МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное агентство по образованию
«Пермский национальный исследовательский политехнический университет»
Кафедра микропроцессорных средств автоматизации

Отчёт по лабораторной работе №2
По дисциплине компьютерная графика
На тему: «Построение динамических изображений»

Работу выполнили
студенты гр. ИСУП-18-2м
_____ А. С. Морозов
_____ В. О. Раскошинский

Проверил доцент кафедры МСА
_____ Л. А. Мыльников

Пермь, 2020 г.

Алгоритм морфинга

Морфинг – технология в компьютерной графике, визуальный эффект, создающий впечатление плавной трансформации одного объекта в другой. Встречается в трёхмерной и двумерной (как растровой, так и в векторной) графике.

Особенностью технологии морфинга является необходимость установления соответствия между точками исходного и конечного изображений.

Морфинг растровых изображений

При работе с растровыми изображениями необходимо, чтобы разрешения исходного изображения и конечного совпадали. Тогда каждая точка исходного изображения становится в соответствие каждой точке конечного изображения, и работа алгоритма будет состоять в плавном преобразовании цвета по формуле:

$$c = c_1 + (c_2 - c_1) \cdot t, \tag{1}$$

где $0 < t < 1$ – фаза морфинга, c_1 и c_2 – цвета точки первого и второго изображений. Такая операция выполняется для каждой составляющей цвета в RGB или CMYK.

Реализация алгоритма морфинга для растровых изображений представлена в листинге 1. Результаты работы алгоритма представлены на рисунке .

```
1  from PIL import Image
2  import numpy as np
3
4  im1 = Image.open('1.png')
5  im2 = Image.open('2.png')
6
7  while True:
8      n = input('enter the number of intermediate stages\n')
9      try:
10         n = int(n)
11         break
12     except:
13         print('the number of intermediate stages is an integer')
14         continue
15
16 pix_1 = np.array(im1)
17 pix_2 = np.array(im2)
18 pix_0 = np.zeros((n,) + pix_1.shape, dtype = int)
19
20 m = np.linspace(0, 1, num = n + 2)[1:-1] # num = number of frames + 2
21
22 iint = np.vectorize(int) # vectorize int
23
24 # morphing for all intermediate stages
25 for i in range(len(m)):
26     pix_0[i] = iint(iint(pix_1) + (iint(pix_2) - iint(pix_1)) * m[i])
27
28 pix = np.hstack((pix_1, *pix_0, pix_2)).astype(np.uint8) # list of images
29 Image.fromarray(pix).show()
```

```
30 Image.fromarray(pix).save('gradient.png')
```

Листинг 1: Реализация алгоритма морфинга для растровых изображений

Рис. 1: Результаты работы алгоритма морфинга для растровых изображений

Морфинг векторных изображений

При использовании векторной графики устанавливается соответствие между точками исходного и целевого изображения. Таким образом получаем пары значений $\left(y_1^{(0)}, x_1^{(0)}\right)$, $\left(y_1^{(1)}, x_1^{(1)}\right)$ и т.д. в зависимости от количества промежуточных состояний и точек изображений. Для изменения значений координат обычно используется уравнение прямой ($y = a + b \cdot x$), но могут использоваться и другие фигуры, описываемые аналитически. Зная две координаты, мы можем определить значения коэффициентов a и b из системы уравнений

$$\begin{cases} y_1^{(0)} = a + b \cdot x_1^{(0)} \\ y_1^{(1)} = a + b \cdot x_1^{(1)} \end{cases},$$

$$b = \frac{y_1^{(1)} - y_1^{(0)}}{x_1^{(1)} - x_1^{(0)}},$$

$$a = y_1^{(0)} - x_1^{(0)} \cdot \frac{y_1^{(1)} - y_1^{(0)}}{x_1^{(1)} - x_1^{(0)}}.$$

Для генерирования промежуточных состояний необходимо определить число шагов Δ и величину одного шага для каждой из рассматриваемых точек:

$\delta_x = \frac{x_1^{(1)} - x_1^{(0)}}{\Delta}$ и $\delta_y = \frac{y_1^{(1)} - y_1^{(0)}}{\Delta}$. Новые значения вершин определяются по формулам: $x_1^{(0)} = x_1^{(0)} + \delta_x$ и $y_1^{(0)} = y_1^{(0)} + \delta_y$ до тех пор, пока $x_1^{(0)} < x_1^{(1)}$ и $y_1^{(0)} < y_1^{(1)}$.

