Homework 2

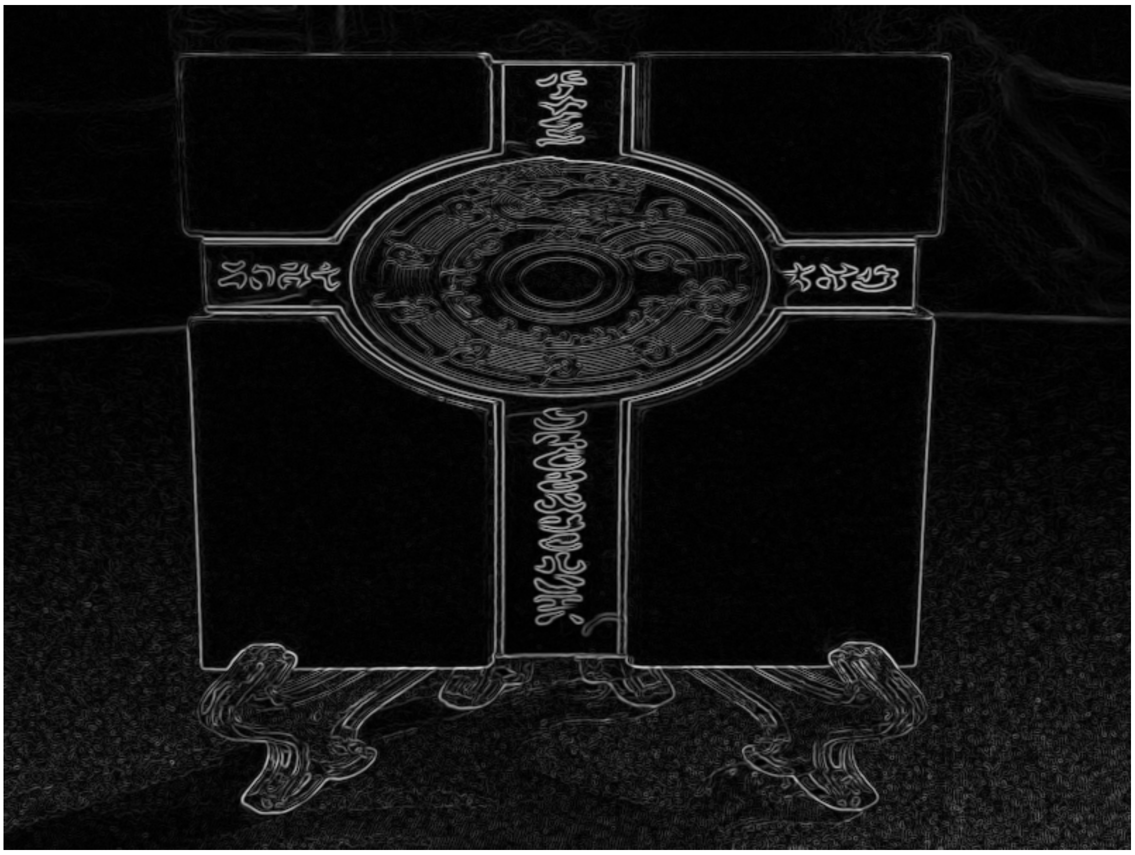
(p) Problem 1

- *Sample1*

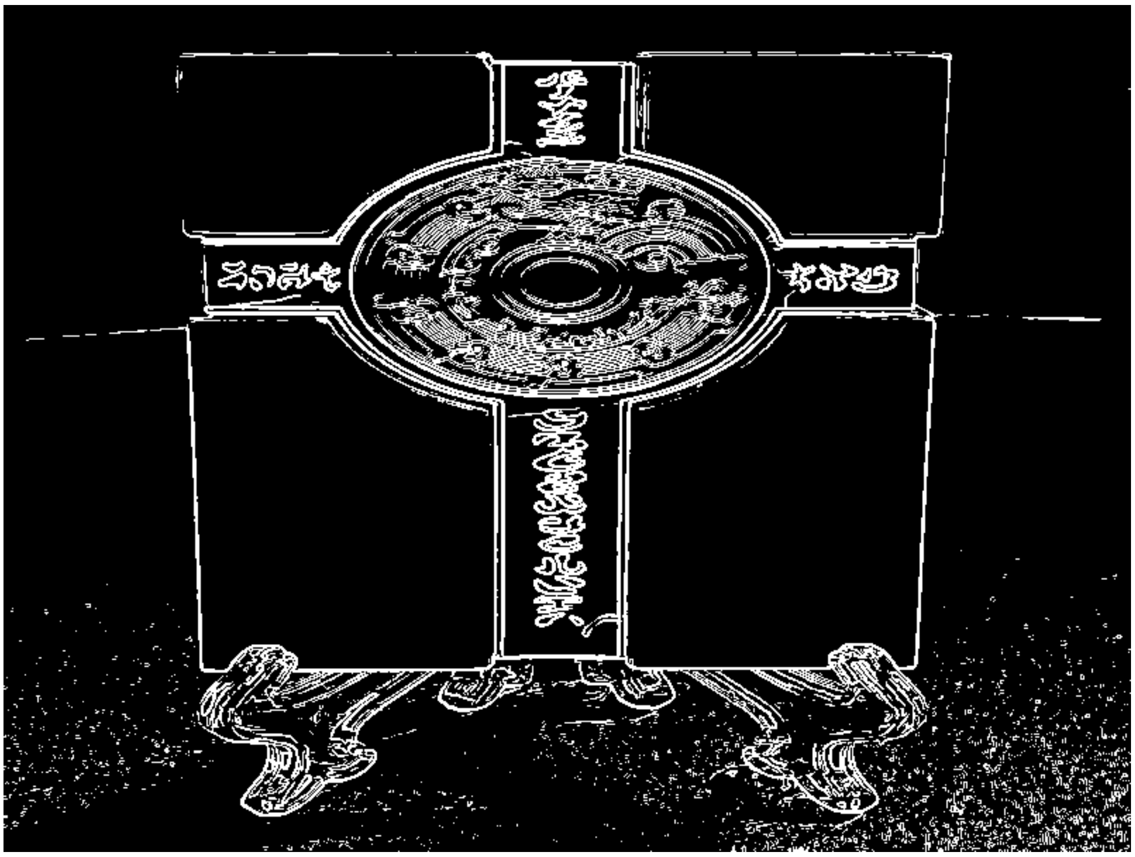
---

a

- *Motivation and Approach*
  - *Simply follow the Sobel filtering fomula in the lecture slides.*
- *result1*



