AML_HW7-Part2

Vishal Dalmiya (Dalmiya2); Himanshu Shah (Hs8); Deepak Nagarajan (deepakn2)

April 7, 2018

Image segmentation using EM You can segment an image using a clustering method - each segment is the cluster center to which a pixel belongs. In this exercise, you will represent an image pixel by its r, g, and b values (so use color images!). Use the EM algorithm applied to the mixture of normal distribution model lectured in class to cluster image pixels, then segment the image by mapping each pixel to the cluster center with the highest value of the posterior probability for that pixel. You must implement the EM algorithm yourself (rather than using a package). Test images are here, and you should display results for all three of them. Till then, use any color image you care to.

Segment each of the test images to 10, 20, and 50 segments. You should display these segmented images as images, where each pixel's color is replaced with the mean color of the closest segment

We will identify one special test image. You should segment this to 20 segments using five different start points, and display the result for each case. Is there much variation in the result?

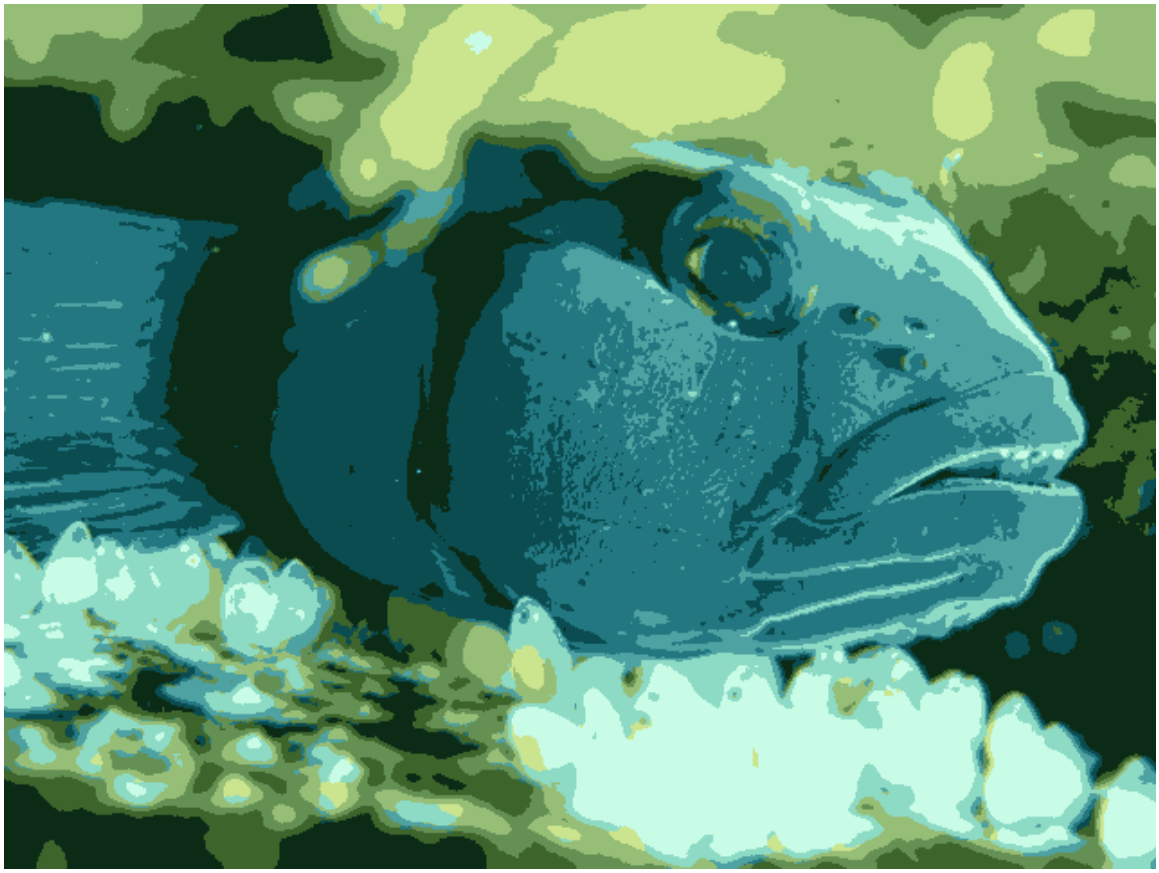
```
library(imager)
library(grid)
library(readr)
library(pdist)

im1 <- load.image('RobertMixed03.jpg')
im2 <- load.image('smallstrelitzia.jpg')
im3 <- load.image('smallsunset.jpg')



#Number of pixels

df1 = data.frame(
  red = matrix(im1[, , 1], ncol = 1),
  green = matrix(im1[, , 2], ncol = 1),
  blue = matrix(im1[, , 3], ncol = 1)
)

df2 = data.frame(
  red = matrix(im2[, , 1], ncol = 1),
  green = matrix(im2[, , 2], ncol = 1),
  blue = matrix(im2[, , 3], ncol = 1)
)

df3 = data.frame(
  red = matrix(im3[, , 1], ncol = 1),
  green = matrix(im3[, , 2], ncol = 1),
  blue = matrix(im3[, , 3], ncol = 1)
```

```
)

dispImg = function(df, seg, img)
{
  # number of segments
  S  = seg
  K  = kmeans(df, S)
  im = img
  N  = dim(im)[1] * dim(im)[2]

  if(seg == 10)
  {
    thres = 0.0000001
  }
  else if(seg == 20)
  {
    thres = 0.0001
  }
  else
  {
    thres = 0.001
  }

  R = matrix(K$centers[K$cluster, "red"], dim(im)[1], dim(im)[2])
  G = matrix(K$centers[K$cluster, "green"], dim(im)[1], dim(im)[2])
  B = matrix(K$centers[K$cluster, "blue"], dim(im)[1], dim(im)[2])


  col = rgb(R, G, B)
  dim(col) = dim(R)


  V = 3


  # R, G & B values of all pixels
  X = matrix(rep(0, V * N), nrow = V, ncol = N)
  X = t(df)
  X = X * 255
  # Weights
  W = matrix(nrow = N, ncol = S)

  # Weightage of each segment
  pi = rep((1 / S), S)

  # Center of normal distribution for each segment
  mu = matrix(nrow = V, ncol = S)
  mu = t(K$centers)
  mu = mu * 255
  oldmu = mu

  iter = 1
  # Run till convergence
```

```
while (1)
{
  # Dim: S X N
  term = matrix(nrow = S, ncol = N)
  for (j in 1:S)
  {
    # Dim: 1 X N
    term[j,] = as.matrix(pdist(t(X), t(mu[, j])))
  }

  # Find the square of the distance
  term = term ^ 2
  # Find the minimum for each of the topics
  # Dim: 1 X N
  dist_min = apply(term, 2, min)

  # Substract the minimum
  adj_term = t(t(term) - dist_min)

  # W : N X S
  W = t(exp(adj_term * (-0.5)))

  for (j in 1:S)
  {
    W[, j] = W[, j] * pi[j]
  }

  # Dim: N X 1
  den = rowSums(W)

  W = W / den

  ############## M Step ##############

  # X : V X N
  # W : N X S
  # numer : V X S
  numer = X %*% W
  den = as.numeric(colSums(W))

  for (j in 1:S)
  {
    mu[, j] = (numer[, j]) / (den[j])
  }

  # Compute pi
  (pi = colSums(W) / N)

  iter = iter + 1
  temp = max(abs(mu - oldmu))

  if (temp < thres)
  {
```

