20160040_assignment07

May 16, 2019

In [1]:
```
import PIL.Image as pilimg
import numpy as np
import matplotlib.pyplot as plt
import random
import copy

#read my image(pixel)
image = pilimg.open("C:\\Users\\recognize_data\\image.jpg")
#image = pilimg.open("C:\\Users\\recognize_data\\color.png")

#image pixel data to array
#315*420
#image_pixel[0][0] = [74 112 175]
image_pixel = np.array(image)
#image size
size_col = image.size[0]    # width of the image 420
size_row = image.size[1]    # height of the image 315
image_size = size_col * size_row

#time
t = 0
```

1 init x, y matrix

In [2]:
```
tempValue = 0;
x_pos =  np.empty((size_row, size_col), dtype = int)
y_pos = np.empty((size_row, size_col), dtype = int)
for i in range(0,size_row):
    for j in range(0,size_col):
        x_pos[i][j] = tempValue
    tempValue += 1
tempValue = 0;
for i in range(0,size_col):
    for j in range(0,size_row):
        y_pos[j][i] = tempValue
    tempValue += 1
```

```
In [3]: class AllProcess:
            def _init_(self, k, image_label, output_image, centroid,
                       count, energy, store_distance, time, spatial_val,
                       intensity_val,lamda):

                self.k = k
                self.image_label = image_label
                self.output_image = output_image
                self.centroid = centroid
                self.count = count
                self.energy = energy
                self.store_distance = store_distance
                self.time = time
                self.spatial_val = spatial_val
                self.intensity_val = intensity_val
                self.lamda =lamda

            def setdata(self,k):

                self.k = k
                self.centroid = np.zeros((k, 5), dtype = int) #x,y,r,g,b
                self.count = np.empty(k, dtype = int)
                self.store_distance = np.empty(k)
                self.image_label = np.empty((size_row, size_col), dtype = int)
                self.output_image = np.empty((size_row, size_col,3), dtype = int)
                self.energy = np.empty(1, dtype = float)
                self.time = 0
                self.lamda = 1
                self.spatial_val = np.empty(k)
                self.intensity_val = np.empty(k)

            def random_labeling(self):
                for i in range(0,size_row):
                    for j in range(0,size_col):
                        self.image_label[i][j] = random.randrange(0,self.k)

            def update_centroid(self):
                #initialize the centroid
                for i in range(0,self.k):
                    self.centroid[i][0] = 0
                    self.centroid[i][1] = 0
                    self.centroid[i][2] = 0
                    self.centroid[i][3] = 0
                    self.centroid[i][4] = 0
                    self.count[i] = 0

                for i in range(0,size_row):
                    for j in range(0,size_col):
```

