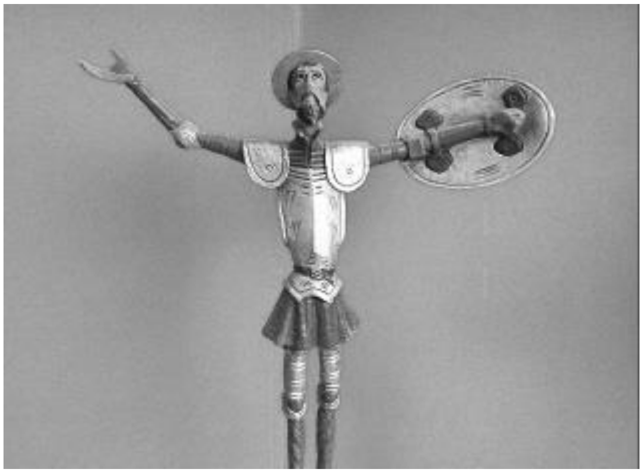
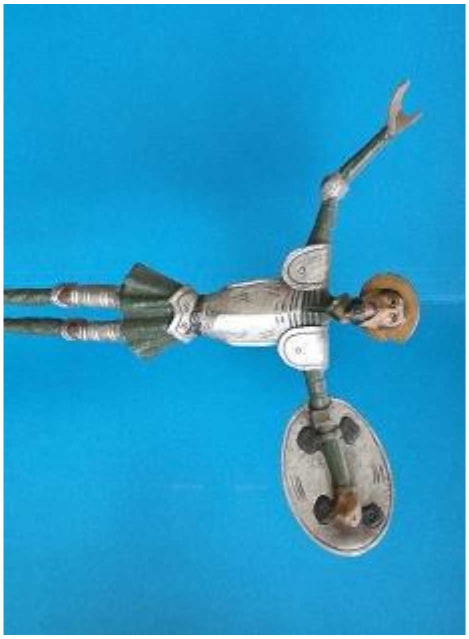
Original

