**Lab 07**

|  |  |
| --- | --- |
| Name: | 劉邦彬 |
| Student ID: | R11631033 |
| Total Score: |  |

**Note:**

Most of the explanations in this lab is optional. However, giving reasonable explanations to your answer or programs will earn you partial credits when your answer is incorrect.

1. **Multiple Choice (20 points, 5 points each question)**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Answer | Explanation (Optional) | Score |
| 1 | (d) |  |  |
| 2 | (a) |  |  |
| 3 | (a) |  |  |
| 4 | (c) |  |  |

1. **Basic Image Processing (60 points, 20 points each question)**

|  |  |  |
| --- | --- | --- |
| # | Explanation (Optional) | Score |
| 1 | def fade\_gradually(img):      processed = img.copy()      # TODO\_B1      height, width, channels = processed.shape      for i in range(width):          for j in range(height):              gray = (                  1 / 3 \* processed[j, i, 0]                  + 1 / 3 \* processed[j, i, 1]                  + 1 / 3 \* processed[j, i, 2]              )              processed[j, i, :] = (1 - i / width) \* processed[j, i, :] + (                  i / width              ) \* gray      return processed  Result: |  |
| 2 | def image\_matting(img):      processed = img.copy()      # TODO\_B2      gray = skimage.color.rgb2gray(processed)      foreground = gray > 0.005      height, width, channels = processed.shape      processed = np.zeros((height, width, 4), dtype=np.uint8)      processed[foreground, :3] = img[foreground, :3]      processed[foreground, 3] = 255      return processed  Result: |  |

1. **Implementation of Image Processing Algorithm (40 points, 20 points each question)**

|  |  |  |
| --- | --- | --- |
| # | Explanation (Optional) | Score |
| 1 | def my\_resize(img, height, width):      # TODO\_C1      new\_h, new\_w = np.meshgrid(np.arange(height), np.arange(width), indexing="ij")      new\_h = new\_h \* img.shape[0] / height      new\_w = new\_w \* img.shape[1] / width      h\_int = (new\_h).astype(int)      w\_int = (new\_w).astype(int)      u = new\_h - h\_int      v = new\_w - w\_int      # Prevent crossing      h\_int = np.clip(h\_int, 0, img.shape[0] - 2)      w\_int = np.clip(w\_int, 0, img.shape[1] - 2)      top\_left = img[h\_int, w\_int]      top\_right = img[h\_int, w\_int + 1]      down\_left = img[h\_int + 1, w\_int]      down\_right = img[h\_int + 1, w\_int + 1]      new\_img = (          (1 - u) \* (1 - v) \* top\_left          + (1 - v) \* u \* down\_left          + v \* (1 - u) \* top\_right          + u \* v \* down\_right      )      return new\_img  Result: |  |
| 2 | def my\_rotation(img, angle):      # TODO\_C2      theta = np.radians(angle)      new\_height = int(np.clip(img.shape[0] \* np.sqrt(2), 0, img.shape[0]))      new\_width = int(np.clip(img.shape[1] \* np.sqrt(2), 0, img.shape[1]))      rotate\_img = np.zeros((new\_height, new\_width))      for i in range(new\_height):          for j in range(new\_width):              y = (                  (i - new\_height / 2) \* np.cos(theta)                  + (j - new\_width / 2) \* np.sin(theta)                  + img.shape[0] / 2              )              x = (                  -(i - new\_height / 2) \* np.sin(theta)                  + (j - new\_width / 2) \* np.cos(theta)                  + img.shape[1] / 2              )              if 0 <= y < img.shape[0] and 0 <= x < img.shape[1]:                  rotate\_img[i, j] = img[int(y), int(x)]      return rotate\_img  Result: |  |