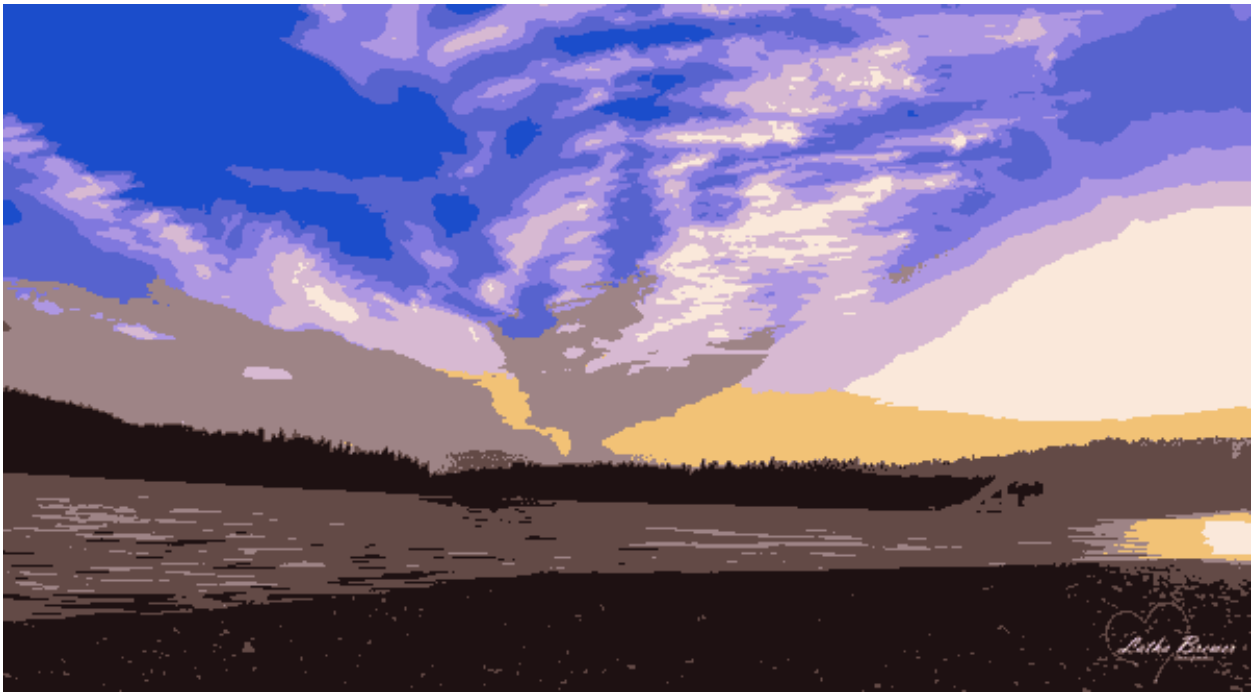
```
      print(paste("Iteration # ",iter))
      print(paste("Treshold ",temp))
      break
    }
    oldmu = mu
  }

  newX2 = matrix(rep(0, V * N), nrow = V, ncol = N)
  for (i in 1:N)
  {
    index = which.max(W[i, ])
    newX2[, i] = mu[, index]
  }

  newX2 = newX2 / 255

  R = matrix(newX2[1, ], dim(im)[1], dim(im)[2])
  G = matrix(newX2[2, ], dim(im)[1], dim(im)[2])
  B = matrix(newX2[3, ], dim(im)[1], dim(im)[2])


  col = rgb(R, G, B)
  dim(col) = dim(R)
  grid.newpage()
  grid.raster(t(col))
}

# 10, 20, and 50 segments for 2 test images
for (i in c(10,20,50))
{
  print(paste("Test Image 1 with segments = ",i))
  dispImg(df1, i, im1)
  print(paste("Test Image 2 with segments = ",i))
  dispImg(df2, i, im2)
  print(paste("Test Image 3 with segments = ",i))
  dispImg(df3, i, im3)

}
```

```
## [1] "Test Image 1 with segments =  10"
## [1] "Iteration #  54"
## [1] "Treshold  1.02235730992106e-09"
```