- *result2*

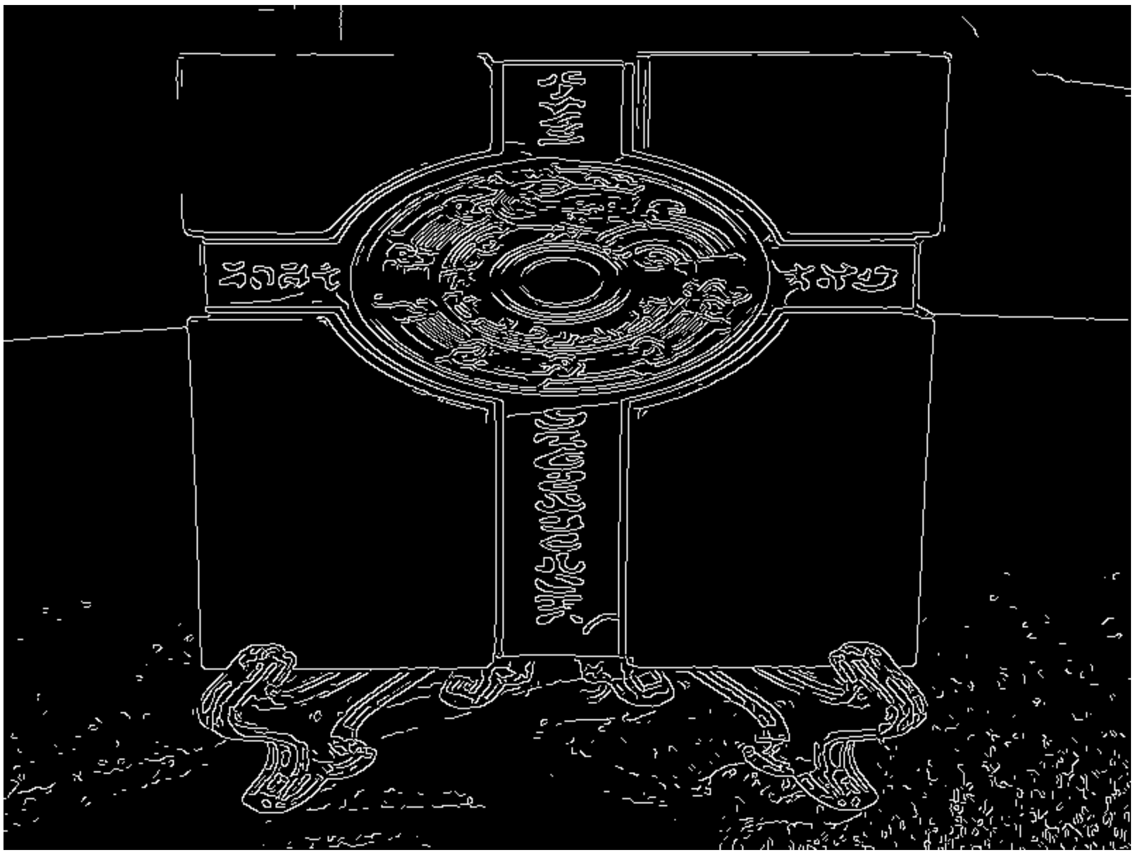


- *Discussion*
  - *The parameters of Sobel filter is fixed.*
  - *I choose (mean(gradient) + std(gradient)) as threshold since it preserves most of the edge with just a little noise at the bottom of result2.*

---

b

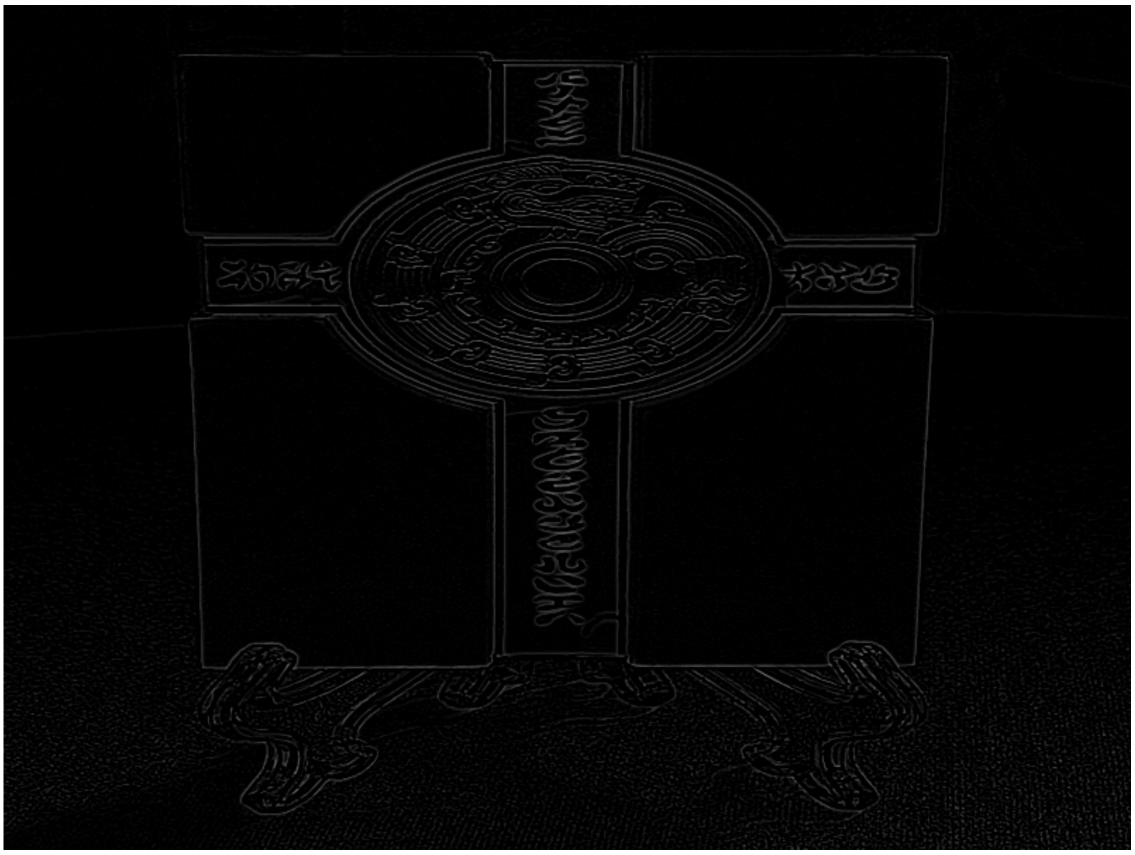
- *Motivation and Approach*
  - *Simply follow the Canny filtering fomula in the lecture slides*
- *result3*