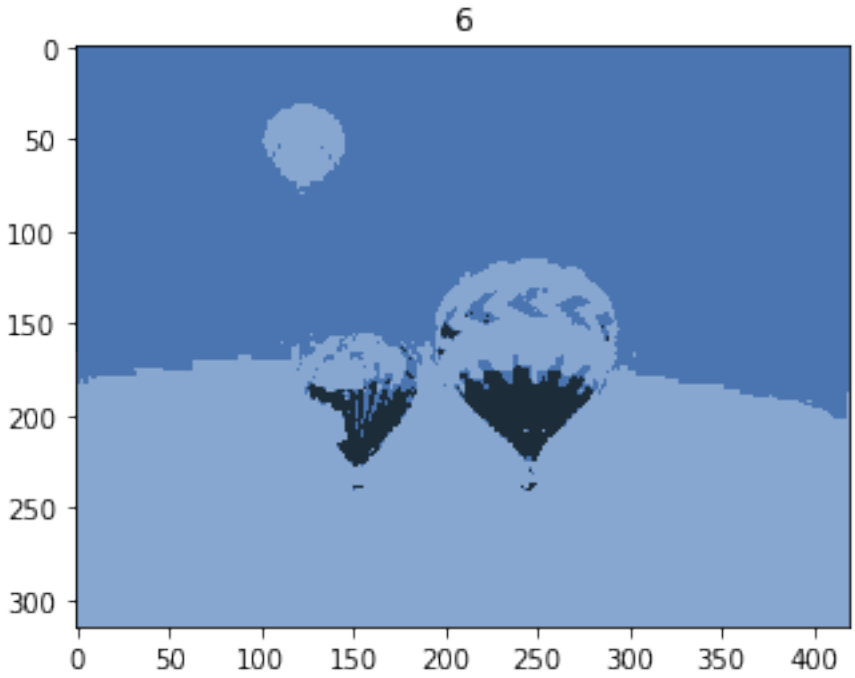
```
                a = self.image_label[i][j]
                self.centroid[a][0] += x_pos[i][j]
                self.centroid[a][1] += y_pos[i][j]
                self.count[a] += 1

        # average center
        for i in range(0,self.k):
            if(self.count[i]==0):
                self.count[i] = 1
            self.centroid[i][0] = self.centroid[i][0]/self.count[i]
            self.centroid[i][1] = self.centroid[i][1]/self.count[i]

        for i in range(0,self.k):
            self.count[i] = 0

            #RGB
        for i in range(0,size_row):
            for j in range(0,size_col):
                a = self.image_label[i][j]
                self.centroid[a][2] += image_pixel[i][j][0]#R
                self.centroid[a][3] += image_pixel[i][j][1]#G
                self.centroid[a][4] += image_pixel[i][j][2]#B
                self.count[a] += 1

            # average center
        for i in range(0,self.k):
            if(self.count[i]==0):
                self.count[i] = 1
            self.centroid[i][2] = self.centroid[i][2]/self.count[i]
            self.centroid[i][3] = self.centroid[i][3]/self.count[i]
            self.centroid[i][4] = self.centroid[i][4]/self.count[i]

        #distance2
    def distance(self, x,cx):
        d = (x-cx) ** 2
        return(d)

    #location(x,y)
    def spatial(self, x, y):
        for i in range(0,self.k):
            self.spatial_val[i] = self.distance(x,self.centroid[i][0])
            + self.distance( y,self.centroid[i][1])

    #color(r,g,b)
    def intensity(self, pixel):
        for i in range(0,self.k):
            self.intensity_val[i] = self.distance(pixel[0],self.centroid[i][2])
            + self.distance(pixel[1],self.centroid[i][3])
```

```
        + self.distance(pixel[2],self.centroid[i][4])
            def find_nearest(self, lamda, rgb, x, y):
    self.intensity(rgb)
    self.spatial(x,y)
    for i in range(0, self.k):
        self.store_distance[i] = self.intensity_val[i]
        + lamda * self.spatial_val[i]
    r = self.store_distance.argmin()
    return r

#update label
def update_label(self):
    for i in range(0,size_row):
        for j in range(0,size_col):
            self.image_label[i][j] = self.find_nearest(self.lamda,image_pixel[i][j]
                                                       x_pos[i][j], y_pos[i][j])

def energy_function(self, time):
    global energy
    energy_sum = 0
    for i in range(0,size_row):
        for j in range(0,size_col):
            a = self.image_label[i][j]
            energy_sum = (image_pixel[i][j][0] -self.centroid[a][2])**2
            +(image_pixel[i][j][1] -self.centroid[a][3])**2
            +(image_pixel[i][j][2] -self.centroid[a][4])**2
            +self.lamda *((x_pos[i][j]- self.centroid[a][0])**2
            +(y_pos[i][j]-self.centroid[a][1])**2)
    energy_sum = energy_sum
    if time == 0:
        self.energy = energy_sum
    else:
        self.energy = np.append(self.energy, energy_sum)

def plot_energy(self):
    plt.plot(self.energy,'ro')#red(r) dot(o)
    plt.xlabel('time')
    plt.ylabel('energy')
    plt.show()

def print_output(self,time):
    for i in range(0,size_row):
        for j in range(0,size_col):
            a = self.image_label[i][j]
            self.output_image[i][j][0] = self.centroid[a][2] #r
            self.output_image[i][j][1] = self.centroid[a][3] #g
            self.output_image[i][j][2] = self.centroid[a][4] #b
    plt.title(time)
```