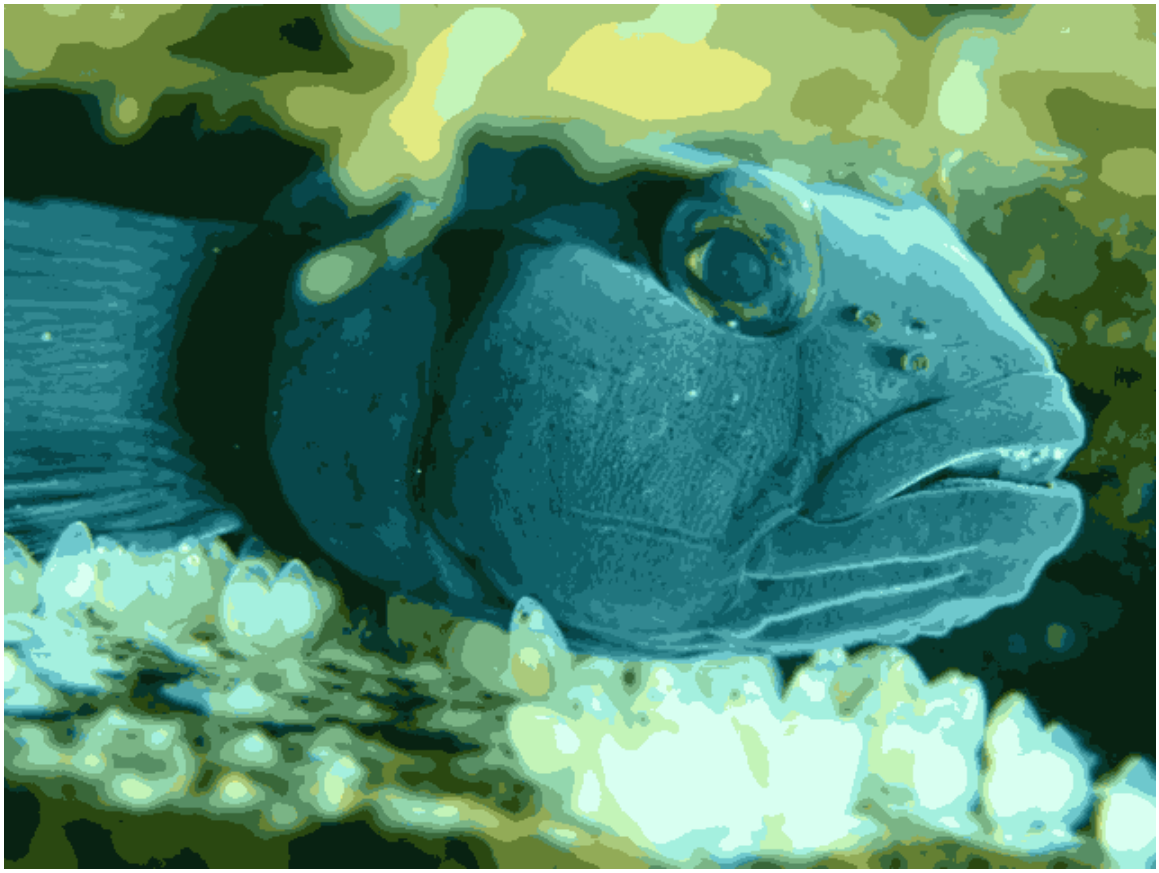


```
## [1] "Test Image 2 with segments =  10"
## [1] "Iteration #  52"
## [1] "Treshold  8.96982044196193e-10"
```

```
## [1] "Test Image 3 with segments =  10"
## [1] "Iteration #  70"
## [1] "Treshold  5.49547962691577e-10"
```

