

שיעור מספר 9

עיבוד תמונה

עיבוד אותות ספרתי

- מדידה, ייצוג ומניפולציה של אותות מידע מקורות שונים (קול, תמונה, מכ"ם וסונאר, חיישנים,...)
- אנחנו נתרכז במעט טכניקות פשוטות הקשורות בעיבוד תמונה.
- מה נראה? -
 - ייצוג תמונה (שחור-לבן ובצבע)
 - פילטרים פשוטים
 - היסטוגרמות ופילטרים שימושיים בהן
 - קונוולוציות ופילטרים שימושיים בהן
 - איטרציה Lloyd ושימושים

הספרייה Pillow

- זו ספרייה הכוללת יכולות בסיסיות רבות של עיבוד תמונה. מבוססת על גרסה קודמת שנקראה PIL (ראשי תיבות של Python Imaging Library).
- בתרגול תראו בין היתר שימוש ב-Pillow. בשעור נעשה הכל לבד, מעט קרייה של קבצי תמונות והצגה על המסר של תמונות.
- מידע והדרכה על Pillow ניתן למצוא כאן:
<https://pillow.readthedocs.org/en/latest/>

ייצוג תמונה

- פיסית, מסך של מחשב (או טלוויזיה) הוא מערכ דו-מידי של פיקסלים.
- פיקסל (pixel) משמעו picture element.
- כל פיקסל הוא תחום זעורי שאפשר להאיר בעוצמה משתנה.
- אם הנקודות צפופות דיו, מקבל הרושם של תמונה רציפה.
- במחשב קבוע כל פיקסל מורכב משלוש נקודות אוור בצבעים אדום, ירוק, כחול (RGB).
- קומבינציה של עוצמות של הצבעים הללו מייצגת גוון ספציפי.

ייצוג תמונה בזיכרון

- היצוג בזיכרון תואם למבנה הפיסי של מסך -
מערך דו-מימדי של פיקסלים.
- כל פיקסל הוא שלשה של מספרים שמצוינים
את העוצמה של אדום, ירוק, כחול.
- ערך העוצמה הוא בדרך כלל מספרשלם בין 0
(כבוי) ל-255 (עוצמה מירבית).
- בתמונות שחור-לבן (יש עדין שימושים שלא
צריכים צבע), כל פיקסל הוא מספר בודד.

ייצוג תמונה בקובץ

- כאשר שומרים תמונה בקובץ, בדרך כלל מעבירים אותה לפורמט דחוס.
- יש מגוון של תקנים של קבצי תמונות, חלкам בדיחה משמרת מידע (למשל GIF או PNG), חלкам בדיחה מאבדת מידע (למשל JPEG).
- על מנת לעבוד ולהציג את התמונה יש לשחרר את היצוג כמערך דו-מימדי של פיקסלים.