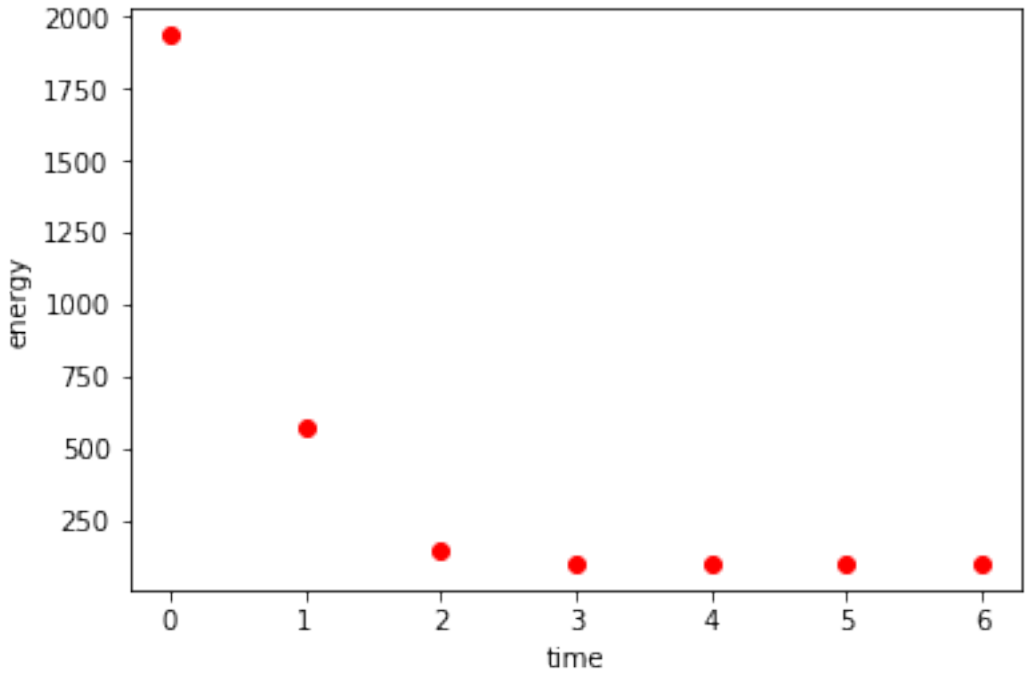
```
            plt.imshow(self.output_image)
```

2 k = 3, lamda = 0.1

```
In [4]: p1 = AllProcess()
        # k = 5
        p1.setdata(3)
        p1.random_labeling()
        p1.lamda = 0.1
        p1.update_centroid()
        p1.energy_function(p1.time)

        while True:

            p1.update_label()
            old_centroid = copy.deepcopy(p1.centroid)
            p1.update_centroid()
            p1.time += 1
        #     p1.print_output(p1.time)
        #     plt.plot(p1.centroid[:,1], p1.centroid[:,0], 'ro')
        #     plt.show()
            p1.energy_function(p1.time)

            if np.array_equal(old_centroid, p1.centroid):
                p1.print_output(p1.time)
                break
```



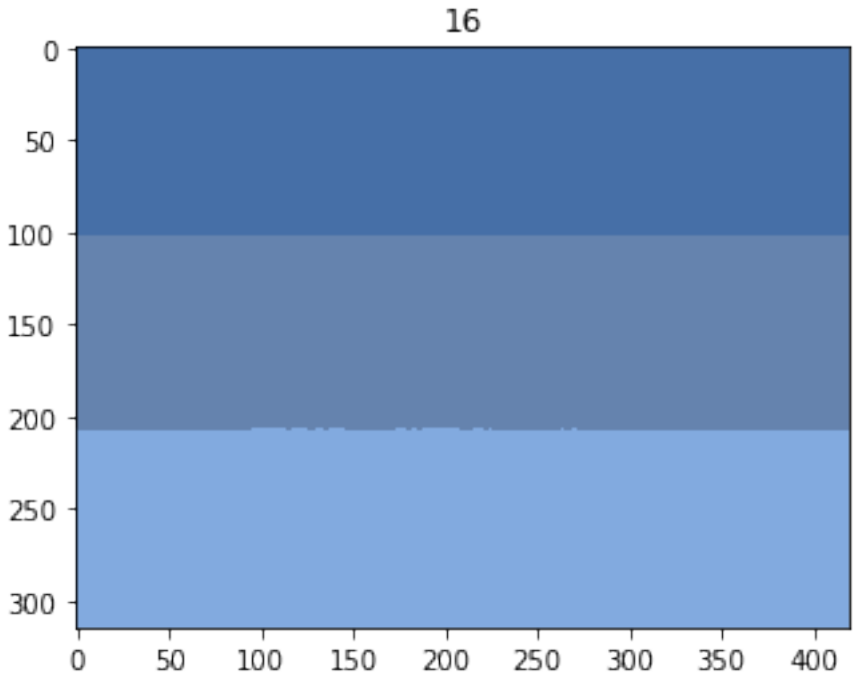
In [5]: p1.plot_energy()



