Introduction to Artificial Intelligence Assignment 2

What is the representation in your algorithm?

In my algorithm I represent a solution as a 2D array of 3-integer tuples. Such array has the same size as input image's resolution in pixels, and each item represents a color encoded by 3 integers. The image is being composed of line segments (strokes) of variable length and thickness. Such stokes either oriented vertically, horizontally, or diagonally. Furthermore, color space can be set either to full or grayscale. All of those parameters (*length*, *thickness*, *color* [space], *orientation*, and *population* or rounds of mutation) are specified in *cnf.json* file.

```
{
    "clr": true,
    "axi": false,
    "dgn": true,
    "thk": [
        2,
        2
    ],
    "lng": [
        4,
        12
    ],
    "ppl": 5
}
```

Picture 1. cnf.json

Upon application startup the .json file is being deserealized into *Cnf* class. Such class is later used to generate random parameters for a stroke.

```python
class Cnf:
    ppl = 10
    clr = False
    rtn = [-.25, 25]
    thk = [1, 2]
    lng = [4, 12]

    def __init__(self, J):
        cnf = json.loads(J)
        self.ppl = cnf["ppl"]
        self.clr = cnf["clr"]

        self.rtn = []
        if cnf["axi"]:
            self.rtn += [0, 2]
        if cnf["dgn"]:
            self.rtn += [1, 3]

        self.thk = cnf["thk"]
        self.lng = cnf["lng"]

    def var(self):
        if self.clr:
            C = [np.random.randint(0, 256) for _ in range(3)]
        else:
            C = [np.random.randint(0, 256)] * 3

        T = np.random.randint(self.thk[0], self.thk[1] + 1)
        L = np.random.randint(self.lng[0], self.lng[1] + 1)
        R = np.random.choice(self.rtn)

        return C, T, L, R
```

Picture 2. Class Cnf

Which selection mechanism is being used?

Every generation starts with a single solution. The very first generation starts with a solid white image of the same size as the input one.

```
88   # source image
89   src = cv2.imread("src.png")
90   # image size
91   H, W = src.shape[:2]
92   # configuration
93   cnf = Cnf(open("cnf.json", "r").read())
94
95
96   rsl = Ind(255 * np.ones((H, W, 3)))
97   rsl.fit = fit(rsl.img, 1, 0, 0, H, W)
```

Picture 3. Initialization

Then there is N rounds of mutation. For each round to the initial image a random stroke is being added. This process yields N images produced by adding a random stroke to the initial one. An image with the best fit is being passed to the next generation. The cycle repeats, with current [best] representation and its fitness being save & printed every 100th generation for assessment.

```
100  gnr = 0
101  while True:
102      bst = rsl
103      for _ in range(cnf.ppl):
104          ind = mut(rsl.cln())
105          if ind.fit < bst.fit:
106              bst = ind.cln()
107
108      rsl = bst
109      gnr += 1
110
111      if not gnr % 100:
112          print(gnr, ":", rsl.fit)
113          cv2.imwrite("rsl.png", rsl.img)
114
```

Picture 4. Selection loop

Which image manipulation techniques have you applied?

For painting stokes OpenCV-Python has been used. First, as mentioned in the first section, I generate random parameters based on the initial configuration. After that, a random point is being selected on the square grid with cell size of T, where T is generated stroke thickness. This

grid snapping allows the strokes of the same size not to overlap which improves aesthetics of the output, bringing it closer to the cotton pattern, where lines align evenly, forming a proper grid. And then, based on the generated position and rotation, and line is being generated, by calculating 2 points – ends of a segment.

```
63  def mut(ind):
64      C, T, L, R = cnf.var()
65
66      y, x = [T // 2 + T * np.random.randint(0, i // T) for i in [H, W]]
67      b, a, d, c = lin(y, x, R, L)
68
69      ind.fit -= fit(ind.img, T, b, a, d, c)
70      cv2.line(ind.img, (a, b), (c, d), C, T)
71      ind.fit += fit(ind.img, T, b, a, d, c)
72
73      return ind
```

Picture 5. Mutation

```
52  def lin(y, x, R, L):
53      if not R:
54          return y, x - L, y, x + L
55      elif R == 2:
56          return y - L, x,  y + L, x
57      elif R == 1:
58          return y - L, x - L, y + L, x + L
59      else:
60          return y + L, x - L, y - L, x + L
```

Picture 6. Line points generation

What is the fitness function?