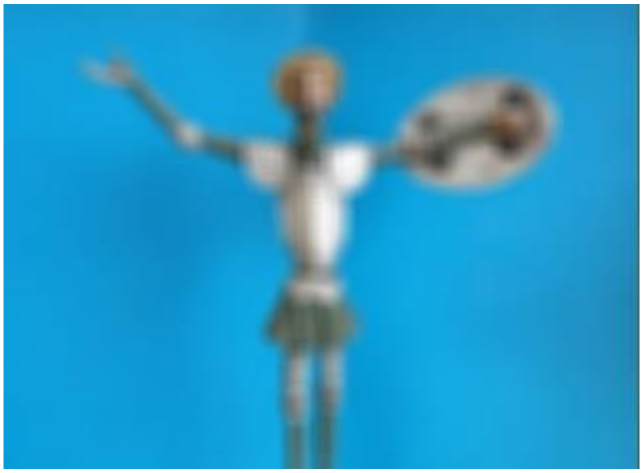
Grayscale



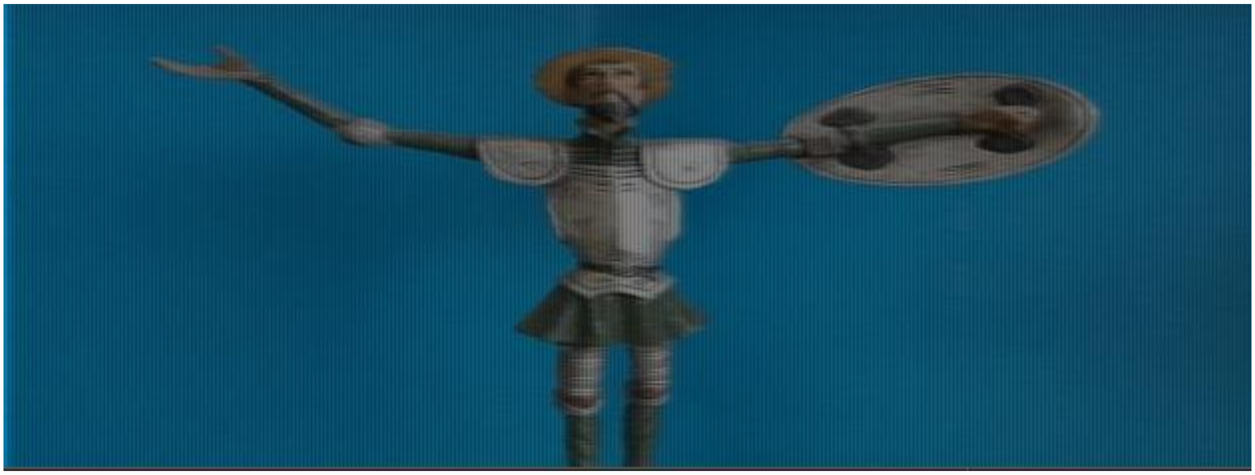
Rotated



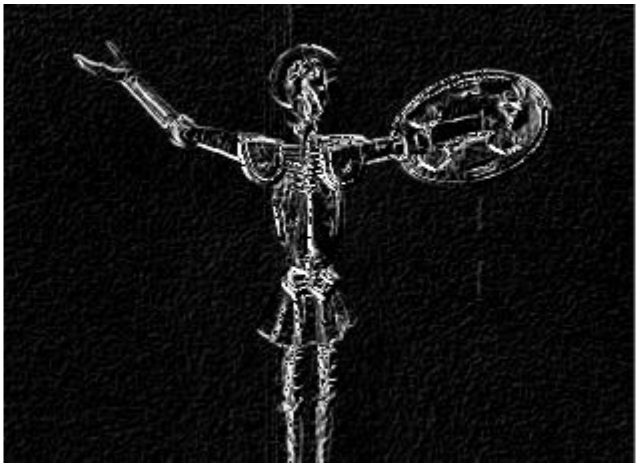
Blurred



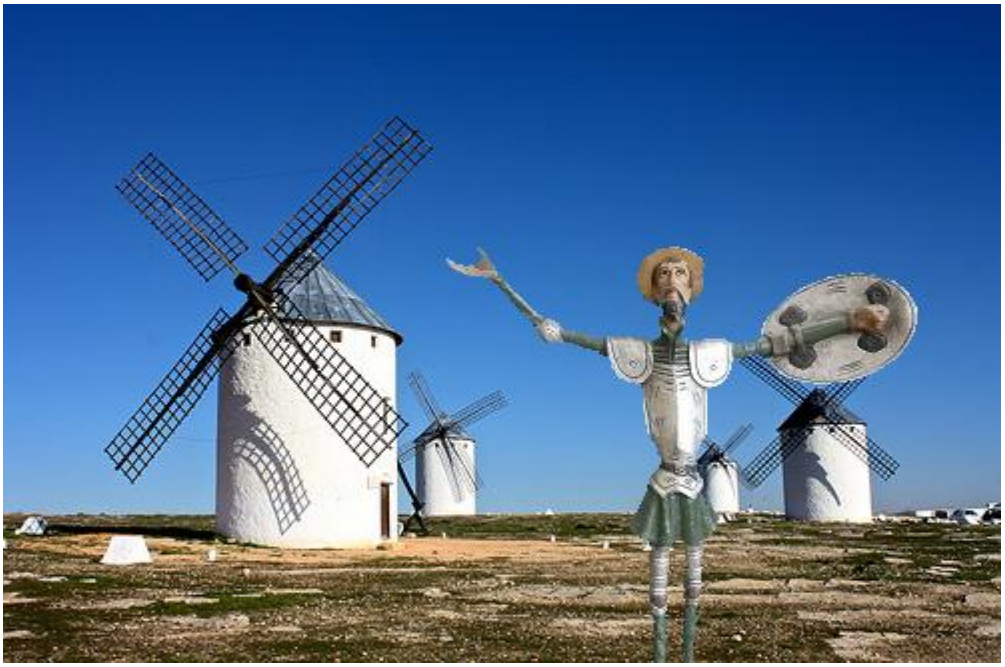
Enlarged



Edges



Blue screening



Code available on github as well!