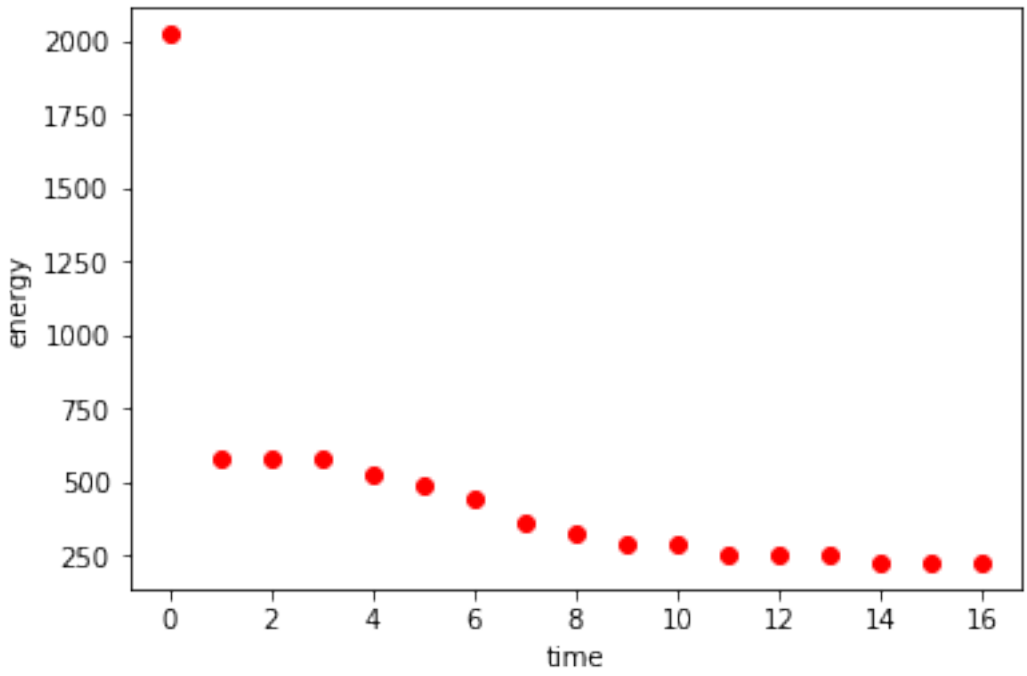
3 k = 3, lamda = 100

```
In [6]: p2 = AllProcess()
        # k = 5
        p2.setdata(3)
        p2.random_labeling()
        p2.lamda = 100
        p2.update_centroid()
        p2.energy_function(p2.time)

        while True:

            p2.update_label()
            old_centroid = copy.deepcopy(p2.centroid)
            p2.update_centroid()
            p2.time += 1
        #     p2.print_output(p2.time)
        #     plt.plot(p2.centroid[:,1], p2.centroid[:,0], 'ro')
        #     plt.show()
```

```
    p2.energy_function(p2.time)

    if np.array_equal(old_centroid, p2.centroid):
        p2.print_output(p2.time)
        break
```

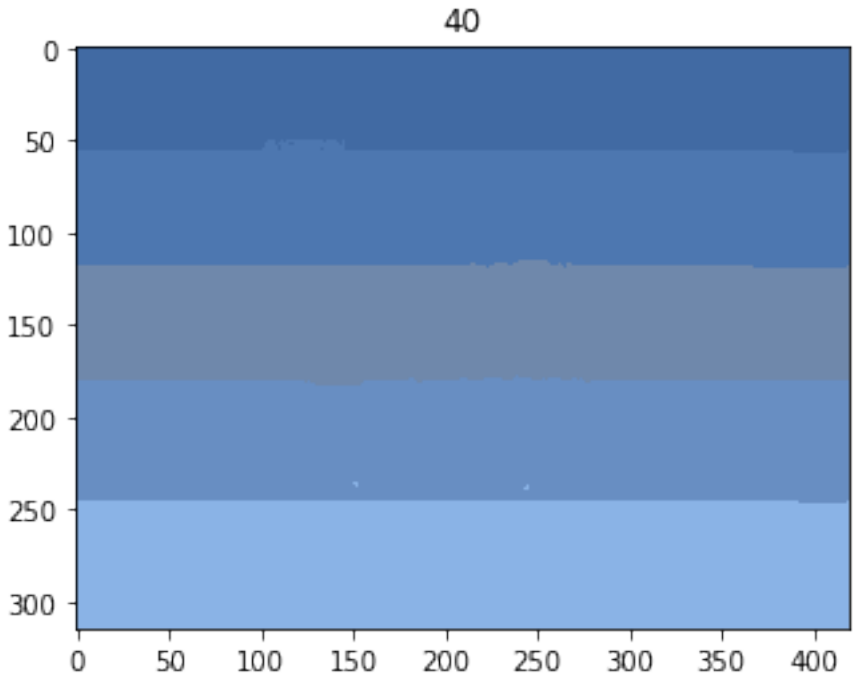


In [7]: p2.plot_energy()