After all parameters are ready, we calculate fitness over the bounding box of the line, subtract it, draw line, recalculate fitness over the same box, and add it. If such manipulation resulted in decrease of the representation's fitness, the picture has been improved. As mentioned before, we pick representation with the lowest fitness and proceed to the next iteration with it. The fitness function is merely sum of absolute values of difference of pixel colors (for all 3 channels) between representation and source images over a selected box. The lower it is – the better.

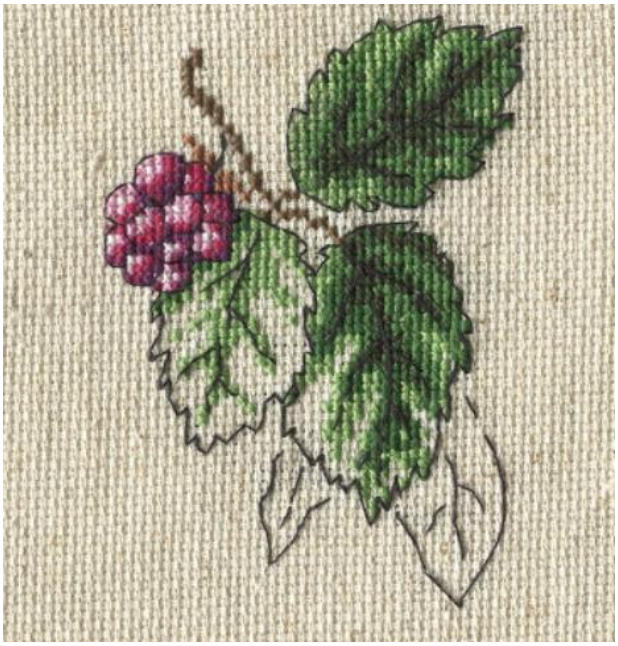
```
76  def fit(img, T, b, a, d, c):
77      b, d = np.sort(np.clip([b, d], 0, H))
78      if b == d:
79          d += T
80
81      a, c = np.sort(np.clip([a, c], 0, W))
82      if a == c:
83          c += T
84
85      return np.sum(np.absolute(np.subtract(img[b:d, a:c], src[b:d, a:c])))
```

Picture 7. Fitness function

Why do you consider the output image as a piece of art?

Art to me is not something I could define very precisely, but something I could feel when I am exposed to. When I say “feel” I mean some positive feeling, but it is not necessarily fun or joy. It can be even grief, so sadness, or feeling of being lost. But what I mean – some feeling from which I ultimately benefit.