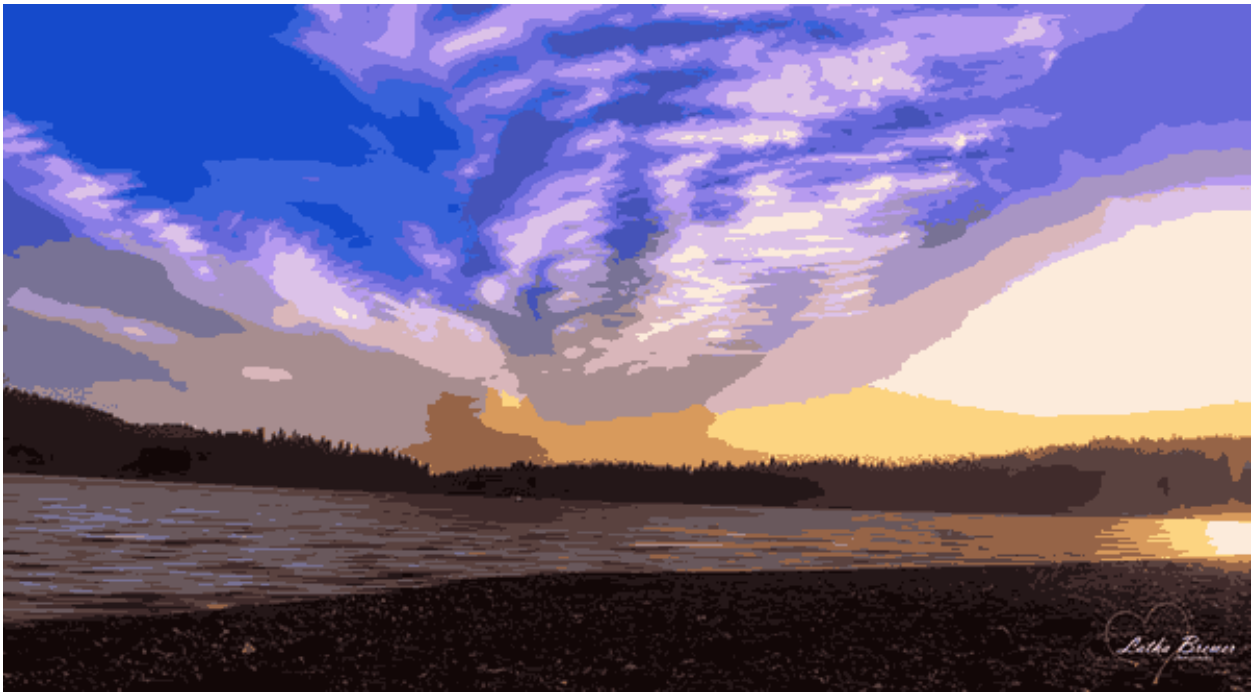


```
## [1] "Test Image 1 with segments =  20"
## [1] "Iteration #  180"
## [1] "Treshold  8.99767762945203e-05"
```



```
## [1] "Test Image 2 with segments =  20"
## [1] "Iteration #  125"
## [1] "Treshold  9.5056348403233e-05"
```

```
## [1] "Test Image 3 with segments =  20"
## [1] "Iteration #  41"
## [1] "Treshold  8.49310487183175e-05"
```