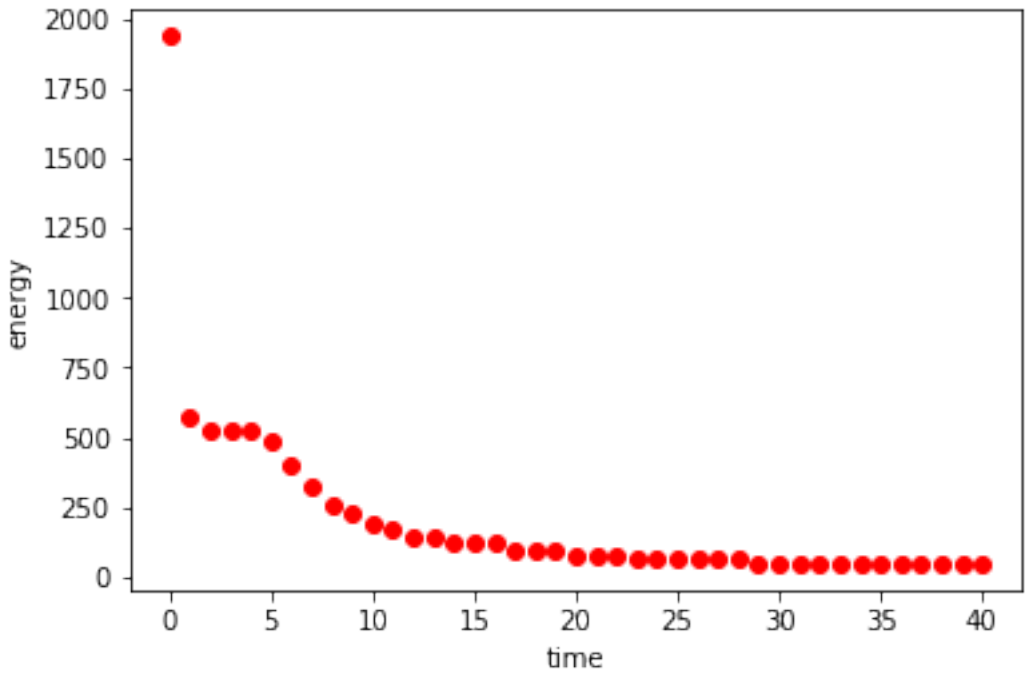


4 k = 5, lamda = 5

```
In [18]: p3 = AllProcess()
         # k = 5
         p3.setdata(5)
         p3.random_labeling()
         p3.lamda = 5
         p3.update_centroid()
         p3.energy_function(p3.time)

         while True:

             p3.update_label()
             old_centroid = copy.deepcopy(p3.centroid)
             p3.update_centroid()
             p3.time += 1
         #     p3.print_output(p3.time)
         #     plt.plot(p3.centroid[:,1], p3.centroid[:,0], 'ro')
         #     plt.show()
             p3.energy_function(p3.time)

             if np.array_equal(old_centroid, p3.centroid):
                 p3.print_output(p3.time)
                 break
```

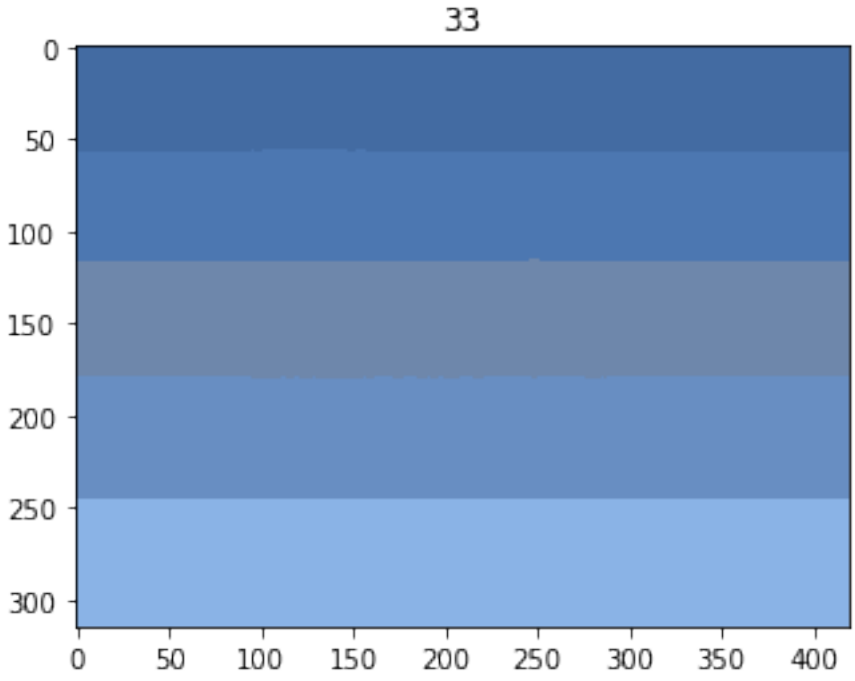


In [19]: p3.plot_energy()