https://github.com/jgcaballero/ComputerVision/tree/master/Extra%20Credit

```
import numpy as np

import cv2

from scipy.interpolate import interp1d


#a)

def grayscale():

    image = cv2.imread('images/quijote_lr.jpg')

    grayValue = 0.07 * image[:,:,2] + 0.72 * image[:,:,1] + 0.21 * image[:,:,0]

    gray_img = grayValue.astype(np.uint8)

    cv2.imshow('grayscale',gray_img)


#b)

def rotate():

    image2 = cv2.imread('images/quijote_lr.jpg')
    upside = image2[::-1,::1]/255

    row = upside.shape[0]
    col = upside.shape[1]

    rotated = np.zeros((col,row,3))

    print('row',upside.shape[0])
    print('col',upside.shape[1])

    for y in range(upside.shape[0]):
        for x in range(upside.shape[1]):
            rotated[x,y] = upside[y,x]

    print(rotated)
    cv2.imshow('rotated',rotated)

#c)
def box_filter():
    image = cv2.imread('images/quijote_lr.jpg')
    box_size = 10
    kernel = np.ones((box_size,box_size))/(box_size*box_size)
    blur = np.abs(cv2.filter2D(image,-1,kernel))
    cv2.imshow('box',blur)

#d)
def enlarge():
    image = cv2.imread('images/quijote_lr.jpg')
    new_row = image.shape[0]
    new_col = image.shape[1]*2
    enlarge = np.zeros((new_row, new_col,3))
    for y in range(image.shape[0]):
        for x in range(image.shape[1]):
            if(x != image.shape[1] - 2):
                enlarge[y,x*2] = image[y,x]

    for y in range(enlarge.shape[0]):
        for x in range(enlarge.shape[1]-2):
            px0 = enlarge[y,x]
            px1 = enlarge[y,x+2]
            new_px = (px0 + px1)//2
            enlarge[y,x+1] = new_px

    for y in range(enlarge.shape[0]):
        for x in range(enlarge.shape[1]-2):
            px0 = enlarge[y,x]
            px1 = enlarge[y,x+2]
            new_px = (px0 + px1)//2
            enlarge[y,x+1] = new_px

    ''' Using interp1d, did not know if this was allowed so I actually implemented my own 1step interpolation, they
        came out looking very similar as well.'''
    x = np.array(range(enlarge.shape[1]))
    xnew = np.linspace(x.min(), x.max(), new_col)
    f = interp1d(x,enlarge, axis=1)
    cv2.imshow('enlarge',f(xnew)/255)

#    cv2.imshow('enlarge',enlarge/255)

#e)
def edges():
    image = cv2.imread('images/quijote_lr.jpg',0)
    kernel_v = np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
    kernel_h = np.array([[-1,-2,-1],[0,0,0],[1,2,1]])
    gray_frame_f = np.abs(cv2.filter2D(image,-1,kernel_v))+np.abs(cv2.filter2D(image,-1,kernel_h))


    cv2.imshow('edges',gray_frame_f)


#2)
def blue_bg():
    #SOURCE: https://docs.opencv.org/master/df/d9d/tutorial_py_colorspaces.html#gsc.tab=0
    #https://stackoverflow.com/questions/38357141/identifying-green-circles-from-this-image/38357999#38357999


    image = cv2.imread('images/quijote_lr.jpg')
    windmill = cv2.imread('images/windmill.jpg')
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    black = np.array([0,0,0])


    lower_blue = np.array([90,50,50])
    upper_blue = np.array([110,255,255])


    mask = cv2.inRange(hsv, lower_blue, upper_blue)
    res = cv2.bitwise_and(image,image, mask= mask)


    print(hsv)


    for y in range(windmill.shape[0],0,-1):
        for x in range(windmill.shape[1],0,-1):
            if(x != 183 and y != 100):
                if(np.all(res[y-101,x-184] == black)):
                    windmill[y-1,x-1] = image[y-101,x-184]


            else:
                break
        if(y == 100):
            break
```

```
#    cv2.imshow('frame',image)
#    cv2.imshow('mask',mask)
#    cv2.imshow('res',res)
    cv2.imshow('res',windmill)


image = cv2.imread('images/quijote_lr.jpg')
cv2.imshow('og',image)
grayscale()
rotate()
box_filter()
enlarge()
edges()
blue_bg()

cv2.waitKey(0)
cv2.destroyAllWindows()
```