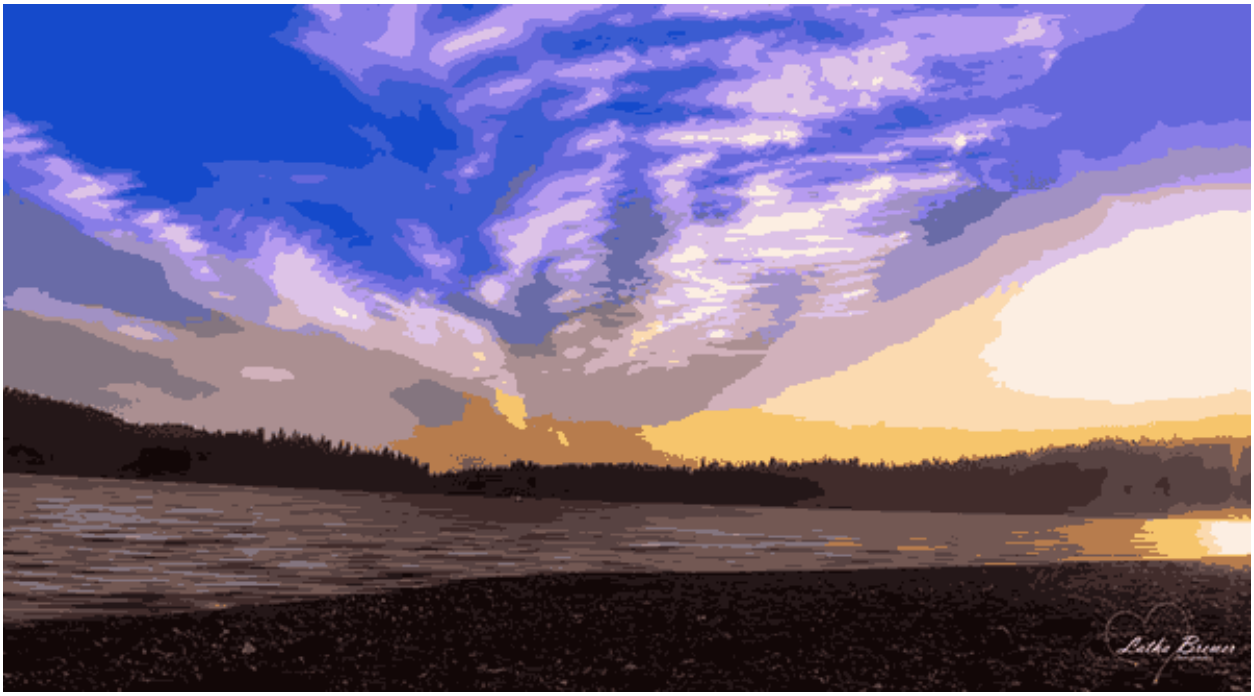
```
## [1] "Test Image 1 with segments =  50"
## [1] "Iteration #  719"
## [1] "Treshold  0.000997814557337051"
```

```
## [1] "Test Image 2 with segments =  50"
## [1] "Iteration #  927"
## [1] "Treshold  0.000904790760017704"
```

