COL783 Assignment 0 Report
Simultaneous Contrast Experiment

Sanket Sanjaypant Hire `2016CS50402`
Yash Malviya `2016CS50403`

1 Introduction

Simultaneous Contrast Effect is an experiment in which a person perceives the difference in the intensity(contrast) of foreground object on changing the intensity of the background object.

In the example shown below, all the inner squares have same intensity, but upper inner square looks lighter in color and the lower inner square look brighter in color. (Optical Illusion)

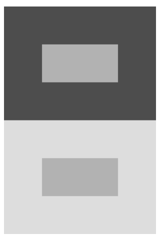


Figure 1: Simultaneous Contrast Effect [2]

Brightness discrimination is the ability of the eye to discriminate between changes in light intensity at any specific adaptation level.

Weber's ratio ($\Delta I/I$) is the ratio of change in the intensity of the foreground ΔI with the respect to the background intensity I that makes it distinguishable. When we measure the increment intensity on various intensity backgrounds, the intensity of foreground increase in proportion to the background (Weber's ratio is constant).

2 Implementation

We were given with the "example.png" which is the concentric square image provided with Assignment 0. For performing experiment, we found the dimensions (side length) and intensity of the squares. They came out to be :-

- Big square
  - Side length: 106
  - Intensity: 77
- Small square
  - Side length: 36
  - Intensity: 180

Then we formed a new image containing concentric squares with same intensities and sizes mentioned above. In this image foreground or background intensity was changed.

We performed two experiments as follows

2.1 Experiment 1

Changing the intensity of the background and observing the change in the contrast. Results are presented in Table 1. Implementation is explained in the code comments under A.1

2.2 Experiment 2

Setting both the background and foreground of same intensity and changing foreground intensity in order to distinguish the foreground from background (Weber's Ratio). Results are presented in Table 2, Table 3 and Table 4. Implementation is explained in A.2

3 Observations

3.1 Optical Illusion

Foreground color is perceived differently by Human Eye (even if Foreground intensity is same in all three figures) depending on changing the background.

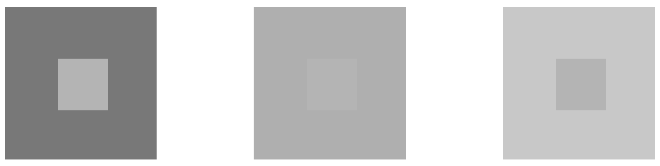


Figure 2: Comparing contrast between Foreground and Background

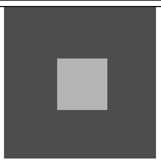
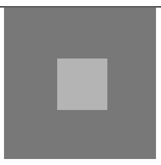
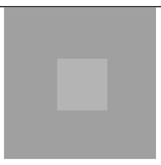
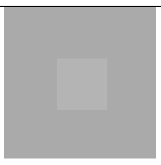
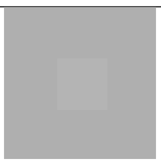
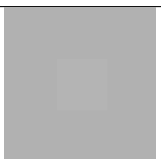
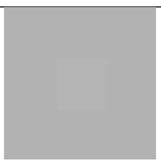
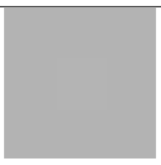
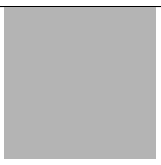
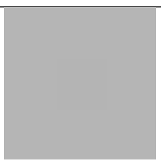
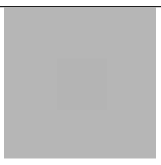
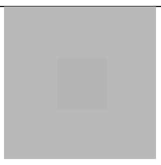
Image	Background Intensity (I)	Foreground Intensity	Change (ΔI)	Weber Ratio ($\Delta I/I$)
	77	180	103	133.7%
	120	180	60	50%
	160	180	20	12.5%
	170	180	10	5.88%
	175	180	5	2.85%
	177	180	3	1.69%
	178	180	2	1.12%
	179	180	1	0.55%
	180	180	0	0%
	181	180	1	0.55%
	182	180	2	1.09%
	185	180	5	2.70%

Table 1: Experiment 1: Fixed foreground intensity of 180

3.2 Weber Ratio is constant

Difference in intensity is observable at 2% Weber ratio. Moving across different values of Weber's ratios for different intensities (Table 2, Table 3, Table 4), the 2% mark is where the foreground square becomes barely visible.