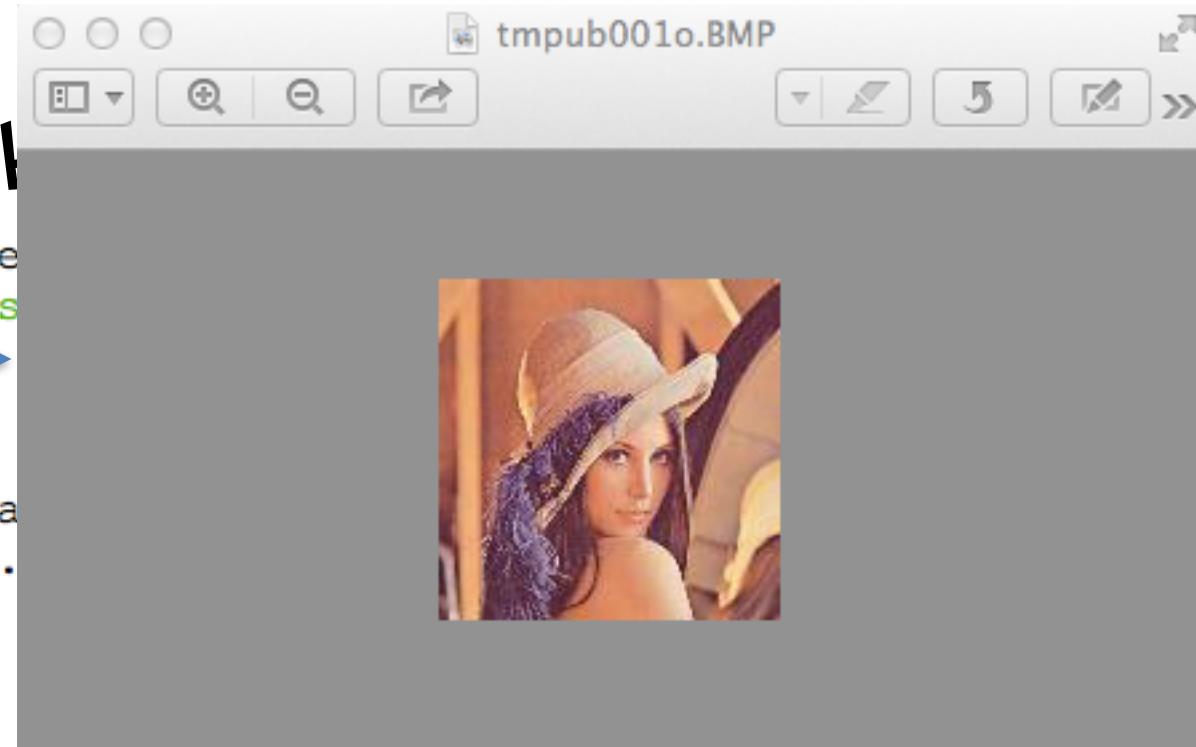
השימוש המזערי שנעשה ב-Pillow

```
>>> from PIL import Image
>>> img = Image.open("/Users/yrabani/lena.jpg")
>>> img.show()
>>> img.size
(128, 128)
>>> px = list(img.getdata())
>>> ar = two_dim(px, img.size[1], img.size[0])
>>> px[0]
(226, 161, 119)
>>> px[129]
(225, 161, 117)
>>> ar[0][0]
(226, 161, 119)
>>> ar[1][1]
(225, 161, 117)
>>> for i in range(len(ar)):
    ar[i][0] = (255, 255, 255)

>>> new_px = one_dim(ar)
>>> img.putdata(new_px)
>>> img.show()
>>>
```

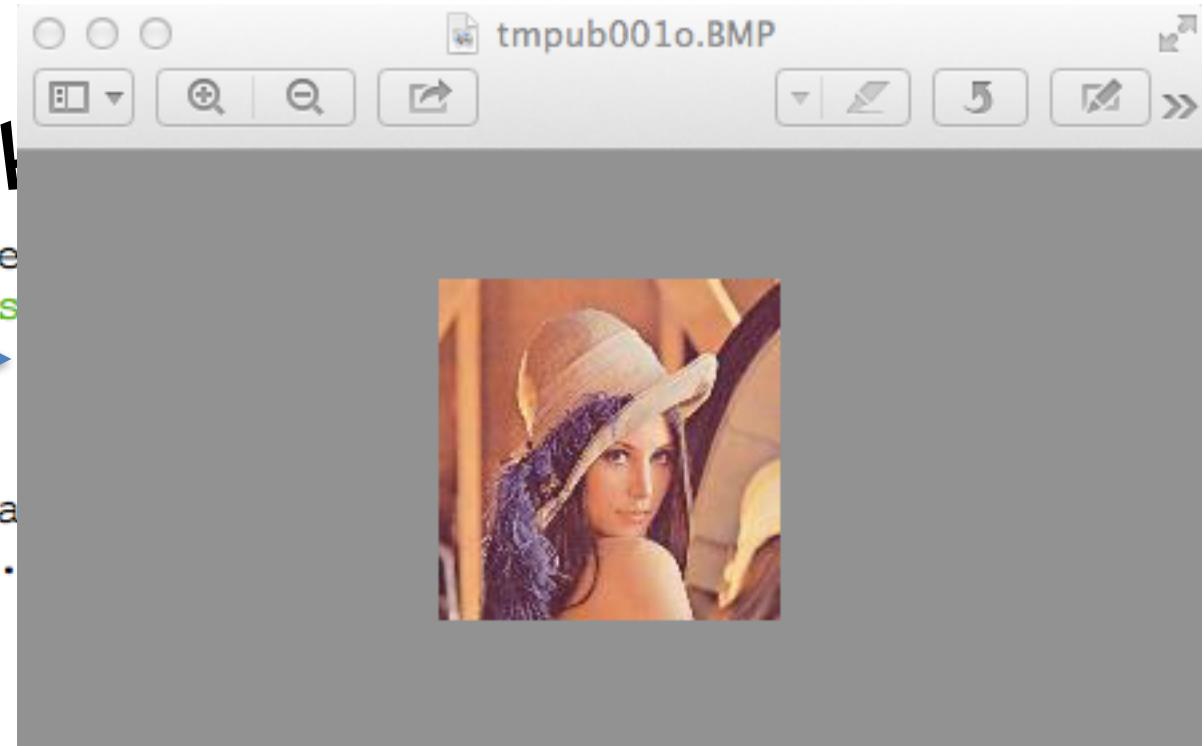
הינה Pillow

```
>>> from PIL import Image  
>>> img = Image.open("/Users/  
>>> img.show() →  
>>> img.size  
(128, 128)  
>>> px = list(img.getdata()  
>>> ar = two_dim(px, img.  
>>> px[0]  
(226, 161, 119)  
>>> px[129]  
(225, 161, 117)  
>>> ar[0][0]  
(226, 161, 119)  
>>> ar[1][1]  
(225, 161, 117)  
>>> for i in range(len(ar)):  
    ar[i][0] = (255, 255, 255)  
  
>>> new_px = one_dim(ar)  
>>> img.putdata(new_px)  
>>> img.show()  
>>>
```