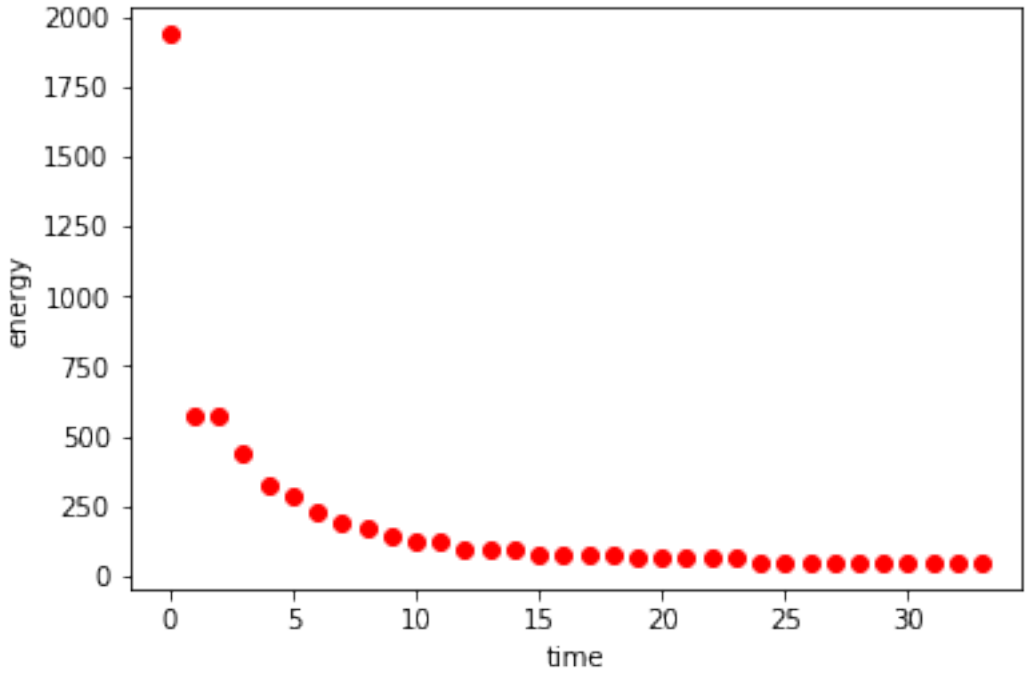


5 k = 5, lamda = 100

In [10]:
```
p4 = AllProcess()
# k = 5
p4.setdata(5)
p4.random_labeling()
p4.lamda = 100
p4.update_centroid()
p4.energy_function(p4.time)

while True:

    p4.update_label()
    old_centroid = copy.deepcopy(p4.centroid)
    p4.update_centroid()
    p4.time += 1
#     p4.print_output(p4.time)
#     plt.plot(p4.centroid[:,1], p4.centroid[:,0], 'ro')
#     plt.show()
    p4.energy_function(p4.time)

    if np.array_equal(old_centroid, p4.centroid):
        p4.print_output(p4.time)
        break
```

