תרגיל בית 4 עיבוד תמונה

מגישים:

אור דינר - 207035809

איתמר- 207931296

שאלה 1

סעיף א': **שיטת סקיילינג יעילה.**

ברוב המקרים שימוש בסקיילינג גאומטרי עדיף על פני התמרת פורייה לצורך שינוי גודל תמונה. השיטה הגאומטרית מהירה יותר ופשוטה ליישום לעומת עבודה במרחב התדר, שדורשת חישובים מורכבים יותר.

סעיף ב': **הוכחת הזהות.**

לפי ההגדרה של טרנספורם פורייה הופכי:

$$f(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{-j2\pi(ux+vy)}du\,dv$$

$$g(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(u,v)e^{-j2\pi(ux+v\ \ )}du\,dv$$

מכיוון שנתון ש-$F(u,v) = G(u,v)$ נקבל:

$$f(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{-j2\pi(ux+vy)}du\,dv =$$

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(u,v)e^{-j2\pi(ux+v\ \ )}du\,dv = g(u,v)$$

$$\rightarrow\ f(u,v) = g(u,v)$$

כנדרש.

שאלה 2

סעיף א': **הצגת תמונה וטרנספורם הפורייה שלה.**

בסעיף הזה טענו את התמונה של הזברה ביצענו מעבר שלה לתמונת פורייה.

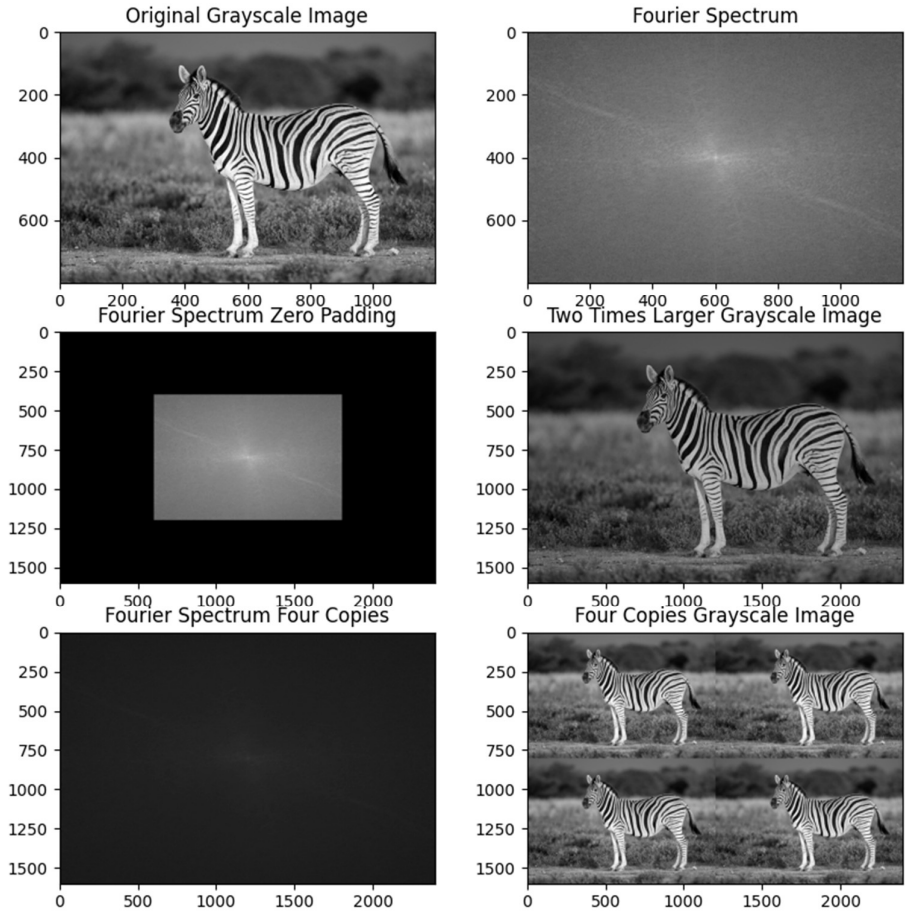
סעיף ב': **הגדלת התמונה על ידי ריפוד אפסים**

בסעיף הזה ריפדנו מסביב לתמונת הפוריה אפסים כך שכשהמרנו אותה חזרה למרחב התמונה קיבלנו תמונה גדולה פי שניים בכל ציר.

```
# Part B: Scaling with Zero Padding (2H x 2W)
padded_F = np.zeros( shape: (2 * H, 2 * W), dtype=np.complex128)
padded_F[H//2:H + H//2, W//2:W + W//2] = F_shifted
padded_F_shifted = ifftshift(padded_F)
padded_image = np.abs(ifft2(padded_F_shifted))
padded_F_magnitude = np.log(1 + np.abs(padded_F))
```

סעיף ג': **שכפול תמונה 4 פעמים**

מימשנו בדומה לשאלה 1 בתרגול 6 את הופעת התמונה ארבע פעמים על ידי ריפוד כל פיקסל במרחב הפוריה באפסים.