הה ב-Pillow

```
>>> from PIL import Image  
>>> img = Image.open("/Us  
>>> img.show() →  
>>> img.size  
(128, 128)  
>>> px = list(img.getdata()  
>>> ar = two_dim(px, img.  
>>> px[0]  
(226, 161, 119)  
>>> px[129]  
(225, 161, 117)  
>>> ar[0][0]  
(226, 161, 119)  
>>> ar[1][1]  
(225, 161, 117)  
>>> for i in range(len(ar)):  
    ar[i][0] = (255, 255, 255)
```



לנה סודרברג, דר האמצע של פלייבוי, נובמבר 1972

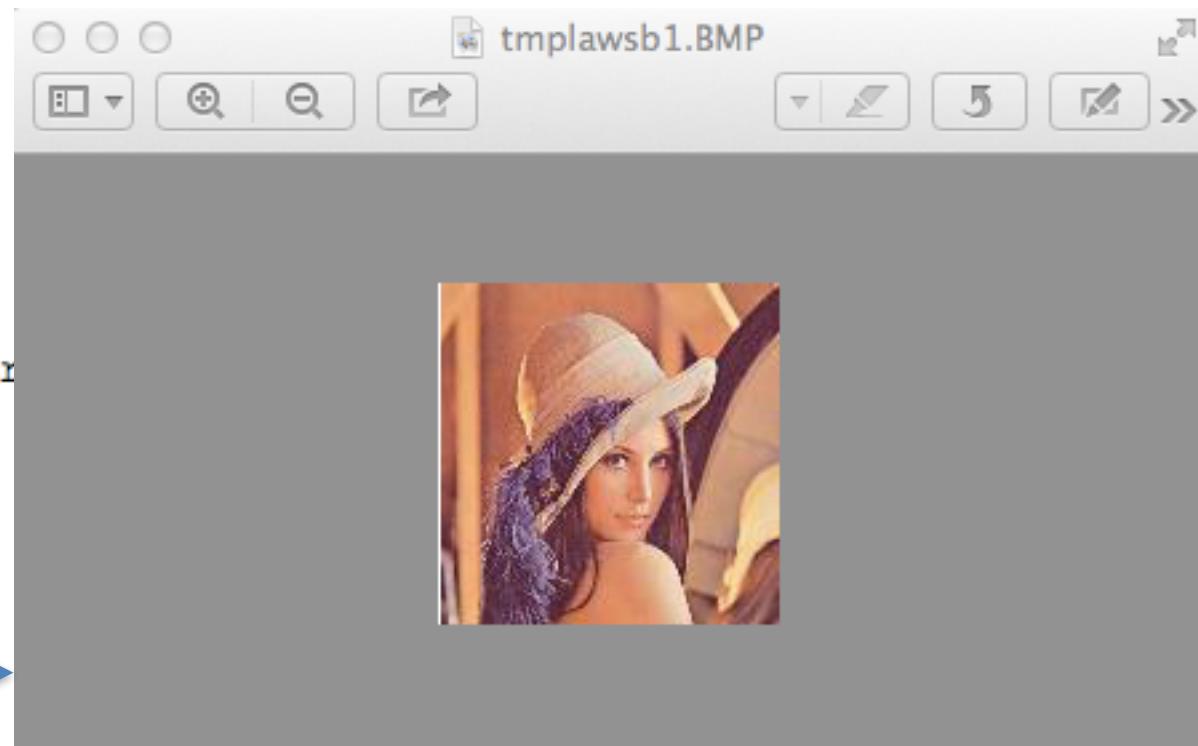
```
>>> new_px = one_dim(ar)  
>>> img.putdata(new_px)  
>>> img.show()  
>>>
```