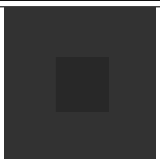
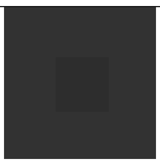
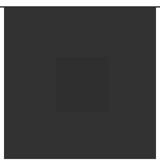
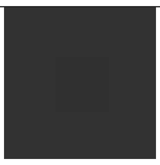
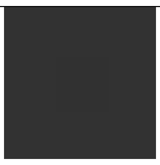
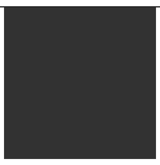
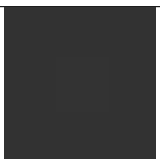
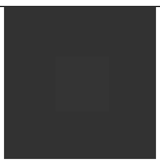
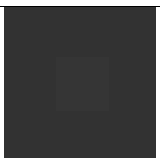
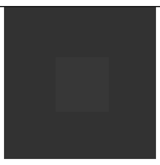
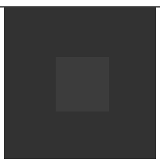
Image	Background Intensity (I)	Foreground Intensity	Change (ΔI)	Weber Ratio ($\Delta I/I$)
	50	40	10	20%
	50	45	5	10%
	50	47	3	6%
	50	48	2	4%
	50	49	1	2%
	50	50	0	0%
	50	51	1	2%
	50	52	2	4%
	50	53	3	6%
	50	55	5	10%
	50	60	10	20%

Table 2: Experiment 2: Fixed background intensity of 50

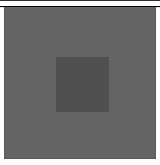
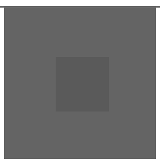
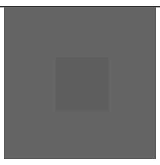
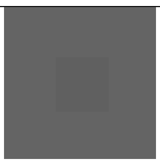
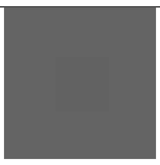
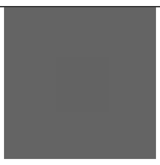
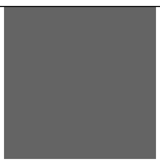
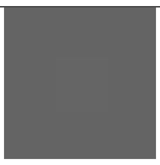
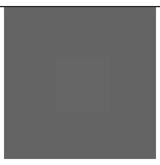
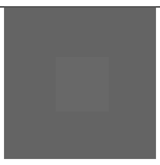
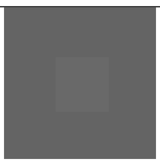
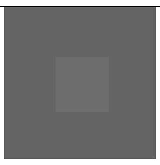
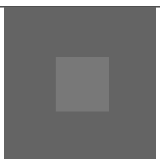
Image	Background Intensity (I)	Foreground Intensity	Change (ΔI)	Weber Ratio ($\Delta I/I$)
	100	80	20	20%
	100	90	10	10%
	100	95	5	5%
	100	96	4	4%
	100	98	2	2%
	100	99	1	1%
	100	100	0	0%
	100	101	1	1%
	100	102	2	2%
	100	104	4	4%
	100	105	5	5%
	100	110	10	10%
	100	120	20	20%

Table 3: Experiment 2: Fixed background intensity of 100

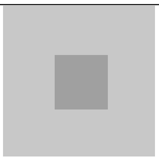
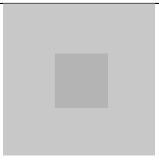
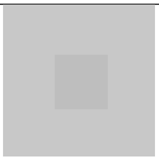
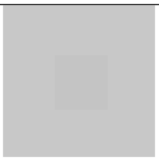
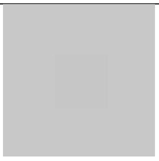
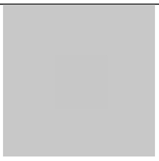
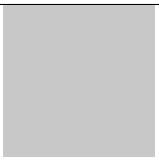
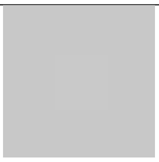
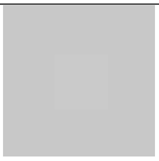
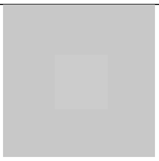
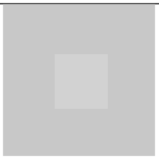
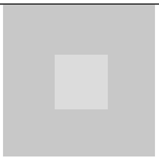
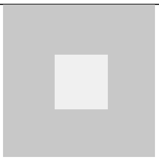
Image	Background Intensity (I)	Foreground Intensity	Change (ΔI)	Weber Ratio ($\Delta I/I$)
	200	160	40	20%
	200	180	20	10%
	200	190	10	5%
	200	196	4	2%
	200	198	2	1%
	200	199	1	0.5%
	200	200	0	0%
	200	201	1	0.5%
	200	202	2	1%
	200	204	4	2%
	200	210	10	5%
	200	220	20	10%
	200	240	40	20%

Table 4: Experiment 2: Fixed background intensity of 200

References

[1] OpenCV: `https://docs.opencv.org/master/`

[2] Simultaneous Contrast Effect Image: `https://en.wikipedia.org/wiki/File:Simultaneous_Contrast.svg`

