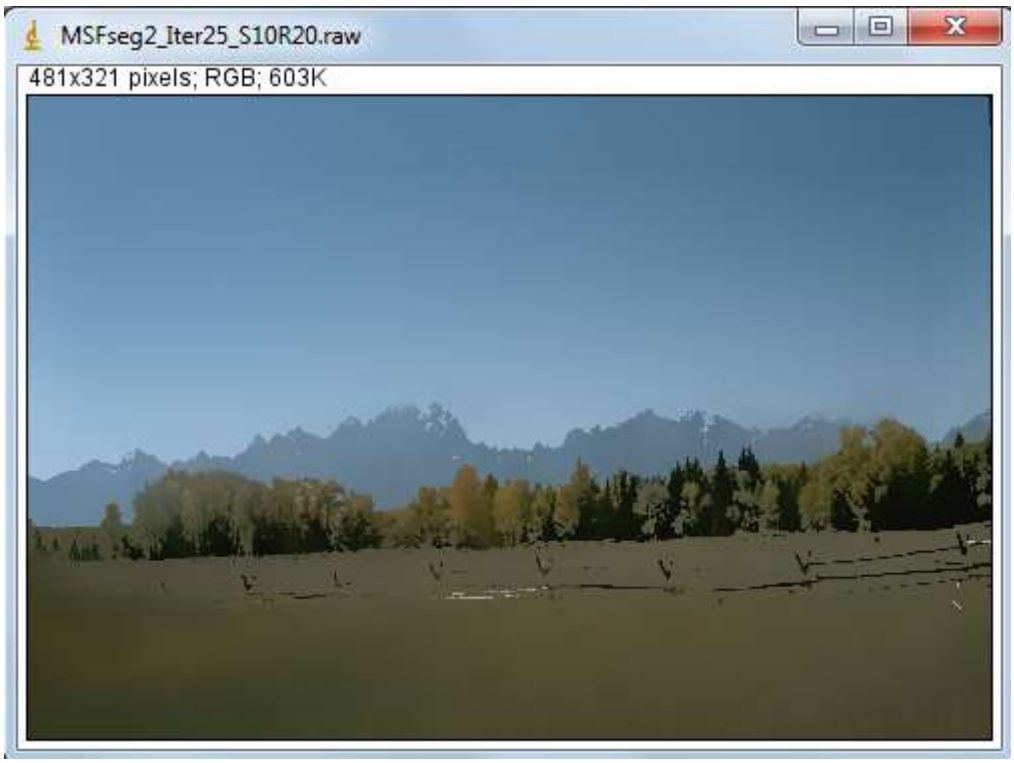


Figure 3.16: Mean Shift Filtering output for seg 2 image with convergence after 25 iterations , sigmaS 10, sigmaR 20, i.e: 30x30 window