השימוש המזערי שנעשה ב-Pillow

```
>>> from PIL import Image
>>> img = Image.open("/Users/yrabani/lena.jpg")
>>> img.show()
>>> img.size
(128, 128)
>>> px = list(img.getdata())
>>> ar = two_dim(px, img.size[1], img.size[0])
>>> px[0]
(226, 161, 119)
>>> px[129]
(225, 161, 117)
>>> ar[0][0]
(226, 161, 119)
>>> ar[1][1]
(225, 161, 117)
>>> for i in range(len(ar)):
    ar[i][0] = (255, 255, 255)

>>> new_px = one_dim(ar)
>>> img.putdata(new_px)
>>> img.show()
>>>
```

השימוש המזררי שנעשה ב-Pillow



פונקציות העזר

```
>>> def two_dim(px, nrows, ncols):
    if len(px) != nrows * ncols:
        return None
    else:
        return [[px[ncols * i + j] for j in range(ncols)] for i in range(nrows)]

>>> def one_dim(ar):
    return [e for row in ar for e in row]

>>>
```

התמורות פשוטות

- נסנה את התמונה על ידי החלפת שלושת הערכים של פיקסל בערך פונקציה f על השלשה זו.
- למשל: תמונה נגטיב מתתקבלת מההמרה הבאה: $(r,g,b) \rightarrow (255-r, 255-g, 255-b)$

קצת קיד

```
>>> def get_image(fname):
    img = Image.open(fname)
    px = list(img.getdata())
    ar = two_dim(px, img.size[1], img.size[0])
    return img, ar

>>> def put_image(img, ar):
    new_img = img.copy()
    px = one_dim(ar)
    new_img.putdata(px)
    return new_img

>>> def simple_transform(ar, f):
    return [[f(ar[i][j]) for j in range(len(ar[i]))] for i in range(len(ar))]

>>>
```

דוגמאות

```
>>> (img, ar) = get_image("/Users/yrabani/IMG_3552.jpg")
>>> img.show()
>>> new_ar = simple_transform(ar, lambda pix: tuple(255-x for x in pix))
>>> new_img = put_image(img, new_ar)
>>> new_img.show()
>>> (img, ar) = get_image("/Users/yrabani/IMG_1730.jpg")
>>> img.show()
>>> new_ar = simple_transform(ar, lambda pix: tuple(int(255*(x/255)**2) for x in pix))
>>> new_img = put_image(img, new_ar)
>>> img.show()
>>> new_img.show()
>>>
```

```
>>> (img, ar) = get_image("C:/Users/Asus/Desktop/Snowman.bmp")
>>> img.show() →
>>> new_ar = simple_transform(img, 100, 100)
>>> new_img = put_image(img, new_ar)
>>> new_img.show()
>>> (img, ar) = get_image("C:/Users/Asus/Desktop/Snowman.bmp")
>>> img.show()
>>> new_ar = simple_transform(img, 100, 100)
>>> new_img = put_image(img, new_ar)
>>> img.show()
>>> new_img.show()
>>>
```



r x in pix))