- *Discussion*
  - *In my inplementation, I seperate five steps into five function*
  - *Noise reduction*
    - *Using Gaussion filter with $window = 5$ and $\sigma^2 = 1$*
  - *Compute gradient magnitude and orientation*
    - *I have tried several different G_R and G_C masks but the results don't change a lot.*
  - *Non-maximal suppression*
    - *Simply follow the fomula in lecture slide and the result from the last step*
  - *Hysteretic thresholding*
    - *Let T store the pixel value from last step without zero pixel values*
    - *To make the result more obvious, I set $T_L = mean(T)$ and $T_H = mean(T) + 0.5(std(T))$*
  - *Connected component labeling method*
    - *Simply follow the fomula in the lecture slides and the result from the last step*

C

- *Motivation and Approach*
  - *I use the Laplaican of Gaussian mask obtained online.*
    - $LoG(x, y) = -\frac{1}{\pi\sigma^4}[1 - \frac{x^2+y^2}{2\sigma^2}]e^{-\frac{x^2+y^2}{2\sigma^2}}$
    - *With $window = 5$, and $\sigma^2 = 0.5$*
- *result4*

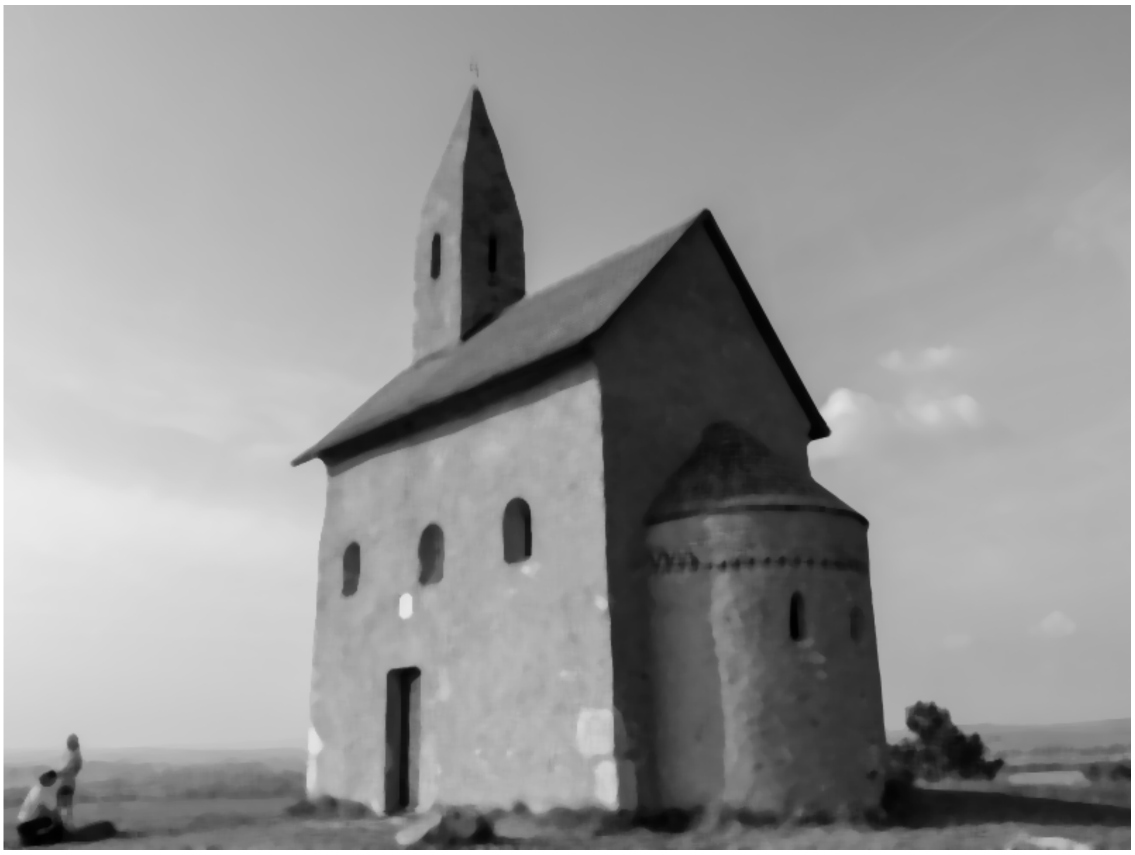


- *Discssion*
  - *Sobel*
    - *Result 2 contains the most noises. Although I have tried to remove the noises, some edges have been eliminated at the same time.*
  - *Canny*
    - *Sor far, result3 is the most attractive to me. The edges in result3 are very clear and with less noise simultaneously. However, it also costs the longest time to finish the edge detection.*
  - *Laplaican of Gaussian*
    - *Obviously, result4 is the darkest among three edge detection results*
    - *Apparently, it's because there are several negative values within the mask according to fomula in the motivation part*
    - *Though I can enhance the light to make the result more appealing, I decide to keep the original output*

