עיבוד ספרתי של תמונות: תרגיל בית 1  
מגישים:  
שלמה עזרא 205359938  
יונתן גולן 208980888

חלק א':  
1. אינטרפולציה:  
.a כתבנו פונקציה " interpolation" אשר מבצעת interpolation bilinear (מגודל nxm ל – 2nx2m).

תמונה שמכילה ירק, פלפל מתוק, פלפל, אוכל טבעי

התיאור נוצר באופן אוטומטי  
.b טענו את התמונה peppers.jpg והגדלנו אותה בפקטור 2 בעזרת הפונקציה שקיבלנו.  
  
 התוצאה שקיבלנו:

ניתן לראות כי התוצאה מטושטשת יותר מהתמונה המקורית מכיוון שבאינטרפולציה הגדלנו את התמונה פי 2 ואנו עושים ממוצע של הפיקסלים מהתמונה המקורית כדי להשלים את הפיקסלים שלא היו קיימים לנו, כתוצאה מהממוצע הזה נוצר לנו טשטוש.  
  
  
  
את התוצאה שמרנו כנדרש בתיקייה של חלק 1 תחת השם "Peppers\_interpolated\_by\_2".

c. חזרנו על הסעיף הקודם והגדלנו בפקטור 8 (מתקבל ע"י הגדלת התמונה המקורית 3 פעמים בפקטור 2 או לחלופין הגדלת התמונה שקיבלנו בסעיף הקודם פעמיים נוספות בפקטור 2 – כך הקוד עצמו יעיל יותר..).

התוצאה שקיבלנו:

בדומה לסעיף הקודם קיבלנו תמונה מטושטשת וגדולה יותר.

את התוצאה שמרנו כנדרש בתיקייה של חלק 1 תחת השם "Peppers\_interpolated\_by\_8".  
  
השוואה בין התוצאות של סעיפים c-b:

ניתן לראות כי בשני הסעיפים קיבלנו תמונה גדולות יותר ומטושטשות יותר מהתמונה המקורית, כאשר בסעיף c קיבלנו תמונה גדולה יותר ומטושטשת יותר מהתמונה מסעיף b.   
באופן כללי ככל שאנו מבצעים אינטרפולציה גדולה יותר אנו נדרשים להשלים יותר פיקסלים ולכן אנו מקבלים תמונה גדולה יותר אך מטושטשת יותר.

תמונה שמכילה טקסט, תרשים, קו, עלילה

התיאור נוצר באופן אוטומטי2. שיווי היסטוגרמה:  
.a חישבנו את ההיסטוגרמה של התמונה – leafs.jpg באמצעות הפונקציה " " plot\_hist – אשר מחשבת את ההיסטוגרמה של התמונה, מציגה ושומרת אותה.

ההיסטוגרמה שקיבלנו לתמונה:   
  
  
ניתן לראות את מספר הפיקסלים שיש לנו בכל רמת בהירות.  
  
בנוסף הערכים של הפיקסלים מרוכזים בערך בטווח 90-140 , כך שיש לנו ניגודיות יחסית נמוכה.  
  
את ההיסטוגרמה של התמונה שמרנו בתיקייה של חלק 1 תחת השם " Leafs Histogram".

.bמתיחת קונטרסט – כתבנו את הפונקציה " stretch\_contrast" בה חישבנו לכל פיקסל In את הערך:

תמונה שמכילה שחור ולבן, אוכל, קרקע, מונוכרום

התיאור נוצר באופן אוטומטימכיוון שקיבלנו ערכים לא שלמים עיגלנו את התוצאה.

התמונה אשר קיבלנו:   
  
מתיחת הקונטרסט אפשרה לנו להעלות את הקונטרסט בתמונה כך שהפריטים בתמונה מעט ברורים יותר, כך למעשה הגדלנו את טווח הערכים של הפיקסלים בתמונה.  
  
שמרנו את התמונה בתיקייה של חלק 1 תחת השם  
 " Stretched Leafs".

תמונה שמכילה טקסט, קו, עלילה

התיאור נוצר באופן אוטומטיההיסטוגרמה של התמונה:   
  
ניתן לראות כי לעומת טווח הערכים המקורי של התמונה אשר היה מצומצם (בערך בטווח 90-140)  
קיבלנו כעת טווח הערכים רחב של 0 עד 255.  
  