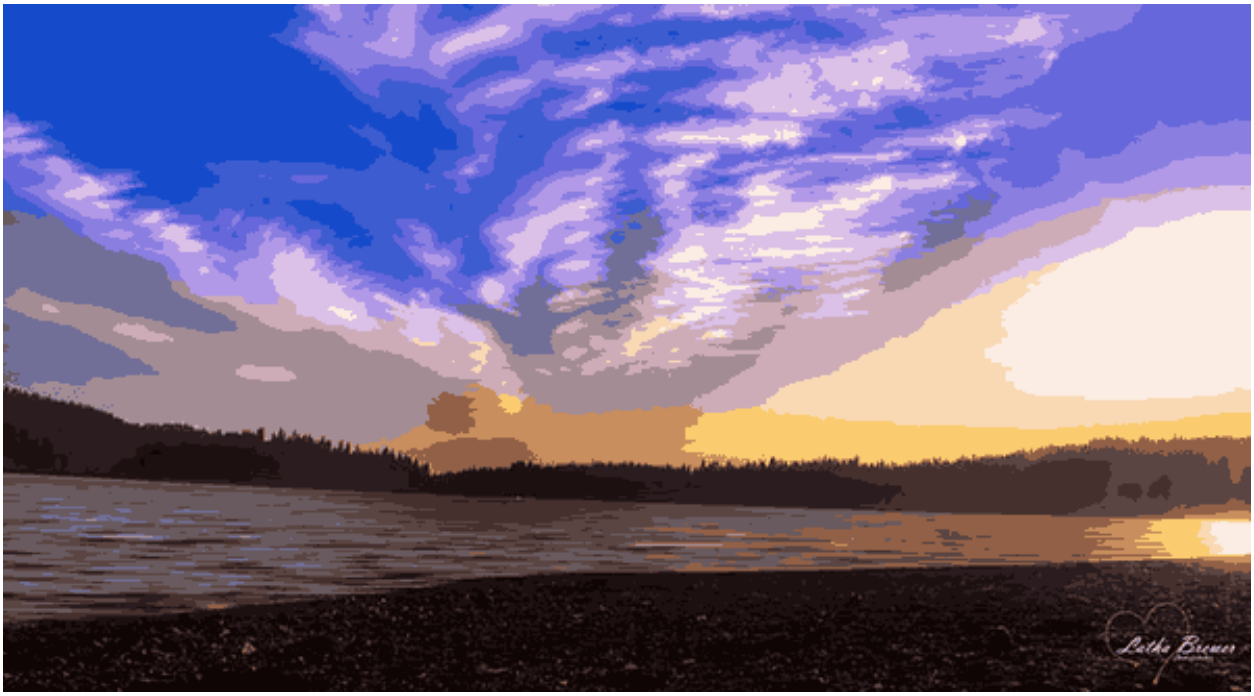
```
## [1] "Test Image 3 with segments =  50"
## [1] "Iteration #  621"
## [1] "Treshold  0.000946360435321481"
```

```
# 20 segments with 5 different starting points for the sunset test image
for (i in 1:5)
{
  sd = 452018 * i
  set.seed(sd)
  seg = 20
  print(paste("Test Image 3 with segments = 20, Starting point #",i))
  dispImg(df3, seg, im3)

}
```