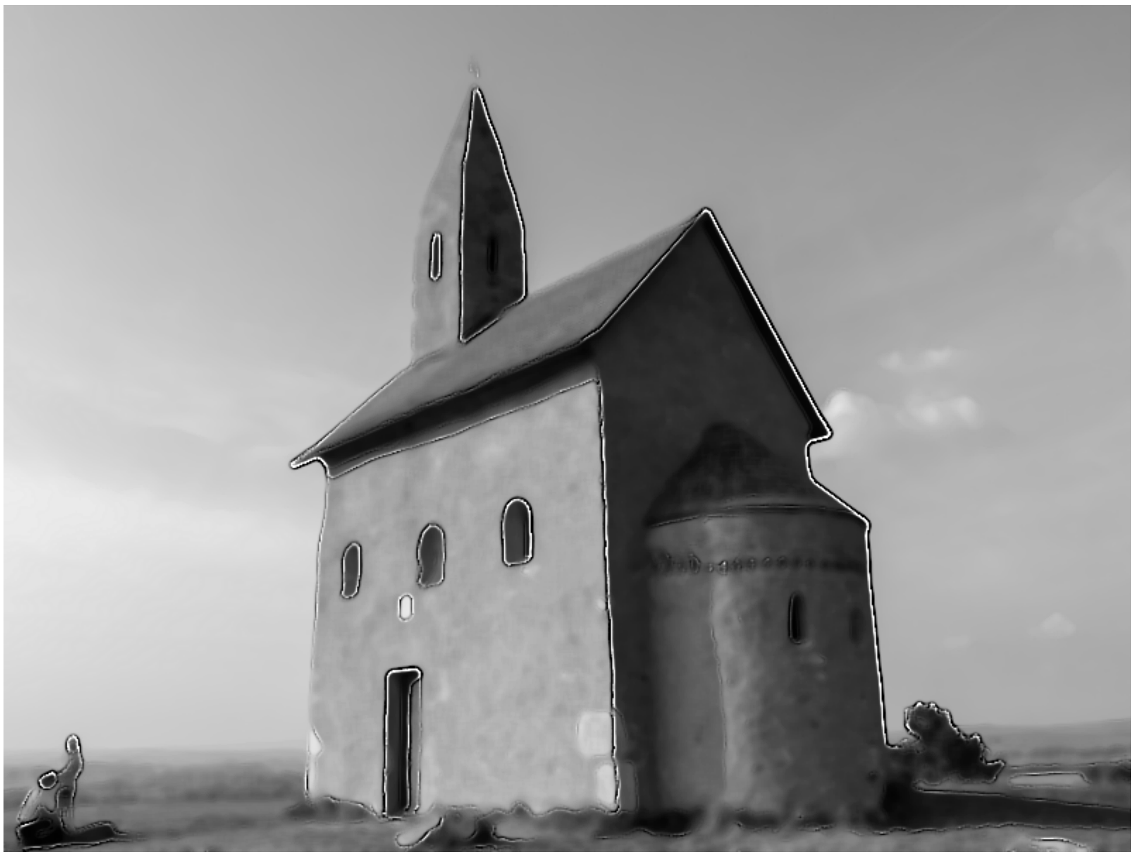
---

d

- *Sample2*



- *Motivation and Approach*
  - *I use $unsharp - masking$ obtained online to implement edge crisping*
    - *source -> Sobel + low-pass + normalization -> res1*
    - *source -> Laplaican of Gaussian -> res2*
    - *output += res1$\times$res2*
- *result5*



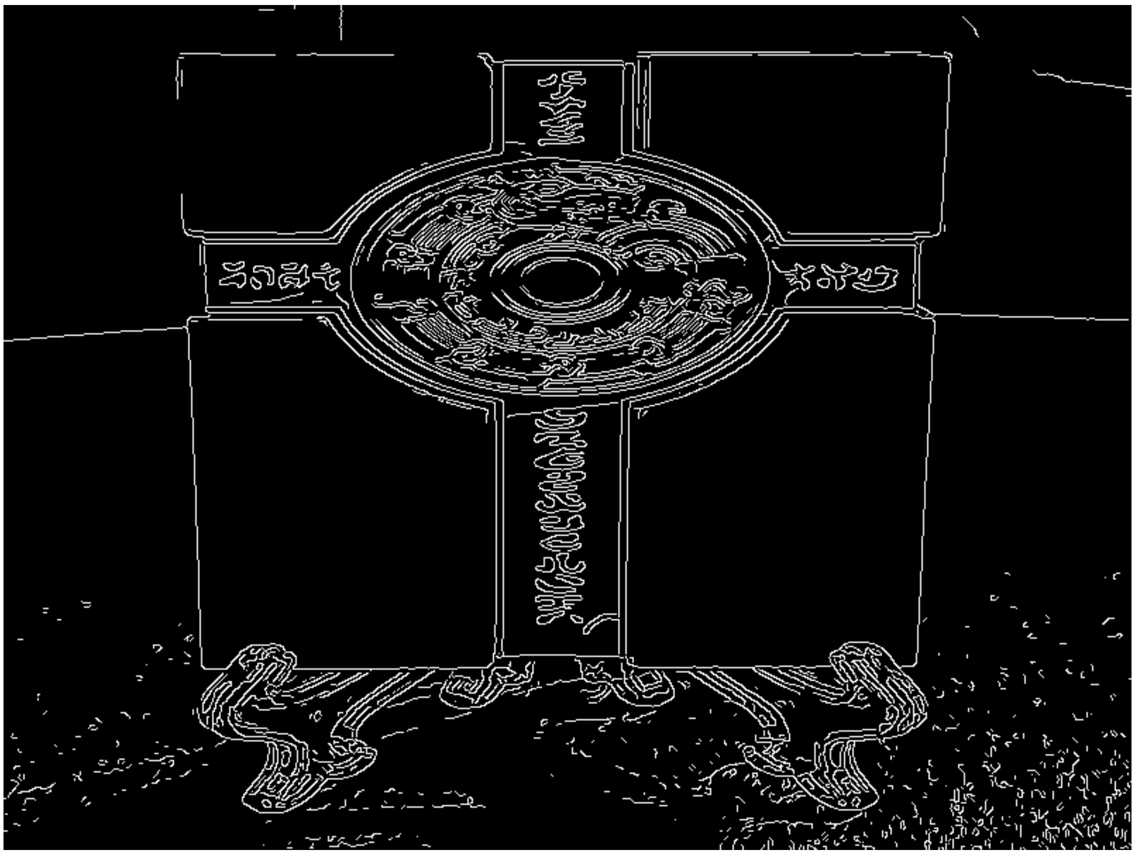
- *Discssion*
  - *I have tried fomula in the lecture slides but the edges in result5 are not obvious.*

- *After using unsharp masking obtained online, the edge become clear. However, I think there are still some edges not been detected yet.*