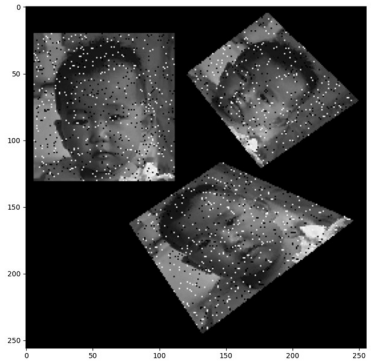
שאלה 3

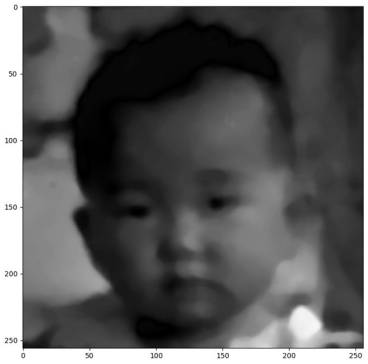
סעיף א': **התינוק**

בדומה לתרגיל 2 בחרנו נקודות על מנת "לחתוך" וליישר את תמונות התינוק, הפעלנו טשטוש חציוני על מנת להפטר כמעט באופן מוחלט מרעשי המלח פלפל, עבור התמונה המיושרת בחרנו טשטוש בעל גרעין בגודל 9 ועבור התמונות שהיה צריך ליישר בחרנו גרעין בגודל 11. לאחר מכן יצרנו תמונה שכל פיקסל בה הוא הממוצע של פיקסלים משלושת התמונות. לבסוף התאמנו את הניגודיות והבהירות של התמונה על מנת לקבל תמונה משופרת.

```python
def clean_baby(im):
    # Define the source points (corners of the input images)
    source_points = np.array([[[6, 20], [111, 20], [111, 130], [6, 130]],
                              [[78, 162], [145, 117], [245, 160], [132, 244]],
                              [[182, 5], [249, 70], [176, 120], [121, 51]]]
                             , dtype=np.float32)

    # Define the destination points (corners of the output image)
    image_height, image_width = im.shape[0:2]
    # image_height,image_width = im.shape

    destination_points = np.array([[0, 0], [image_width-1, 0], [image_width-1, image_height-1], [0, image_height-1]], dtype=np.float32)

    # Extracted images array
    images_arr = []

    # Extract all the images
    for points in source_points:

        # Compute the perspective transformation matrix
        matrix = cv2.getPerspectiveTransform(points, destination_points)

        # Perform the perspective warp
        transformed_image = cv2.warpPerspective(im, matrix, (image_width, image_height))

        if points[0][0] == 6:
            kernel_size = 9
        else:
            kernel_size = 11
        filtered_image = cv2.medianBlur(transformed_image, ksize=kernel_size)

        # Add the new image to the array
        images_arr.append(filtered_image)

    # Stack the images into a 3D array
    stacked_images = np.stack(images_arr, axis=-1)

    # Compute the median along the last axis (across the three images)
    median_image = np.mean(stacked_images, axis=-1).astype(np.uint8)

    adjusted = cv2.convertScaleAbs(median_image, alpha=1.2, beta=-35)

    return adjusted
```

התמונה:

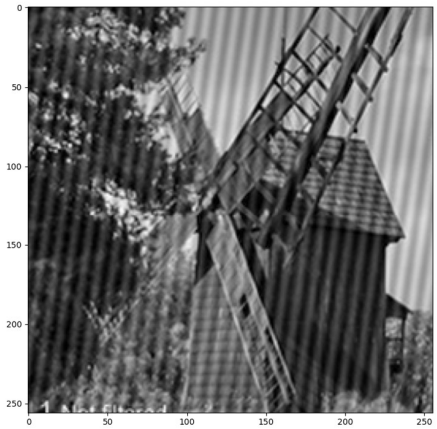


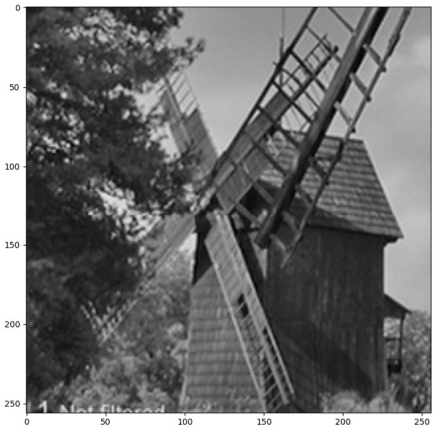


סעיף ב': **תחנת רוח**

זיהינו שלתמונה יש רעש תדרי לכן המרנו אותה למרחב הפורייה ואפסנו את התדרים בעלי ערך גבוהה מ-200000 החזרנו את התמונה ממרחב הפורייה וקיבלנו את התמונה המתוקנת ללא הרעש התדרי.