Реализация алгоритма морфинга для векторных изображений представлена в листинге 2. Результаты работы алгоритма представлены на рисунках 2.

```
import numpy as np
from PIL import Image, ImageDraw

k = 20
# first frame coordinates
x_1  = np.array((
    8*k,        # 0 point left shoulder
    8*k + 3*k,  # 1 point neck
    8*k + 6*k,  # 2 point right shoulder
    8*k + 2*k,  # 3 point left hip
    8*k + 3*k,  # 4 point
    8*k + 4*k,  # 5 point right hip
    8*k,        # 6 point left hand
    8*k + 6*k,  # 7 point right hand
    8*k + 2*k,  # 8 point left leg
    8*k + 4*k   # 9 point right leg
))
y_1 = np.array((
    8*k,        # 0
    8*k,        # 1
    8*k,        # 2
    8*k + 6*k,  # 3
    8*k + 6*k,  # 4
    8*k + 6*k,  # 5
    8*k + 7*k,  # 6
    8*k + 7*k,  # 7
    8*k + 13*k, # 8
    8*k + 13*k  # 9
))
# The second frame
x_2 = x_1.copy()
y_2 = y_1.copy()
x_2[6] = 8*k - 3*k # move left hand
y_2[6] = 8*k - 3*k # move left hand
# x_2[9] = 8*k + 6*k # move right leg
# The third frame
x_3 = x_2.copy()
y_3 = y_2.copy()
x_3[7] = 8*k + 9*k # move right hand
y_3[7] = 8*k - 3*k # move right hand
# x_3[9] = 8*k + 8*k # move right leg
# The fourth frame
x_4 = x_3.copy()
y_4 = y_3.copy()
x_4[6] = 8*k - 5*k # move both hands
x_4[7] = 8*k + 11*k # move both hands
y_4[6] = 8*k # move both hands
y_4[7] = 8*k # move both hands
# x_4[9] = 8*k + 6*k # move right leg

# all frames
x_set = np.array((x_1, x_2, x_3, x_4, x_1)).T
y_set = np.array((y_1, y_2, y_3, y_4, y_1)).T
```

```
54
55  x = []
56  xx = []
57  y = []
58  yy = []
59
60  for i in range(len(x_set)):
61      for j in range(len(x_set[i]) - 1):
62          x += np.linspace(x_set[i, j], x_set[i, j + 1], num=5, endpoint=False).tolist()
63          y += np.linspace(y_set[i, j], y_set[i, j + 1], num=5, endpoint=False).tolist()
64      else:
65          x += [x_set[i][-1]]
66          y += [y_set[i][-1]]
67      xx.append(x)
68      yy.append(y)
69      x = []
70      y = []
71
72  # connecting dots in a line
73  line_seq = [
74      [0, 1], #zero with first
75      [1, 2], # first with second etc
76      [0, 6], [1, 4], [2, 7], [3, 4], [4, 5], [3, 8], [5, 9]]
77  gif = []
78  for i in range(len(xx[0])):
79      l = []
80      for d in line_seq:
81          l.append(np.array(([xx[d[0]][i], yy[d[0]][i]], [xx[d[1]][i], yy[d[1]][i]])))
82      # head
83      ellip = [(10*k, 5*k), (12*k, 7*k)]
84      # empty white frame
85      img = Image.new('RGB', (22*k, 29*k), 'white')
86      img1 = ImageDraw.Draw(img)
87      # drawing head
88      img1.ellipse(ellip, fill = "white", outline = "red", width = 3)
89      # drawing all lines
90      for j in l:
91          img1.line(list(map(tuple, j)), fill = "red", width = 3)
92      # add frame to the list
93      gif.append(img)
94      # save the frame
95      img.save('frames0/frame' + str(i) + '.png')
96  # save the gif
97  gif[0].save(
98      'dance.gif', format = 'GIF', append_images = gif[1:],
99      save_all=True, duration=120, loop=0
100 )
```