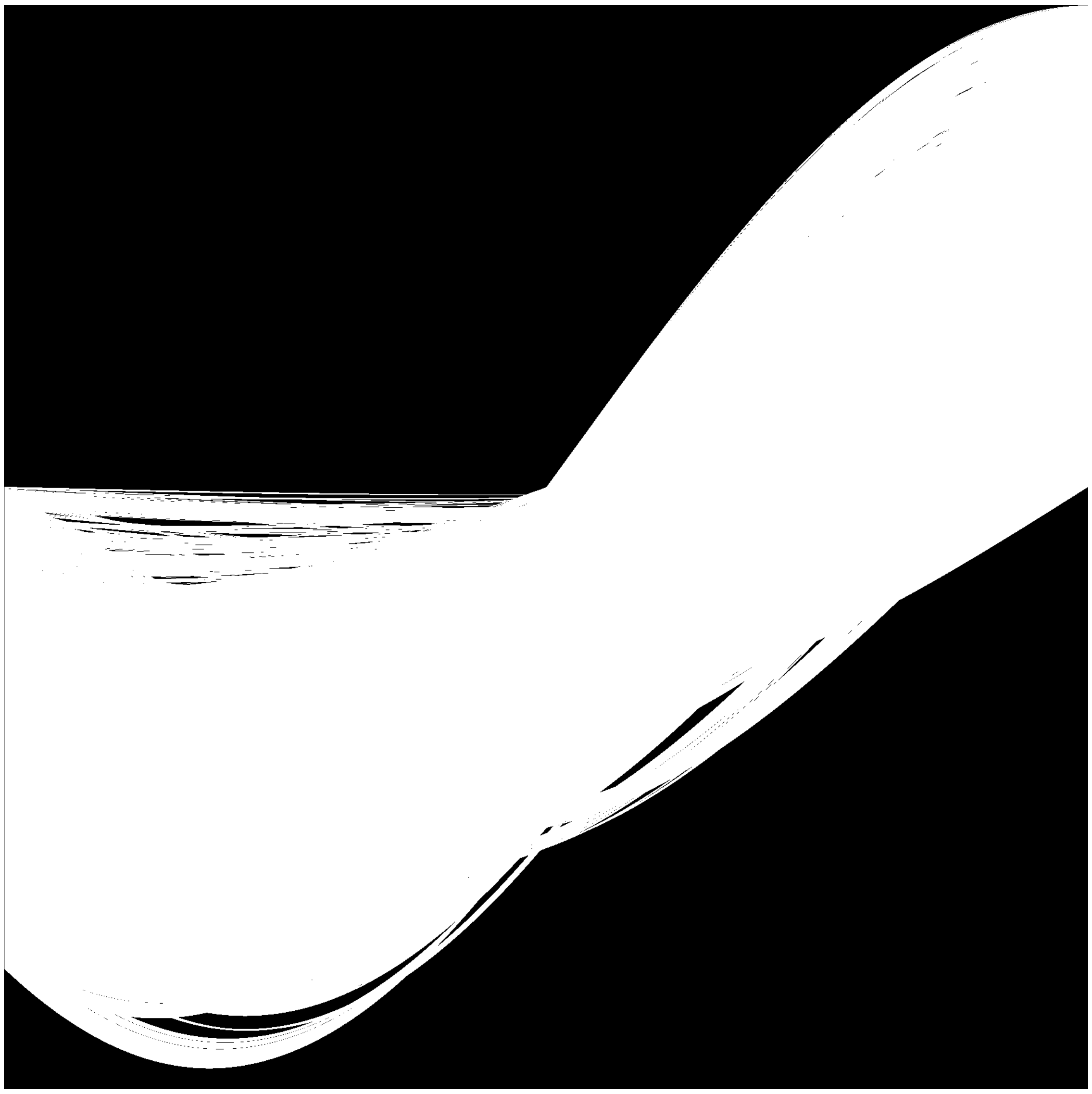
---

e

- *Result3*



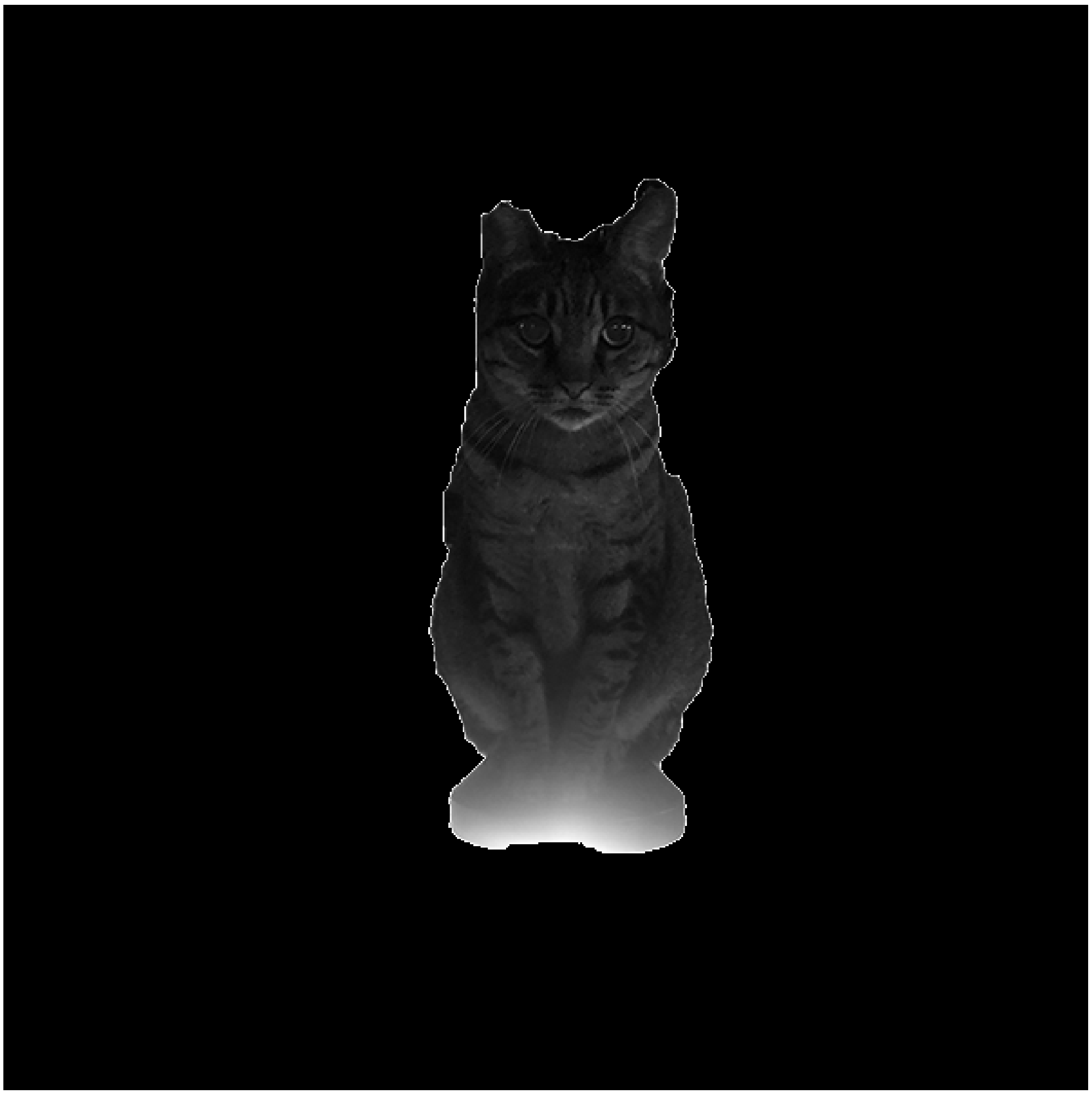
- *Motivation and Approach*
  - *For every edge pixel in result3, do the Hough transformation to a discrete space. Since* $-800 < \rho < 1000$ *and* $0^\circ < \theta < 180^\circ$, *I set Hough space with 1800 rows and 1800 cols.*
  - *For every edge pixel* $(x_0,\ y_0)$, *I try 1800 different* θ*s form* 0° *to* 180°
    - $m = -\frac{1}{tan(\theta)}$
    - $\rho = \frac{(y_0 - m \times x_0)}{(m^2+1^2)^{\frac{1}{2}}}$
    - *map all the* θ*s and* ρ*s to Hough space*
- *Result6*