  
שמרנו את ההיסטוגרמה בתיקייה של חלק 1 תחת השם " Stretched Leafs Histogram".

תמונה שמכילה אוכל מהיר, שחור ולבן, אוכל, מונוכרום

התיאור נוצר באופן אוטומטיc.שיווי היסטוגרמה – ממישנו את הפונקציה " equalize\_hist" אשר מבצעת שיווי היסטוגרמה לתמונה.

התמונה שקיבלנו:

לאחר שביצענו שיווי היסטוגרמה אנו יכולים לראות את הפרטים בתמונה באופן טוב יותר (מכיוון שיש לנו הרבה יותר קונטרסט בתמונה) אולם נוסף לנו הרבה רעש שלא היה קיים בתמונה המקורית.

את התמונה שקיבלנו שמרנו בתיקייה של חלק 1 תחת השם " Equalized Leafs".

תמונה שמכילה טקסט, קו, עלילה, תרשים

התיאור נוצר באופן אוטומטיההיסטוגרמה של התמונה שקיבלנו:

ההיסטוגרמה כעת לאחר שיווי היסטוגרמה מתפלגת באופן יוניפורמי יותר ומצביעה לנו על כך שהקונטרסט בתמונה גדול יותר(אפילו מעט גדול יותר מהסעיף הקודם).

שמרנו את ההיסטוגרמה בתיקייה של חלק 1 תחת השם " Equalized Leafs Histogram".

חלק ב':

נספח 1- הקוד המלא של החלק הראשון:

**import** numpy **as** np

**import** cv2

**from** matplotlib **import** pyplot **as** plt

**def** interpolation**(**img**):** # 1.a

""" This function does super resolution by bilinear interpolation

img: 2D numpy array of the original image (n\*m)

return: 2D numpy array of the new image (2n\*2m)"""

h**,** w **=** img**.**shape

new\_h**,** new\_w **=** h **\*** 2**,** w **\*** 2

new\_img **=** np**.**zeros**((**new\_h**,** new\_w**))**

# Iterate over every pixel in the new image

**for** i **in** **range(**new\_h**):**

**for** j **in** **range(**new\_w**):**

# Find the coordinates in the original image

x **=** i **/** 2

y **=** j **/** 2

# Get the coordinates of the surrounding pixels

x0 **=** **int(**np**.**floor**(**x**))**

y0 **=** **int(**np**.**floor**(**y**))**

x1 **=** **min(**x0 **+** 1**,** h **-** 1**)**

y1 **=** **min(**y0 **+** 1**,** w **-** 1**)**

# Calculate the differences

dx0 **=** x **-** x0

dx1 **=** x1 **-** x

dy0 **=** y **-** y0

dy1 **=** y1 **-** y

# Get the pixel values

top\_left **=** img**[**x0**,** y0**]**

top\_right **=** img**[**x0**,** y1**]**

bottom\_left **=** img**[**x1**,** y0**]**

bottom\_right **=** img**[**x1**,** y1**]**

**if** x0 **==** x1 **and** y0 **==** y1**:**

new\_img**[**i**,** j**]** **=** top\_left # no need to interpolate - take original pixel

**continue**

**if** x0 **==** x1**:**

new\_img**[**i**,** j**]** **=** top\_left **\*** dy1 **+** top\_right **\*** dy0 # interpolate only in the y direction

**continue**

**if** y0 **==** y1**:**

new\_img**[**i**,** j**]** **=** top\_left **\*** dx1 **+** bottom\_left **\*** dx0 # interpolate only in the x direction

**continue**

top **=** top\_left **\*** dy1 **+** top\_right **\*** dy0 # interpolate in the y direction

bottom **=** bottom\_left **\*** dy1 **+** bottom\_right **\*** dy0

new\_img**[**i**,** j**]** **=** top **\*** dx1 **+** bottom **\*** dx0 # interpolate in the x direction

**return** new\_img

**def** plot\_hist**(**img**,** title**=**'Image Histogram'**):**

hist**,** bins **=** np**.**histogram**(**img**.**flatten**(),** 256**,** **[**0**,** 256**])**

plt**.**plot**(**hist**,** color**=**'gray'**)**

plt**.**xlim**([**0**,** 256**])**

plt**.**xlabel**(**'Intensity'**)**

plt**.**ylabel**(**'Number of pixels'**)**

plt**.**title**(**title**)**

plt**.**savefig**(**'./' **+** title **+** '.jpg'**)** # Save the histogram - optional

plt**.**show**()**

**def** stretch\_contrast**(**img**,** title**=**'Stretched Image'**):**

"""