התמונה:



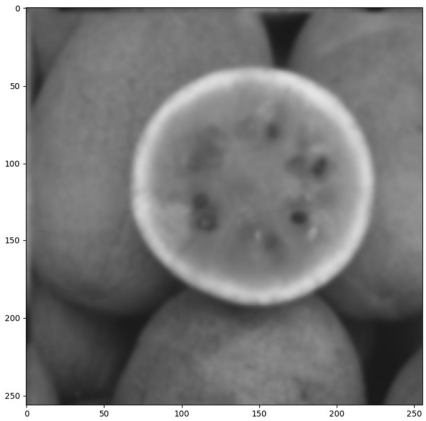


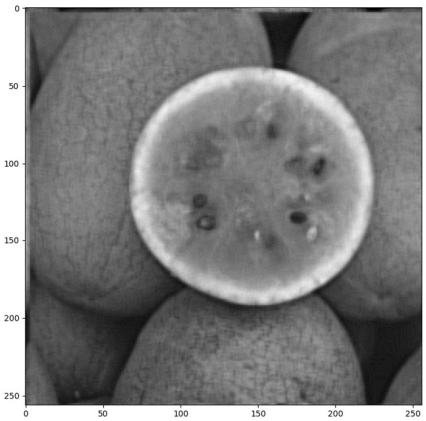
סעיף ג': **אבטיח**

זיהינו שהתמונה מטושטשת לכן הפעלנו גרעין לחידוד התמונה.

```
# CLEAN WATERMELON
def clean_watermelon(im):  1 usage
    # sharpen image
    kernel = np.array([[-1, -1, -1],
                       [-1, 9, -1],
                       [-1, -1, -1],])
    im_sharpened = cv2.filter2D(im, -1, kernel)

    return im_sharpened
```

התמונה:

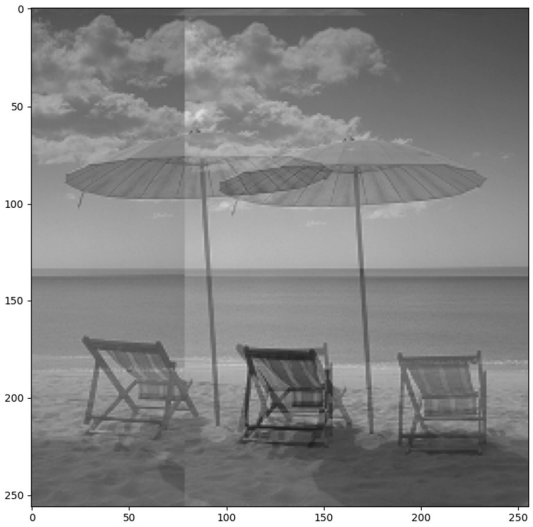


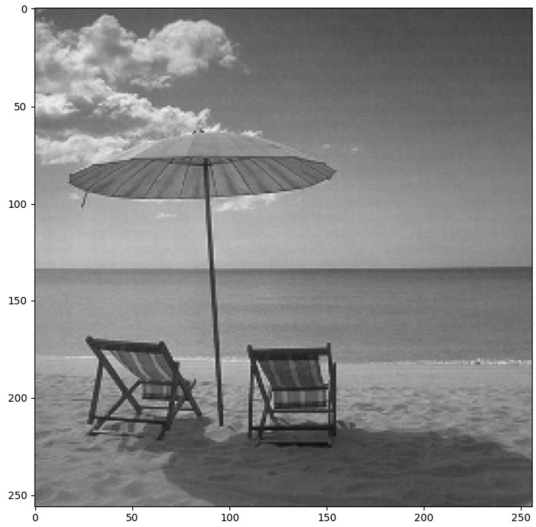


סעיף ד': **מטרייה**

על מנת לתקן את התמונה השתמשנו בפילטר ווינר. בהתחלה בנינו את הגרעין שהוא מטריצה בגודל התמונה כאשר הכול אפסים חוץ מה-[0, 0] שהוא 1 ותיקון התזוזה הרצויה [4, 79] שהוא גם 1. המרנו את התמונה והגרעין למרחב הפורייה וחישבנו את הדה קונבולוציה לפי ווינר. המרנו חזרה ממרחב הפורייה וקיבלנו את התמונה המתוקנת.