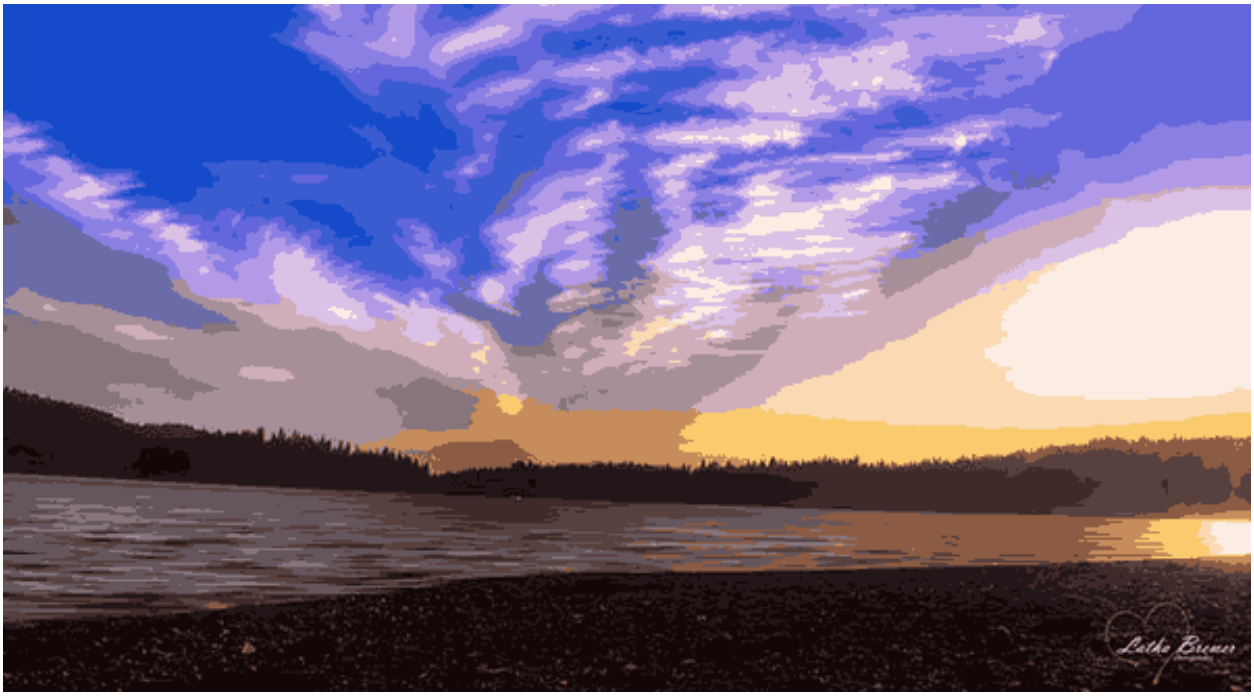
```
## [1] "Test Image 3 with segments = 20, Starting point # 1"
## [1] "Iteration #  52"
## [1] "Treshold  8.84808802226189e-05"
```



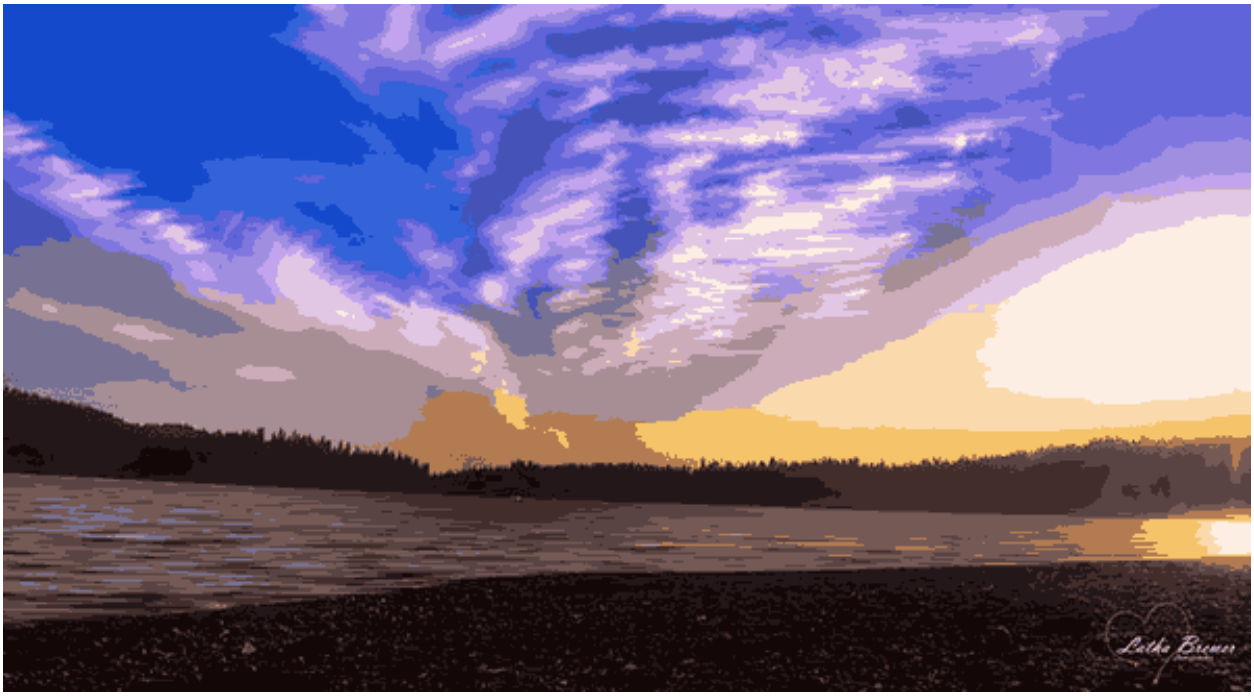
```
## [1] "Test Image 3 with segments = 20, Starting point # 2"
## [1] "Iteration #  40"
## [1] "Treshold  7.87777872801598e-05"
```