The following results were obtained for seg 3 image,

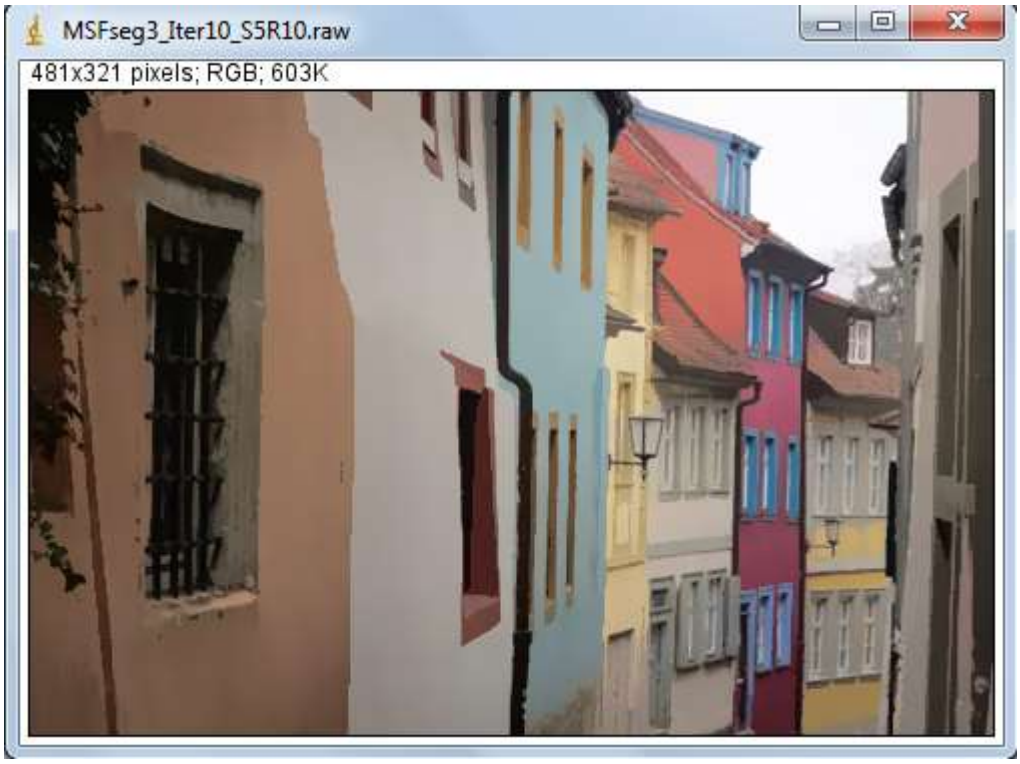


Figure 3.17: Mean Shift Filtering output for seg 3 image with convergence after 10 iterations , sigmaS 5, sigmaR 10, i.e: 15x15 window

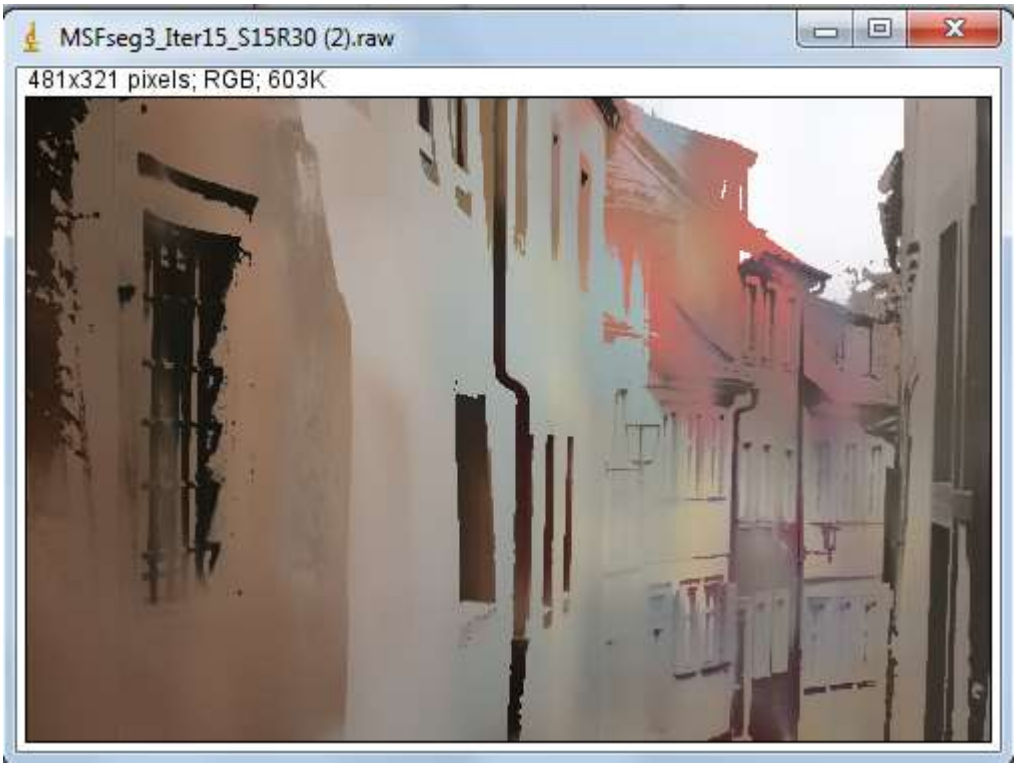


Figure 3.18: Mean Shift Filtering output for seg 3 image with convergence after 15 iterations , sigmaS 15, sigmaR 10, i.e: 45x45 window