דוגמאות

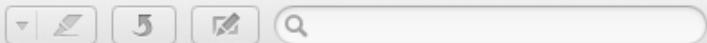
```
>>> (img, ar) = get_image("/Users/yrabani/IMG_3552.jpg")
>>> img.show()
>>> new_ar = simple_transform(ar, lambda pix: tuple(255-x for x in pix))
>>> new_img = put_image(img, new_ar)
>>> new_img.show()
>>> (img, ar) = get_image("/Users/yrabani/IMG_1730.jpg")
>>> img.show()
>>> new_ar = simple_transform(ar, lambda pix: tuple(int(255*(x/255)**2) for x in pix))
>>> new_img = put_image(img, new_ar)
>>> img.show()
>>> new_img.show()
>>>
```

```
>>> (img, ar) = get_image("/User  
>>> img.show()  
>>> new_ar = simple_transform(ar  
>>> new_img = put_image(img, new  
>>> new_img.show() —————→  
>>> (img, ar) = get_image("/User  
>>> img.show()  
>>> new_ar = simple_transform(ar  
>>> new_img = put_image(img, new  
>>> img.show()  
>>> new_img.show()  
>>>
```



דוגמאות

```
>>> (img, ar) = get_image("/Users/yrabani/IMG_3552.jpg")
>>> img.show()
>>> new_ar = simple_transform(ar, lambda pix: tuple(255-x for x in pix))
>>> new_img = put_image(img, new_ar)
>>> new_img.show()
>>> (img, ar) = get_image("/Users/yrabani/IMG_1730.jpg")
>>> img.show()
>>> new_ar = simple_transform(ar, lambda pix: tuple(int(255*(x/255)**2) for x in pix))
>>> new_img = put_image(img, new_ar)
>>> img.show()
>>> new_img.show()
>>>
```



99
מילקי פלאני גבינה

Napca
99

99
סואיגיך

93
טולביה עוגת גבינה

נורילג רולעפנ'

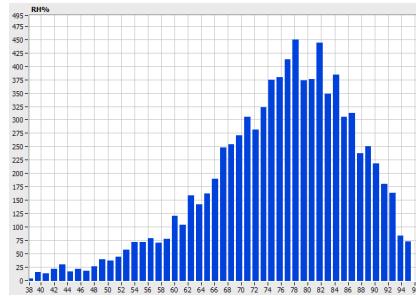
דוגמאות

```
>>> (img, ar) = get_image("/Users/yrabani/IMG_3552.jpg")
>>> img.show()
>>> new_ar = simple_transform(ar, lambda pix: tuple(255-x for x in pix))
>>> new_img = put_image(img, new_ar)
>>> new_img.show()
>>> (img, ar) = get_image("/Users/yrabani/IMG_1730.jpg")
>>> img.show()
>>> new_ar = simple_transform(ar, lambda pix: tuple(int(255*(x/255)**2) for x in pix))
>>> new_img = put_image(img, new_ar)
>>> img.show()
>>> new_img.show()
>>>
```