A Source Code for Implementation of Experiments

A.1 Experiment 1

```
import cv2 as cv
import numpy as np

np.set_printoptions(threshold=np.inf)

# calculates size and intensity of inner outer box
def size_intensity_calculator(img):
        big, _ = img.shape
        big_i = img[0][0]

        thresh = 3
        max_x = 0
        small_i = 0
        top_left = (0, 0)

        h = 0
        # iterate over image pixels from left most pixel, until intensity rise by a value more than
        ↪  thresh
        for row in img:
                w = 0
                for x in row:
                        x_i = int(x)
                        # print(x_i)
                        if x_i - max_x >= thresh:
                                # print("reached")
                                max_x = x_i
                                top_left = (w, h)
                                small_i = x_i
                        w += 1
                h += 1

        small = big - 2*top_left[0]
        return small, big, small_i, big_i

# Create concentric boxes just as the example image
def concentric_box(small, big, small_i, big_i):
        img = np.ones([big, big], dtype=np.uint8) * big_i
        for i in range(big//2-small//2, big//2+small//2+1):
                for j in range(big//2-small//2, big//2+small//2+1):
                        img[i][j] = small_i
        return img

# Overwrite pixels intensity of outer box i.e. background
def bg_change(small, big, big_i, img):
        for i in range(0, big):
                for j in range(0, big):
                        if (big//2-small//2 >= i or i >= big//2+small//2+1) or (big//2-small//2+1
                        ↪  >= j or j >= big//2+small//2+1):
                                img[i][j] = big_i

if __name__ == "__main__":
        img = cv.imread("example.png", cv.IMREAD_GRAYSCALE)
        small, big, small_i, big_i = size_intensity_calculator(img)
        print("small square size: {}, big square size: {}, small intensity: {}, big intensity:
        ↪  {}".format(small, big, small_i, big_i))
        new_img = concentric_box(small, big, small_i, big_i)

        while True:
                bg_intensity = int(input("enter background intensity: "))
                weber = 100*abs(small_i - bg_intensity)/bg_intensity
                print("weber ratio: {}%".format(weber))
                if bg_intensity < 0 or bg_intensity > 255:
                        break
                bg_change(small, big, bg_intensity, new_img)
                # concat_img = np.concatenate((img, new_img))
                # cv.imshow("concatenated images", concat_img)
                cv.imshow("images", new_img)
                cv.imwrite("bg-change-bgi-{}-fgi-{}-wbr-{}.png".format(bg_intensity, small_i,
                ↪  weber), new_img)
                k = cv.waitKey(1)
```

A.2 Experiment 2

```
import cv2 as cv
import numpy as np
```

```python
np.set_printoptions(threshold=np.inf)

def size_intensity_calculator(img):
        big, _ = img.shape
        big_i = img[0][0]

        thresh = 3
        max_x = 0
        small_i = 0
        top_left = (0, 0)

        h = 0
        for row in img:
                w = 0
                for x in row:
                        x_i = int(x)
                        # print(x_i)
                        if x_i - max_x >= thresh:
                                # print("reached")
                                max_x = x_i
                                top_left = (w, h)
                                small_i = x_i
                        w += 1
                h += 1

        small = big - 2*top_left[0]
        return small, big, small_i, big_i

def concentric_box(small, big, small_i, big_i):
        img = np.ones([big, big], dtype=np.uint8) * big_i
        for i in range(big//2-small//2, big//2+small//2+1):
                for j in range(big//2-small//2, big//2+small//2+1):
                        img[i][j] = small_i
        return img

# Overwrite pixels intensity of inner box i.e. foreground
def fg_change(small, big, big_i, img):
        for i in range(0, big):
                for j in range(0, big):
                        if (big//2-small//2 <= i <= big//2+small//2+1) and (big//2-small//2+1 <= j
                        ↪  <= big//2+small//2+1):
                                img[i][j] = big_i

if __name__ == "__main__":
        img = cv.imread("example.png", cv.IMREAD_GRAYSCALE)
        small, big, _, _ = size_intensity_calculator(img)
        print("small square size: {}, big square size: {}".format(small, big))
        # Background intensity taken from user
        big_i = int(input("Enter background intensity: "))
        # Same background foreground intensity
        new_img = concentric_box(small, big, big_i, big_i)

        while True:
                # Foreground intensity modified gradually
                bg_intensity = int(input("enter foreground intensity: "))
                weber = 100*abs(bg_intensity - big_i)/big_i
                print("weber ratio: {}%".format(weber))
                if bg_intensity < 0 or bg_intensity > 255:
                        break
                fg_change(small, big, bg_intensity, new_img)
                cv.imshow("images", new_img)
                cv.imwrite("fg-change-bgi-{}-fgi-{}-wbr-{}.png".format(big_i, bg_intensity, weber),
                ↪  new_img)
                k = cv.waitKey(1)
```