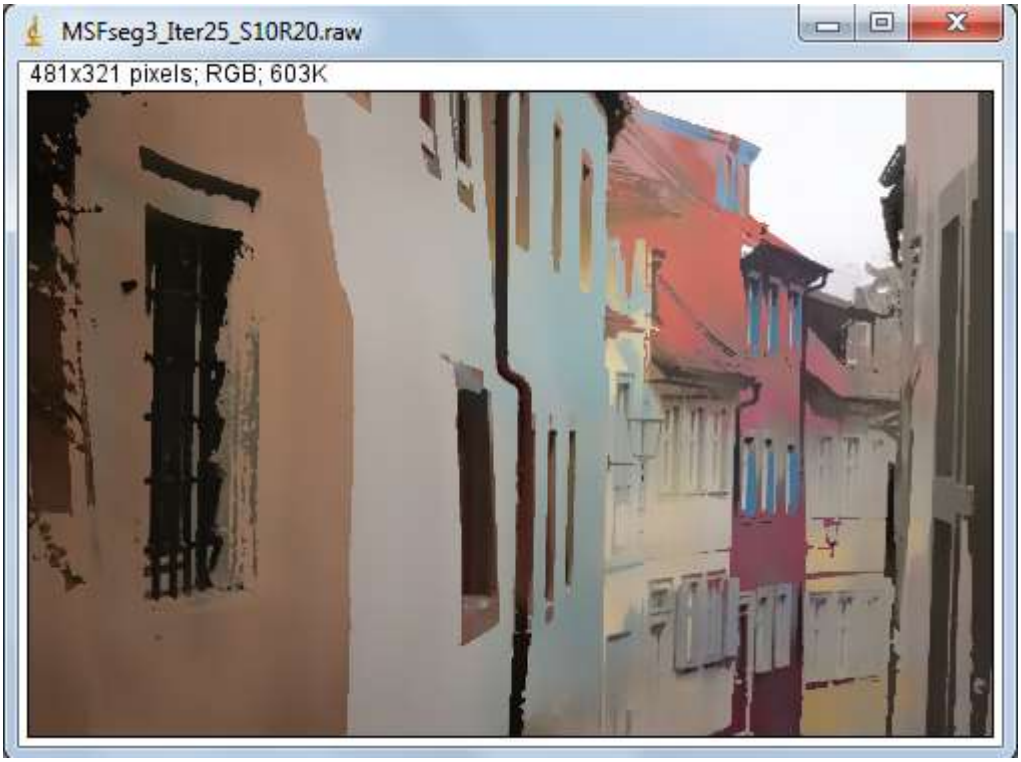


Figure 3.19: Mean Shift Filtering output for seg 3 image with convergence after 25 iterations , sigmaS 10, sigmaR 20, i.e: 30x30 window

- It is thus observed that larger the size of the window, which depends upon the value of sigmaS , more the size window produces more 'colour blending' within the different parts of the image. There are less precisely defined edges or boundaries between differents parts if objects within an image.

- More the number of iterations of convergence, more amounts of pixels in window are replaced by the center of the weights (also called the local maxima value) and better the image segmentation. However, time complexity the program increases manifold.

- It is thus concluded, that mean shift filtering does a better job of image segmentation than the techniques in Problem 3(a)1-2 because the job of handling the clusters is automated in this particulalar case and it is handled by the window size for mean shifted local maxima calculation.

- Just by adjusting the kernel parametrs, sigmaS and sigmaR we can set the window size and the threshold for segmentation , similar to what we could do in Biltaeral filtering. This is also one of the reasons why mean shift filter is better than the clustering solution discussed in Problem3(a1-2).

- Mean shift filtering becomes the basis for mean shift clustering problem discussed next

3. C: MEAN SHIFT CLUSTERING:

Task: To implement mean shift clustering using the mean shift filtering from problem 3(b) and apply it on the given three images.

Code written in: C++.

Approach and Implementation:

Mean shift clustering is based on labelling the common pixels that are to be contained within a cluster and also finding out the centres of the common clusters from the mean shift filtering output.

It is one of the important techniques for object recognition, visual tracking etc.

The output of the means shift filtering is given as the input to the mean shift clustering problem. The following algorithm was implemented for mean shift clustering using mean shift.

Step 1: Read the mean shift filtered image

Step 2: Initialze initial RGB centers as the RGB intensites of the pixel at 0,0 location of the image.

Step 3: Initlaize intial labels as 0 and some arbitrary labelling threshold between [0,255]

Step 4: Traverse the whole image pixel by pixel and calculate the threshold for labelling

Step 5: For any current pixel Ii(I,j) with RGB values (Ri,Gi,Bi) and calculate the threshold as the L2 norm of the current pixel and the current center Ic(I,j) with RGB values given as (Rc,Gc,Bc). It is given by,

$$t = \sqrt{(Ri - Rc)^2 + (Gi - Gc)^2 + (Bi - Bc)^2}$$

If the current pixels intensity value is lesser than the above calculated threshold then assign the same label that is assigned to the center to the current pixel of interest.

If the current pixels intensity value is greater than the above calculated threshold then increment label by 1 and assign the current pixel the updated label and calculate new center (assuming there were n pixels that constituted the current center) by the formula ;

$$new\ center\ R = \frac{\big((current\ center\ R\ value\ x\ n) + current\ pixel\ R\ value\big)}{n}\ ; similiarly\ for\ G, B$$

Thus, suppose if we have 4 current centers and the current pixel intensity lies above the current threshold then assign an incremental label to the current pixel of interest and the for the center computation we must compare this with the 4 previously calculated center to determine to which center that the current pixel is closest to.