דוגמאות

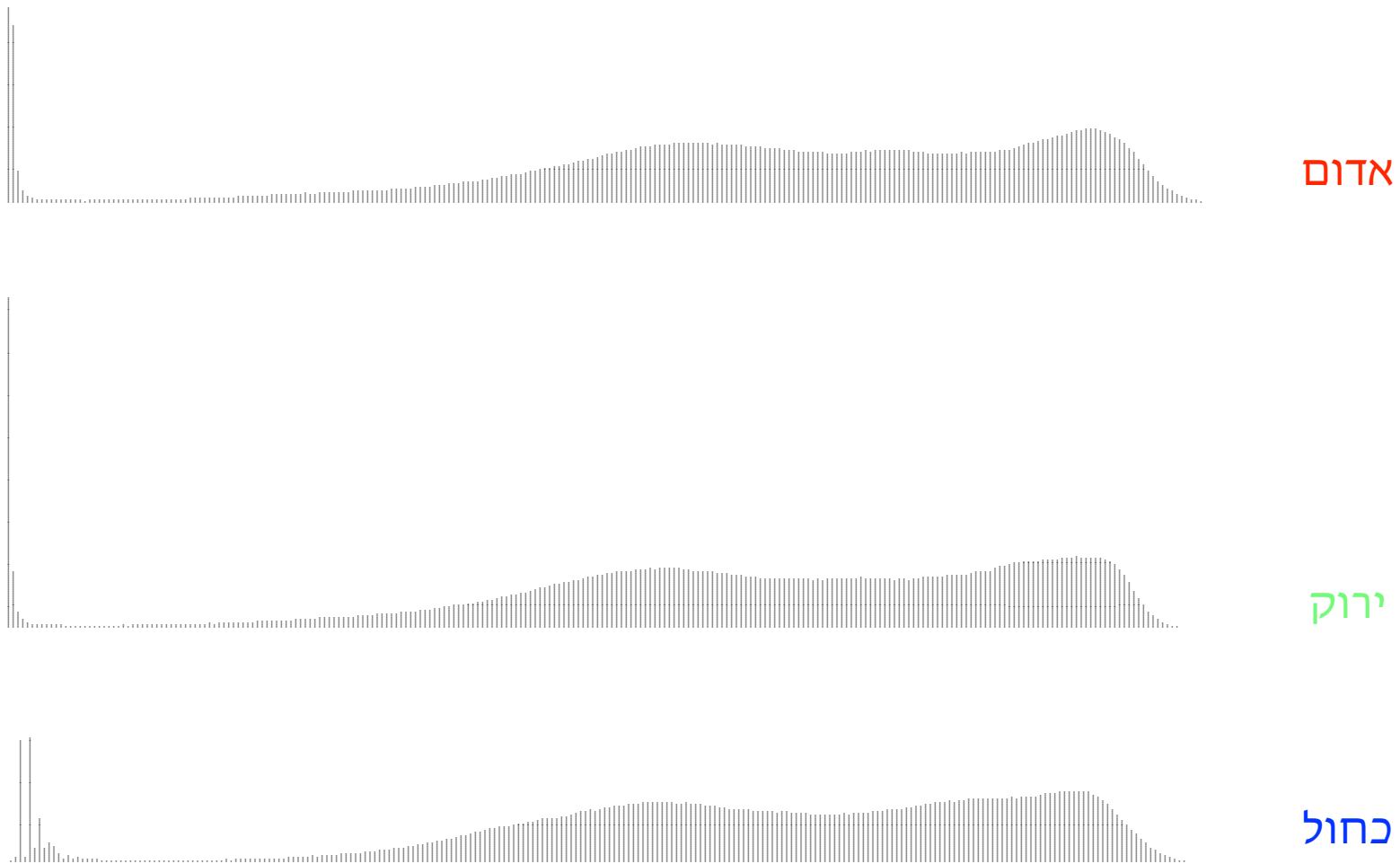
```
>>> (img, ar) = get_image("/Users/yrabani/IMG_3552.jpg")
>>> img.show()
>>> new_ar = simple_transform(ar, lambda pix: tuple(255-x for x in pix))
>>> new_img = put_image(img, new_ar)
>>> new_img.show()
>>> (img, ar) = get_image("/Users/yrabani/IMG_1730.jpg")
>>> img.show()
>>> new_ar = simple_transform(ar, lambda pix: tuple(int(255*(x/255)**2) for x in pix))
>>> new_img = put_image(img, new_ar)
>>> img.show()
>>> new_img.show()
>>>
```