- *Discssion*
    - *Actually, I not sure whether I do the transformation correctly ot not.*
    - *Assum that I'm right. I think that the Hough space looks wierd because there are too many edge pixels in my result3.*

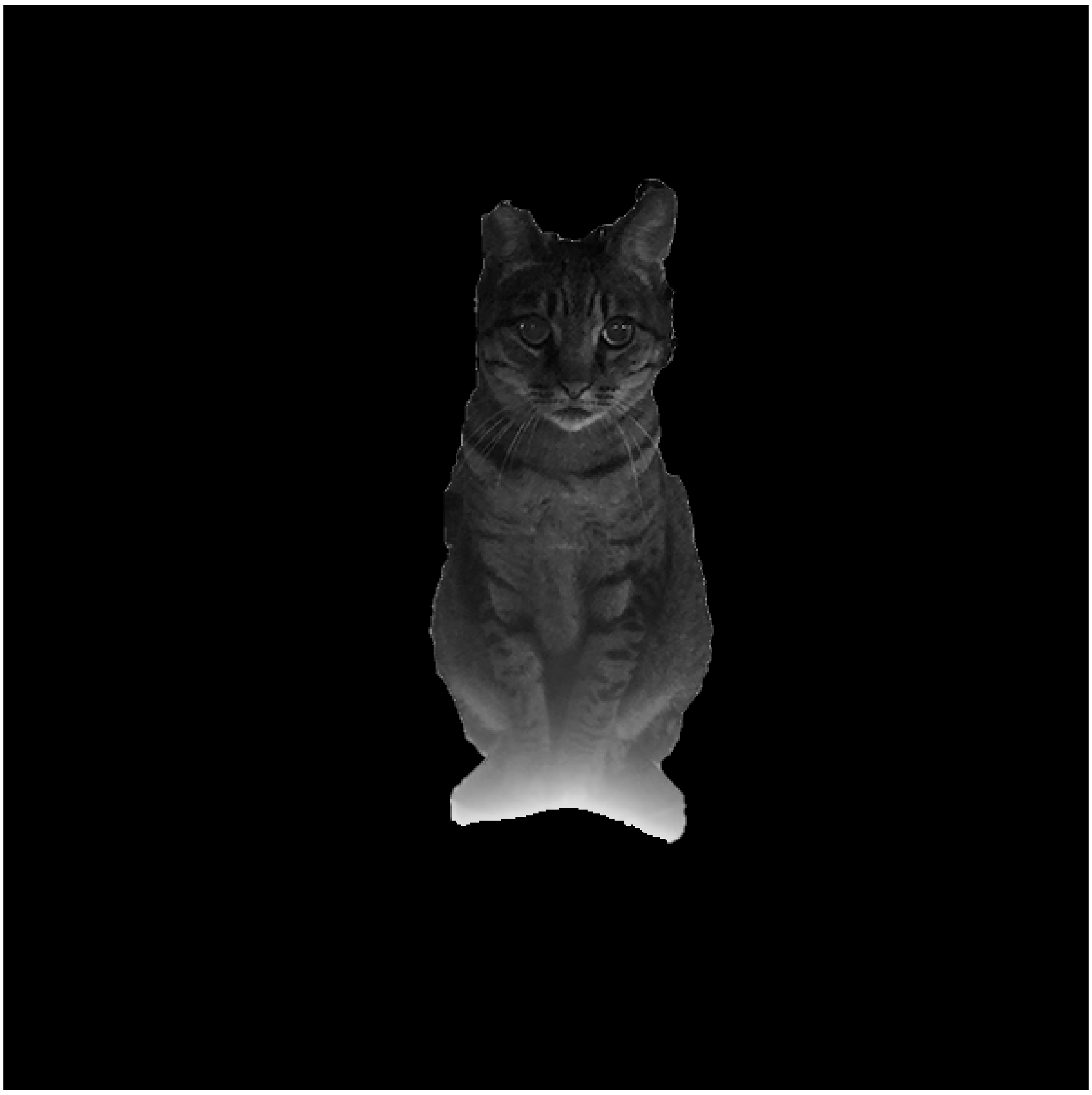
(p) Problem 2

- *Sample3*



a

- *Motivation and Approach*
  - *Since the edage pixel value are relatively high, I pick all pixel* $> T_L$ *and do the mean filtering to eliminate the white edge*
    - *Note that I only consider the neighbor pixels value which* $< T_H$
    - $T_L = 100,\ T_H = 160$
  - *In the end, I also enhance the light a little to make the result more clear.*
- *Improvement*