התמונה:



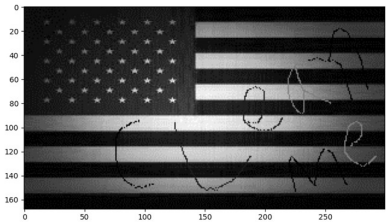


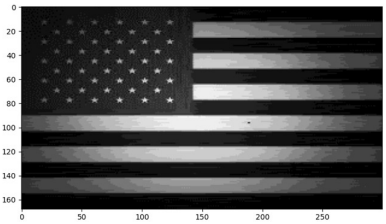
סעיף ה': **דגל**

על מנת להפטר מהכיתוב ביצענו פילטר חציוני אופקי כך שכל פיקסל שהוא לא בתחום הכוכבים של הדגל יקבל את הערך החציוני של הפיקסלים בין 6 ימינה ל-6 שמאלה. כך ייעלם הכיתוב כמעט לגמרי ונקבל תמונה נקייה.

```python
def clean_USAflag(image):  1 usage
    # Your code goes here
    # application of median filter HORIZONTALLY starting at predetermined thresholds
    filtered_image = np.copy(image)

    # Apply horizontal median filter only to rows > 75 and cols > 120
    for x in range(image.shape[0]):  # Iterate over each row
        if x < 85:
            for y in range(140, image.shape[1]):
                filtered_image[x, y] = np.median(get_neighbors(image, x, y))
        else:
            for y in range(image.shape[1]):
                filtered_image[x, y] = np.median(get_neighbors(image, x, y))

    return filtered_image
```

התמונה:

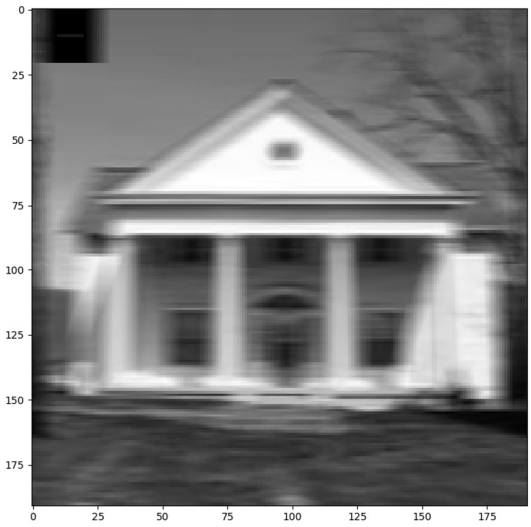


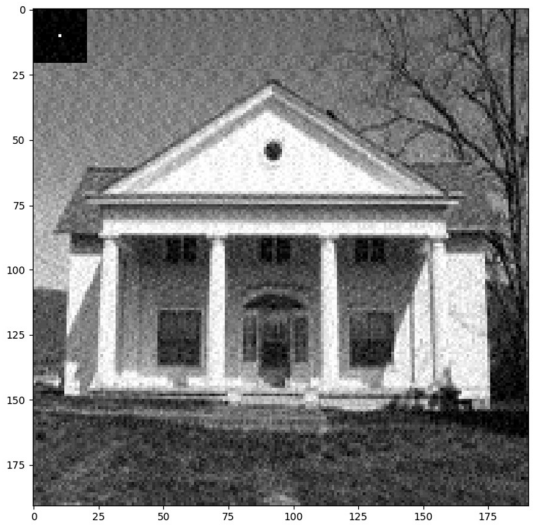


סעיף ו': **בית**

על מנת לתקן את התמונה שנוצרה ממוצע של התמונה מוזזת יצרנו את הגרעין שלדעתנו הוביל ליצירת התמונה הממוצעת. המרנו את התמונה ואת הגרעין למרחב פורייה. ביצענו חלוקה של התמונה בגרעין (דה קונבולוציה) על מנת לקבל את התמונה המקורית והחזרנו אותה ממרחב פורייה למרחב התמונה.

התמונה:





סעיף ז: **דובים**

התמונה נראית חשוכה לכן התחלנו לנסות לתקן אותה בעזרת תיקון גמא. לאחר שראינו שהתמונה לא הופכת להיות דונה יותר עברנו לנסות לשנות את הניגודיות והבהירות עד שהגענו לתוצאה קרובה מאוד לתמונה המתוקנת שמסופקת לנו.