This function stretches the contrast of the image - for each pixel value, the new value is calculated by:

new\_value = (value - min\_value) \* 255 / (max\_value - min\_value)

:param img: 2D numpy array of the original image

:param title: title of the plot

:return: stretched image

"""

min\_value **=** np**.min(**img**)**

max\_value **=** np**.max(**img**)**

new\_img **=** np**.round(((**img **-** min\_value**)** **/** **(**max\_value **-** min\_value**))** **\*** 255**).**astype**(**np**.**uint8**)**

# Plot the new image and its histogram

plt**.**imshow**(**new\_img**,** cmap**=**'gray'**)**

plt**.**title**(**title**)**

cv2**.**imwrite**(**'./' **+** title **+** '.jpg'**,** new\_img**)** # Save the image - optional

plt**.**show**()**

**return** new\_img

**def** equalize\_hist**(**img**,** title**=**'Equalized Image'**):**

"""

This function equalizes the histogram of the image

:param img: 2D numpy array of the original image

:param title: title of the plot

:return: equalized image

"""

hist**,** bins **=** np**.**histogram**(**img**.**flatten**(),** 256**,** **[**0**,** 256**])**

cdf **=** hist**.**cumsum**()**

cdf\_normalized **=** cdf **\*** hist**.max()** **/** cdf**.max()**

cdf\_m **=** np**.**ma**.**masked\_equal**(**cdf**,** 0**)**

cdf\_m **=** **(**cdf\_m **-** cdf\_m**.min())** **\*** 255 **/** **(**cdf\_m**.max()** **-** cdf\_m**.min())**

cdf **=** np**.**ma**.**filled**(**cdf\_m**,** 0**).**astype**(**'uint8'**)**

new\_img **=** cdf**[**img**]**

# Plot the new image and its histogram

plt**.**imshow**(**new\_img**,** cmap**=**'gray'**)**

plt**.**title**(**title**)**

cv2**.**imwrite**(**'./' **+** title **+** '.jpg'**,** new\_img**)** # Save the image - optional

plt**.**show**()**

**return** new\_img

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:** # Part A

# 1.Interpolation:

# 1.b

**print(**'Loading peppers image'**)**

peppers **=** cv2**.**imread**(**'./peppers.jpg'**,** 0**)**

**print(**'Interpolating peppers image by 2'**)**

peppers\_interp\_by\_2 **=** interpolation**(**peppers**)**

**print(**'Saving interpolated peppers image (by 2)'**)**

cv2**.**imwrite**(**'./Peppers\_interpolated\_by\_2.jpg'**,** peppers\_interp\_by\_2**)**

# 1.c

**print(**'Interpolating peppers image by 8 '**)** # 2 times interpolation by 2(to already interpolated by 2 image)

peppers\_interp\_by\_8 **=** interpolation**(**interpolation**(**peppers\_interp\_by\_2**))**

**print(**'Saving interpolated peppers image (by 8)'**)**

cv2**.**imwrite**(**'./Peppers\_interpolated\_by\_8.jpg'**,** peppers\_interp\_by\_8**)**

# 2.Equal Histogram:

**print(**'Loading leaf image'**)**

leaf **=** cv2**.**imread**(**'./leafs.jpg'**,** 0**)**

# 2.a

**print(**'Calculating and plotting the histogram of the leaf image by the pixels values (0-255)'**)**

plot\_hist**(**leaf**,** 'Leafs Histogram'**)**

# 2.b

**print(**'Stretching contrast of the leaf image'**)**

strech\_leaf **=** stretch\_contrast**(**leaf**,** 'Stretched Leafs'**)**

plot\_hist**(**strech\_leaf**,** 'Stretched Leafs Histogram'**)**

# 2.c

**print(**'Equalizing the histogram of the leaf image'**)**

equalize\_leaf **=** equalize\_hist**(**leaf**,** 'Equalized Leafs'**)**

plot\_hist**(**equalize\_leaf**,** 'Equalized Leafs Histogram'**)**

**print(**'Done'**)**