Листинг 2: Реализация алгоритма морфинга для векторных изображений

Весь код и результаты его работы представлены в открытом репозитории.

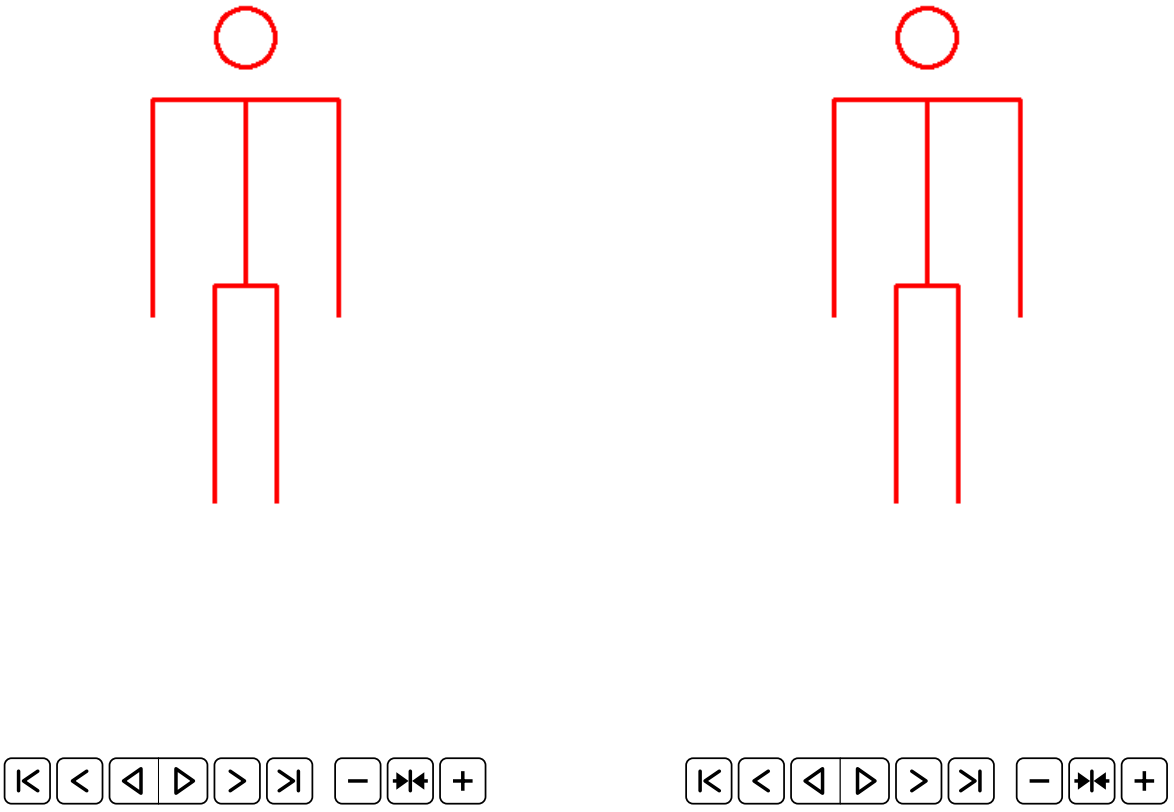

Рис. 2: Результаты работы алгоритма морфинга для изображений