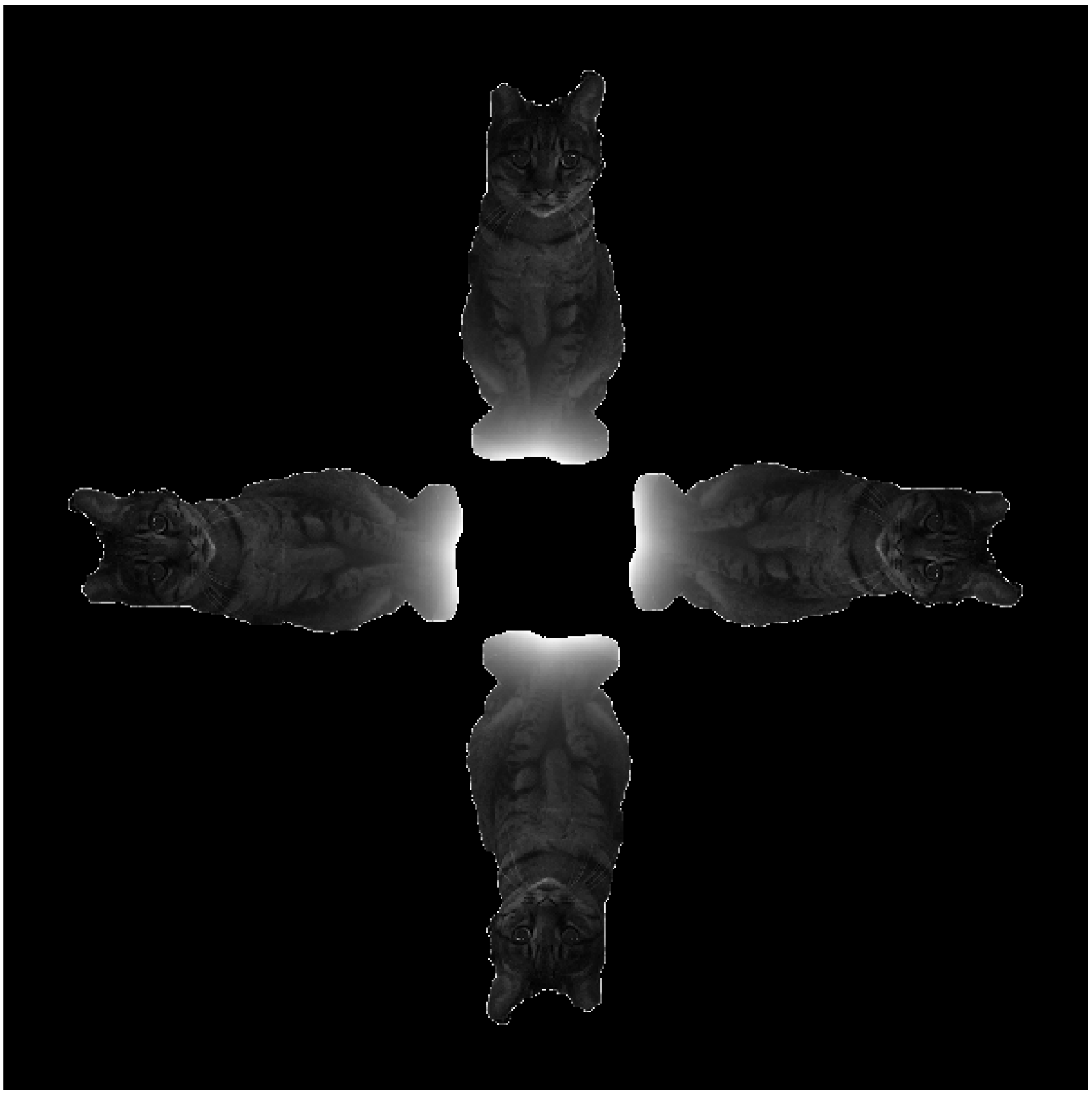


- *Discussion*
  - *In the beginning, I have tried using Canny to form the edge map. However, since the edge is quite simple to detect, I change to the method in motivation part.*
  - *In order to eliminate the effect of high pixel value, I only consider the low pixel values ($< T_H$) when implementing mean filtering.*

---

b

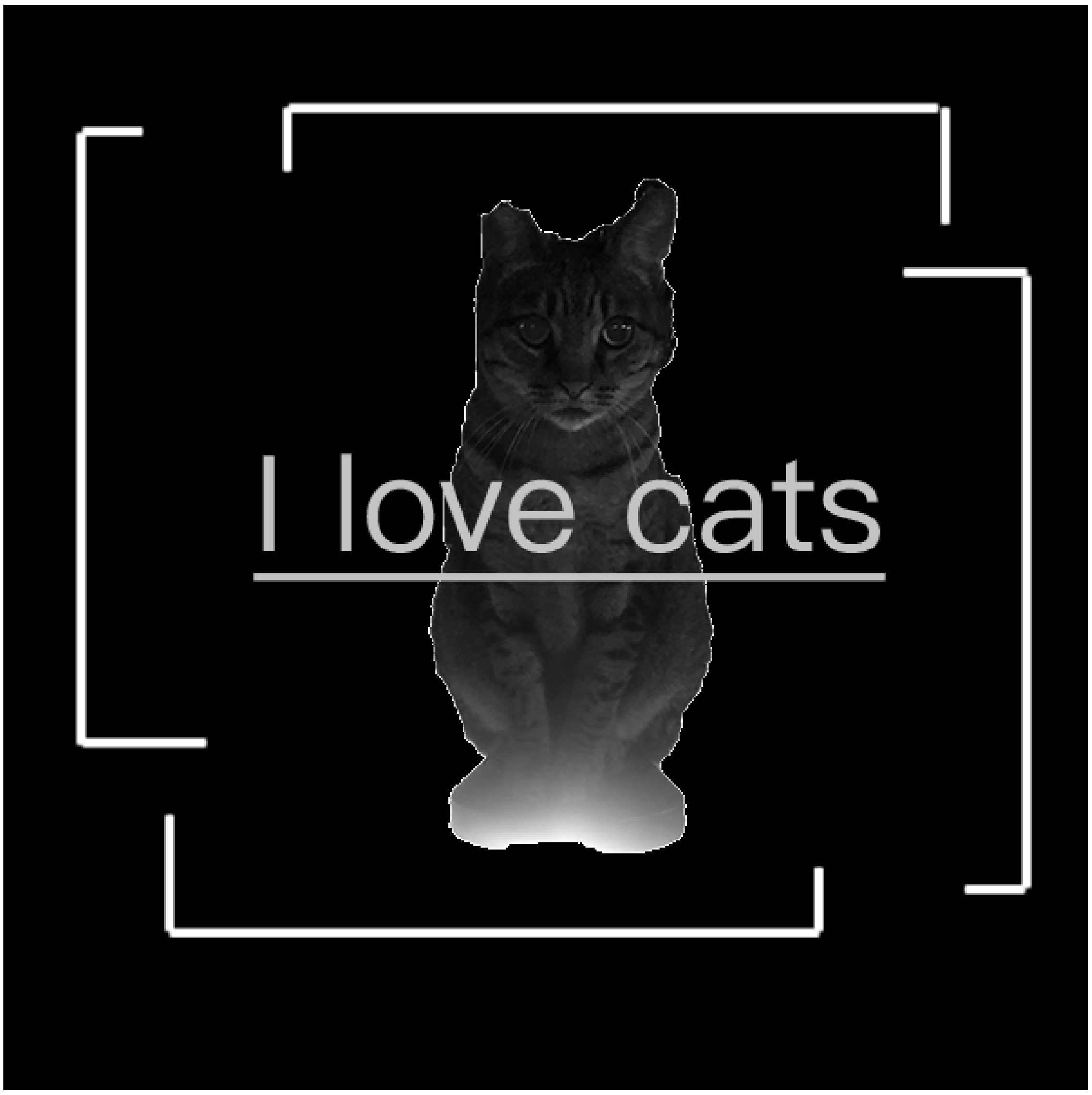
- *Motivation and Approach*
  - *Scaling, shifting, and rotation (under Cartesian coordinate)*
  - *Scaling sample3 to 350 x 350 and removing redundant background to 300 x 160. Then, shifting to the top-middle of result7 (all pixel values are 0 in the beginning)*
  - *result7 += rotation of result7 ($\pi/2$, clockwisely)*
  - *result7 += rotation of result7 (π, clockwisely)*
- *result7*