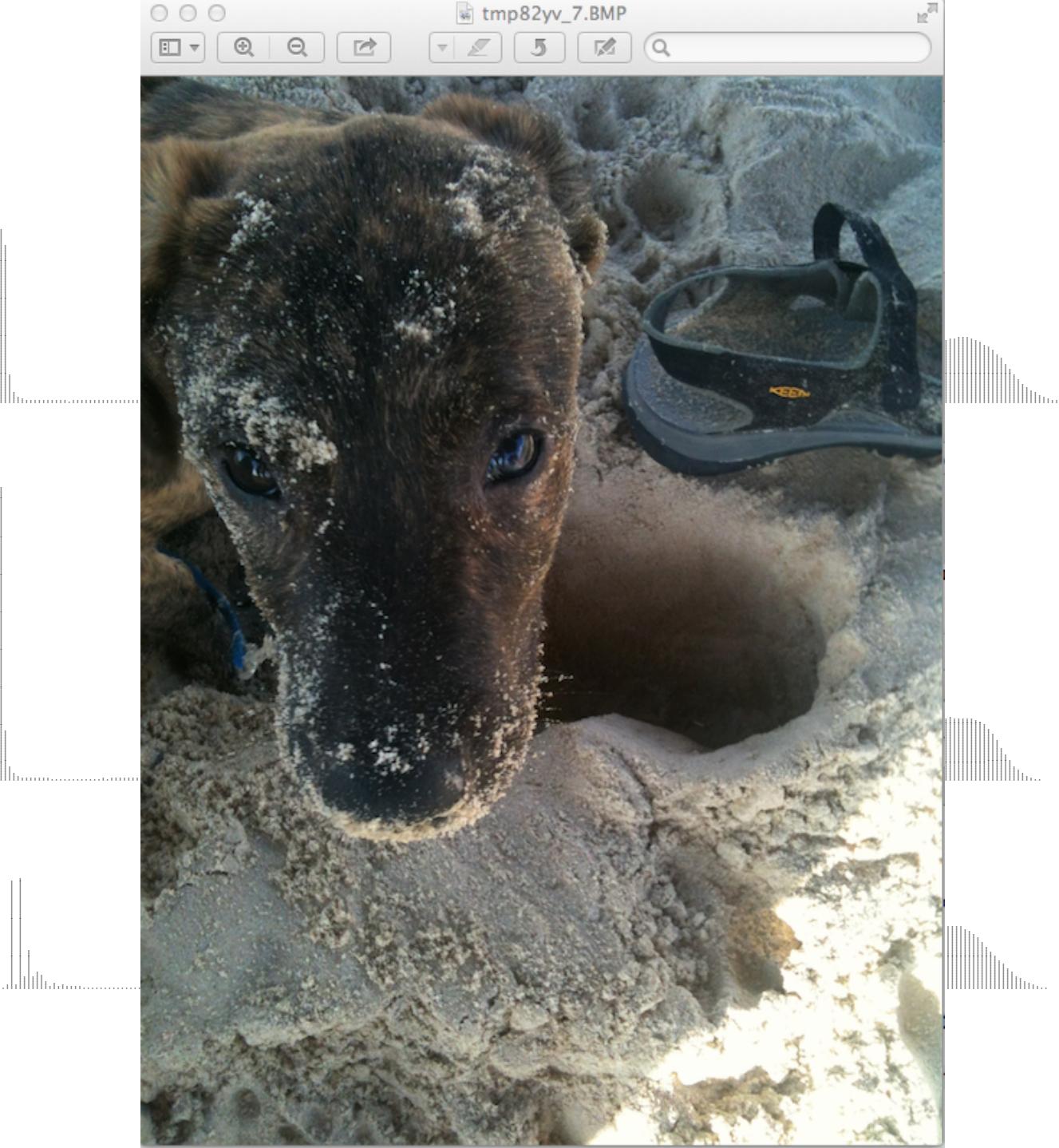


חישוב היסטוגרماה

- ההיסטוגרמה מצינית לכל ערך בין 0 ל-255 כמה פיקסלים יש עם ערך זה.
- עבור תמונה צבעונית נחשב בנפרד שלוש הסטוגרמות לכל אחד מהצבעים.
- לרוב נרצה לנורמל לאחוזים או הסתברויות במקום מונחים מוחלטים.