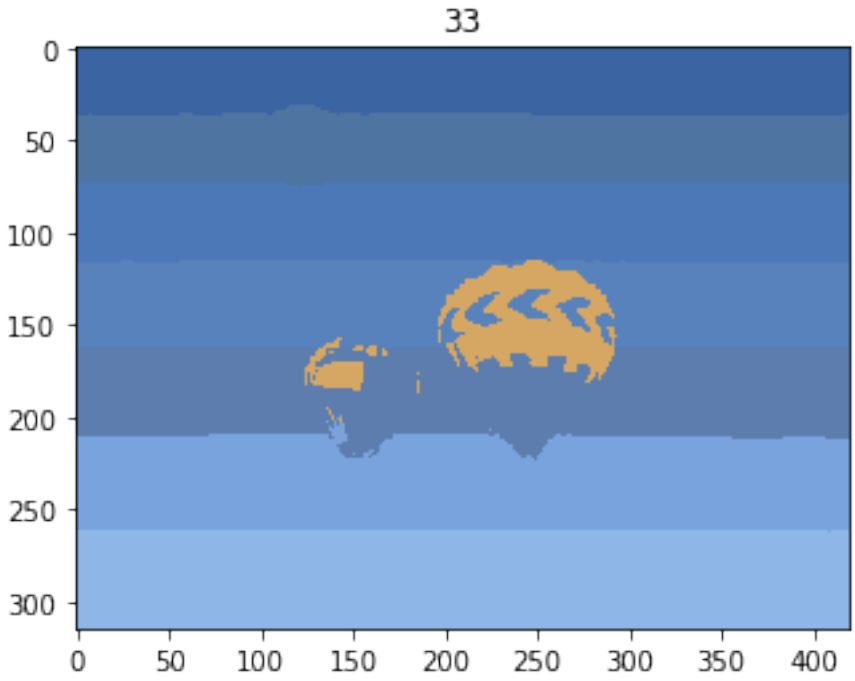


In [11]:
```
p4.plot_energy()
```

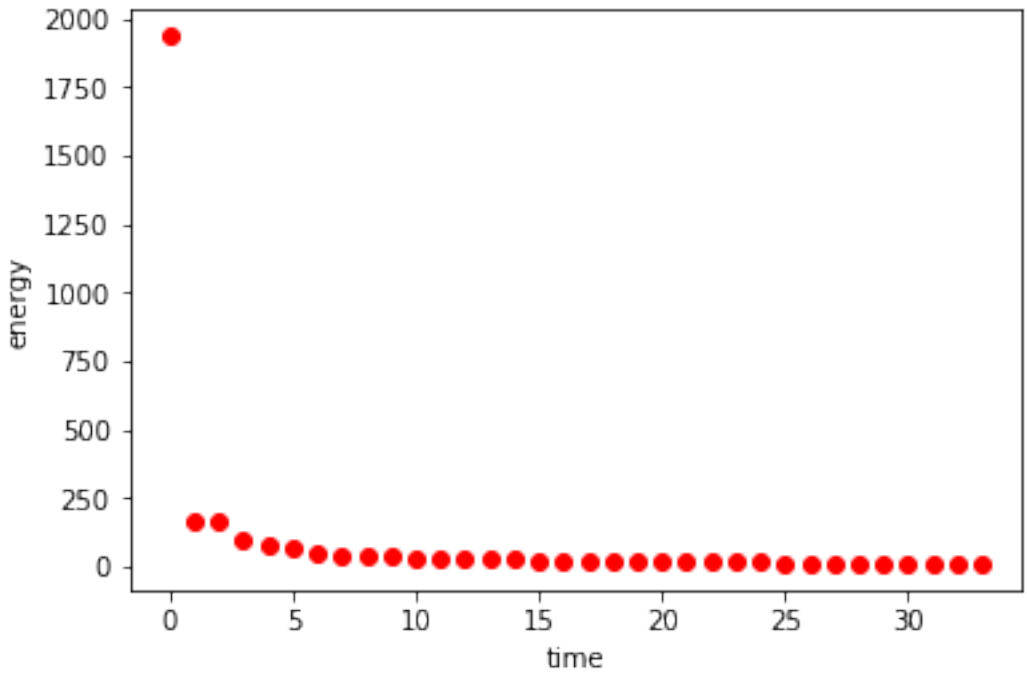


6 k = 10, lamda = 5

```
In [16]: p5 = AllProcess()
         # k = 5
         p5.setdata(10)
         p5.random_labeling()
         p5.lamda = 5
         p5.update_centroid()
         p5.energy_function(p5.time)

         while True:

             p5.update_label()
             old_centroid = copy.deepcopy(p5.centroid)
             p5.update_centroid()
             p5.time += 1
         #     p5.print_output(p5.time)
         #     plt.plot(p5.centroid[:,1], p5.centroid[:,0], 'ro')
         #     plt.show()
             p5.energy_function(p5.time)

             if np.array_equal(old_centroid, p5.centroid):
                 p5.print_output(p5.time)
                 break
```