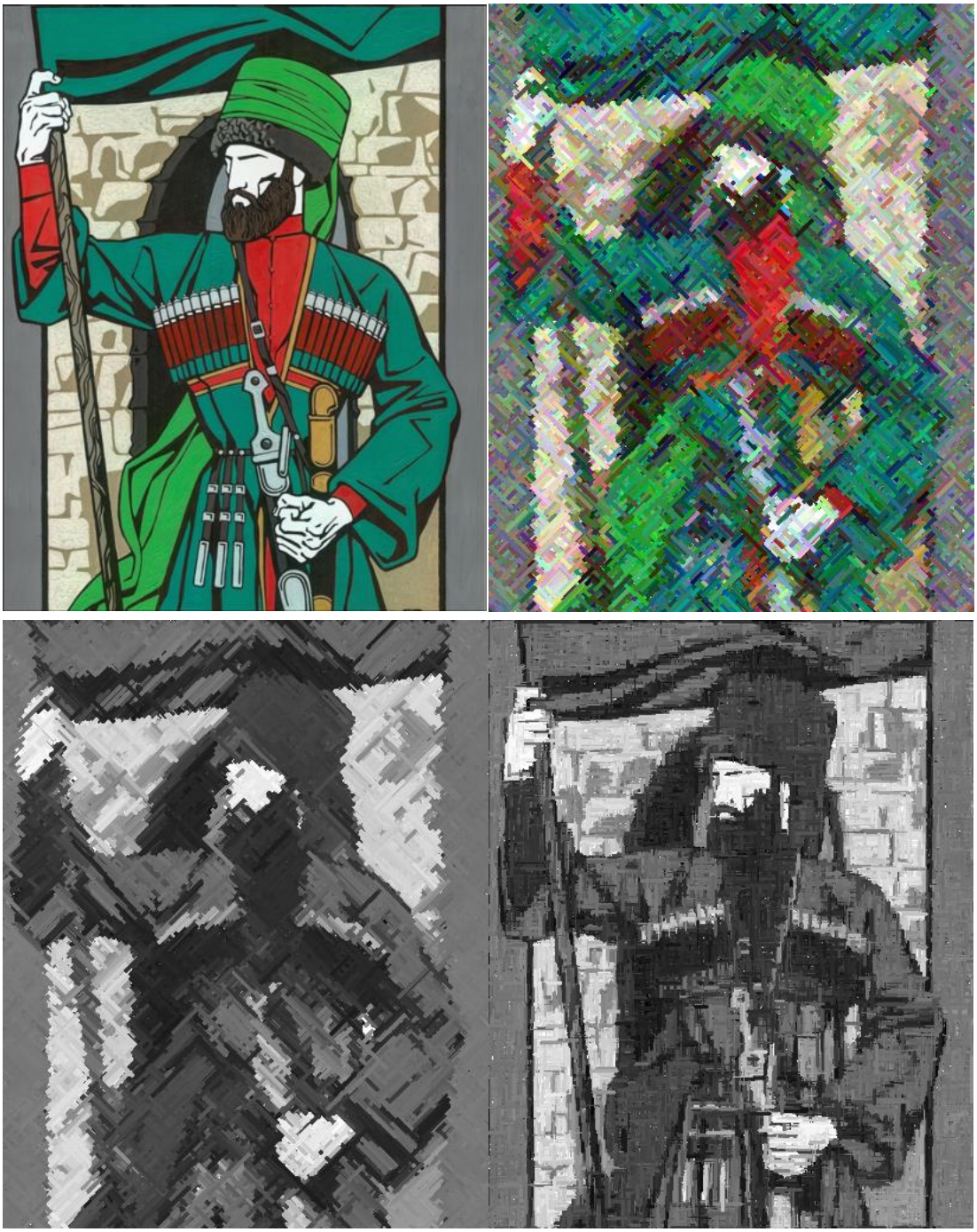
Like I briefly mentioned above, with my algorithm I seek to achieve resemblance of cotton fabric, with evenly aligned grid of threads. I like very much when I see some interwoven picture in clothes or other pieces of fabric, and the output the algorithm produces makes be remember those embroideries on fabric.



Picture 8. Embroidery on fabric

Outcomes

Below original image and 3 various outputs depending on configuration can be seen.



Picture 9. Source image and 3 generated ones

To run code, you have to install Python 3.8 (though not tested, any 3.x should work) as well as Numpy and OpenCV packages through pip. Valid *cnf.json and src.png* representing configuration parameters and input image respectively should be present in the same folder with *gnt.py*. The numerical arrays in *cnf.*json represent min-max range of possible values for corresponding parameters. Feel free to adjust them to achieve different outcomes. After all preparation have been completed, simply run *gnt.py*. Resulted image will be generated as *rsl.png*.

Magomed Magomedov, BS18-05
27 April, 2020