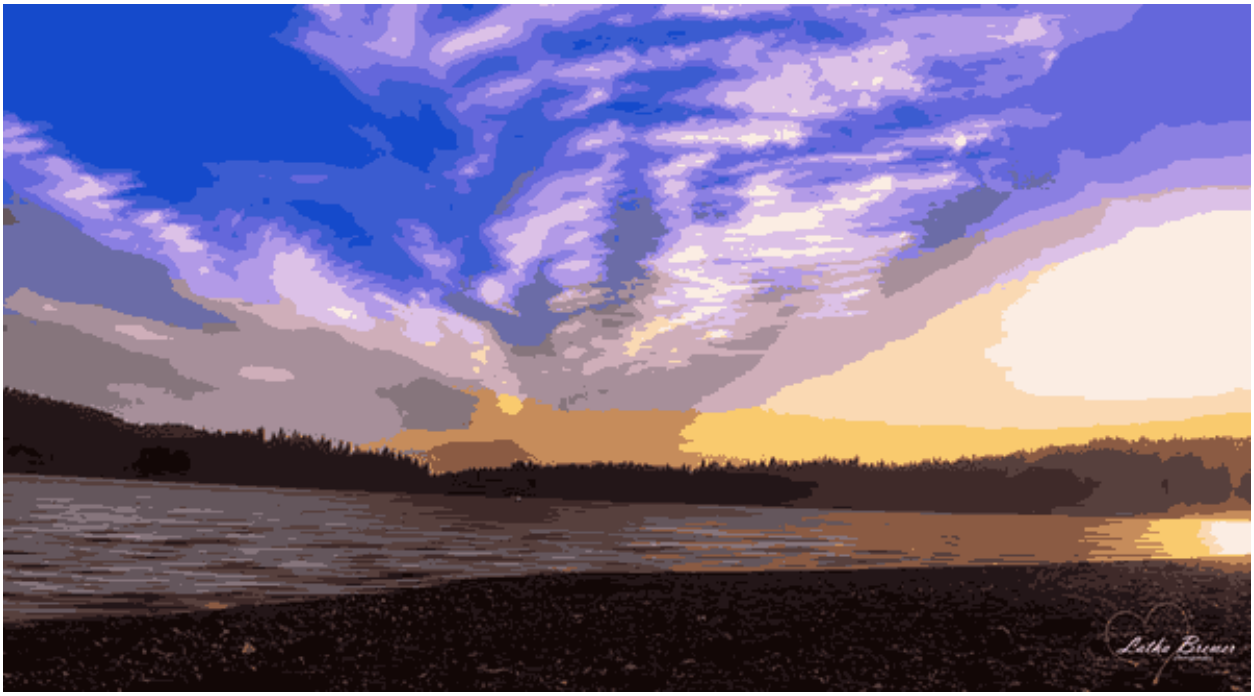
```
## [1] "Test Image 3 with segments = 20, Starting point # 3"
## [1] "Iteration #  44"
## [1] "Treshold  8.67313150649807e-05"
```



```
## [1] "Test Image 3 with segments = 20, Starting point # 4"
## [1] "Iteration #  54"
## [1] "Treshold  9.50861839328354e-05"
```



```
## [1] "Test Image 3 with segments = 20, Starting point # 5"
## [1] "Iteration #  63"
## [1] "Treshold  9.5349276421075e-05"
```



- From the above results, it can be seen that as the number of segments increases, the picture is more sharper and takes longer to converge.
- The different starting points of the sunset image looks a bit identical with very minimal differences, as this could be the result of them converging to a different local maxima.