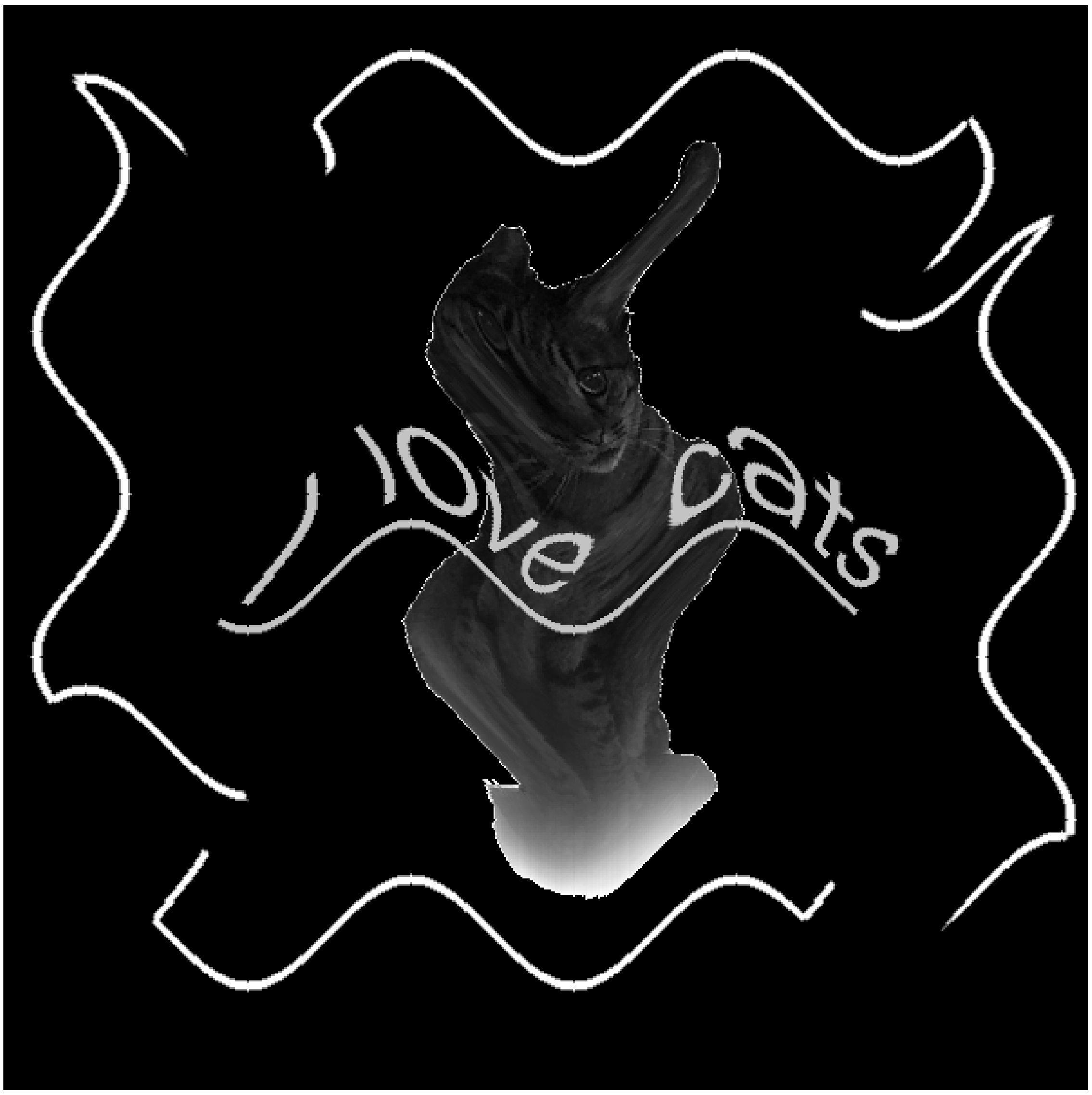
- *Discussion*
  - *In order to make result7 looks like sample4, I have tried different scaling ratio.*
  - *Note that the rotation center is (299.5, 299.5) in Cartesian coordinate*

C

- *Original Image*



- *Motivation and Approach*
    - *Using sin and cos function to warping sample6*
    - *Let $\theta = \pi/90$ and $K = 25$*
    - $result8[i][j] = sample6[(i + I)\%R][(j + J)\%C]$
        - $I = 1.2K(sin(j\theta))$
        - $J = K(cos(i\theta))$
        - $R = C = 600$
- *result8*



- *Discussion*
    - *I have tried different combinations of sin, cos, θ, and K to indirectly adjust the wave strength, direction and frequency.*
    - *However, there is still a little difference between sample6 an result8. For example, wave frequency.*

---

tags: `DIP`