Step 6: Do this procedure for all the pixels in the image.

Step 7: Assign a particular unique RGB value (preferable the center RGB value to the cluster that the label belongs to) and display the output image which is the mean shift clustered image of the original input image.

Results and Discussion:

The following results were obtained for the mean shift clustering.

Figure 3.20: Mean Shift Clustering output for seg 1 image

Figure 3.21: Mean Shift Clustering output for seg 2 image

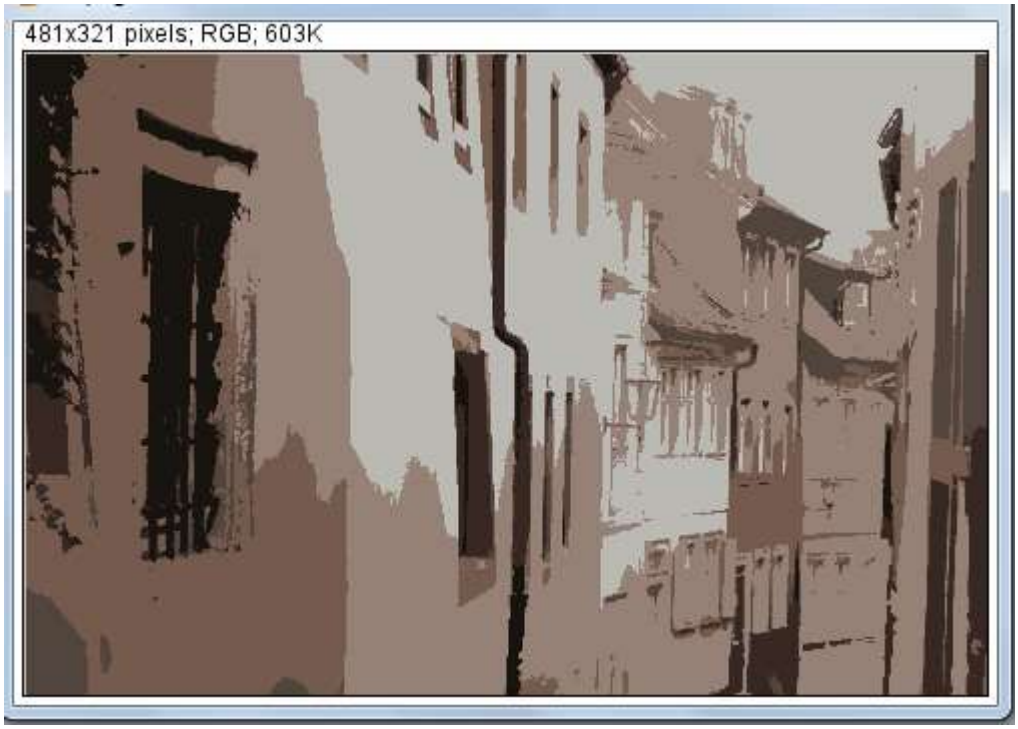


Figure 3.22: Mean Shift Clustering output for seg3 image

- The main advantage of mean shift clustering over most of the other clustering method is that it produces the labels automatically and hence the program automatically decides the number of clusters (and consequently number of cluster centers for a particular image) and it is highly dependent upon the contents of the image.

- This it can be seen that for the seg1 image, the mean shift clustering produces 3 clusters (Flower petals, flower centers and background).

- For the seg2 image, it produces 4 clusters, which are grass, trees, mountains and sky (a part of the darker shade of the sky is clustered same as that of mountain, probably because they contain similar RGB values)

- For seg 3 image, the program produces 6 different clusters with 4 corresponding to the different buildings, 1 related to the sky (small visible portion) and the last corresponding the windows.

- Mean shift clustering is one of the important steps towards the higher goals of object recognition and visual tracking of objects within the computer vision domain.

References:

[1]. EE569 HW#4 Fall 2013. Page 1. Figure #2.

[2]. (online) < http://en.wikipedia.org/wiki/Barycentric_coordinate_system > dated 11-03-2013

[3]. (online) < http://en.wikipedia.org/wiki/Bilinear_interpolation > dated 11-30-2013

[4] D. Comaniciu and P. Meer, \Mean shift: A robust approach toward feature space analysis," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 5, pp. 603{619, 2002.

[5] EE569 Fall 2013.Discussion7 : Slide # 15

[6] (online) < http://www.cse.usf.edu/~mshreve/rgb-to-hsi.pdf> dated 30th Nov,2013.