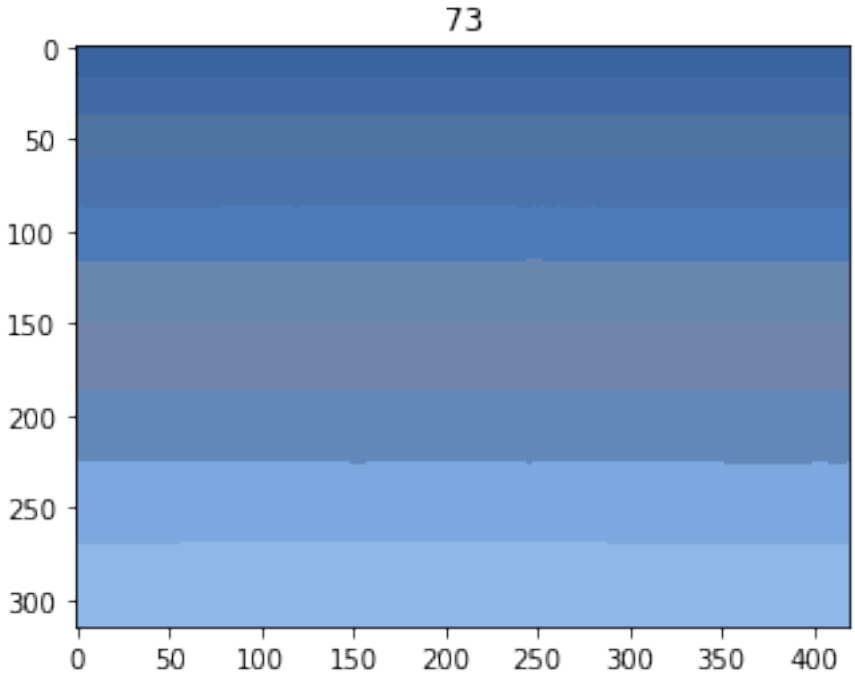


In [17]: p5.plot_energy()

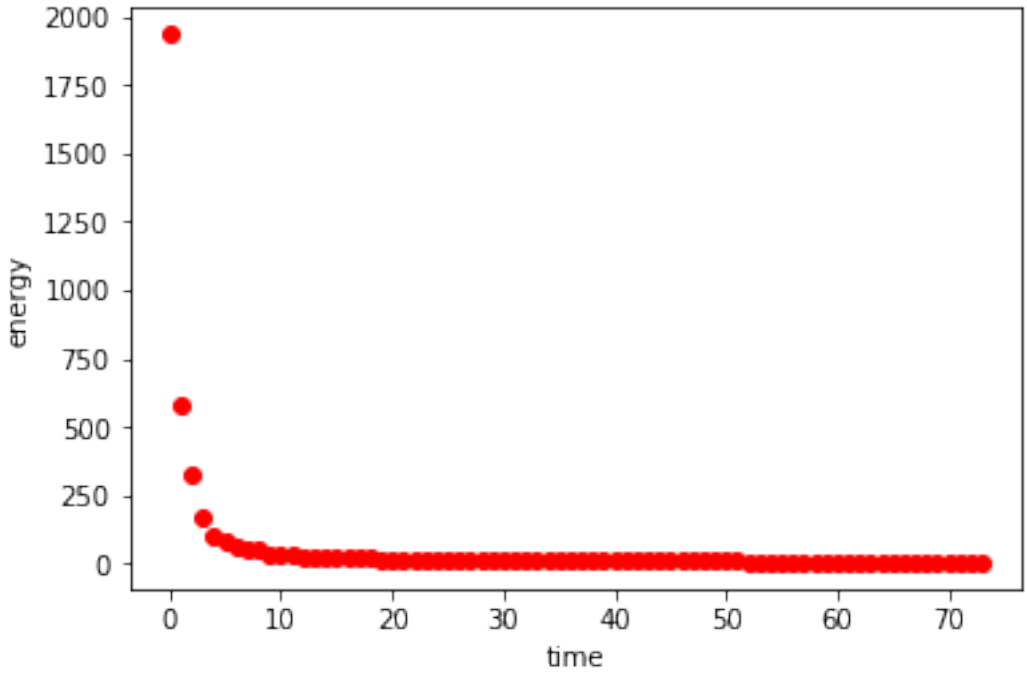


7 k = 10, lamda = 100

In [14]:
```
p6 = AllProcess()
# k = 5
p6.setdata(10)
p6.random_labeling()
p6.lamda = 100
p6.update_centroid()
p6.energy_function(p6.time)

while True:

    p6.update_label()
    old_centroid = copy.deepcopy(p6.centroid)
    p6.update_centroid()
    p6.time += 1
#     p6.print_output(p6.time)
#     plt.plot(p6.centroid[:,1], p6.centroid[:,0], 'ro')
#     plt.show()
    p6.energy_function(p6.time)

    if np.array_equal(old_centroid, p6.centroid):
        p6.print_output(p6.time)
        break
```



In [15]:
```
p6.plot_energy()
```


2000
1750
1500
1250
1000
750
500
250
0
energy
0
10
20
30
40
50
60
70
time