דוגמה



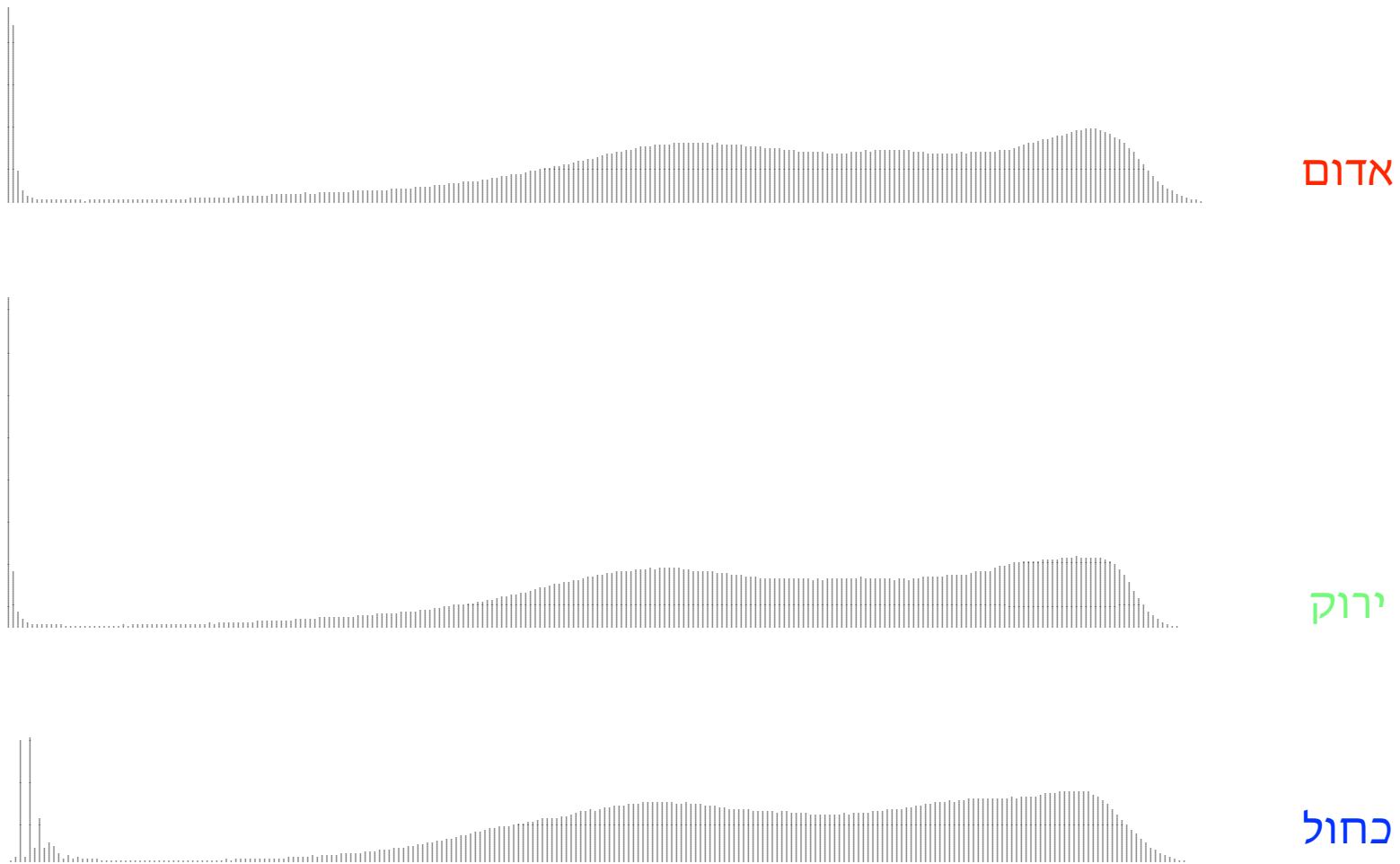


אדום

ירוק

כחול

דוגמה



שיטת ההיסטוגרמה

- ההתמרה משנה את הגוונים כך שתיווצר ההיסטוגרמה "שטוחה" ככל האפשר.
- פיקסלים עם אותו גוון נשארים עם אותו גוון.
- נסמן ב- $h(p)$ את שיעור הפיקסלים עם ערך p עבור צבע ספציפי).
- נסמן ב- $cdf(p)$ את הסתברות המצרפית, כלומר, $h(0) + h(1) + \dots + h(p)$.
- הערך החדש של פיקסל שערכו המקורי הוא $.int(255 * cdf(p))$

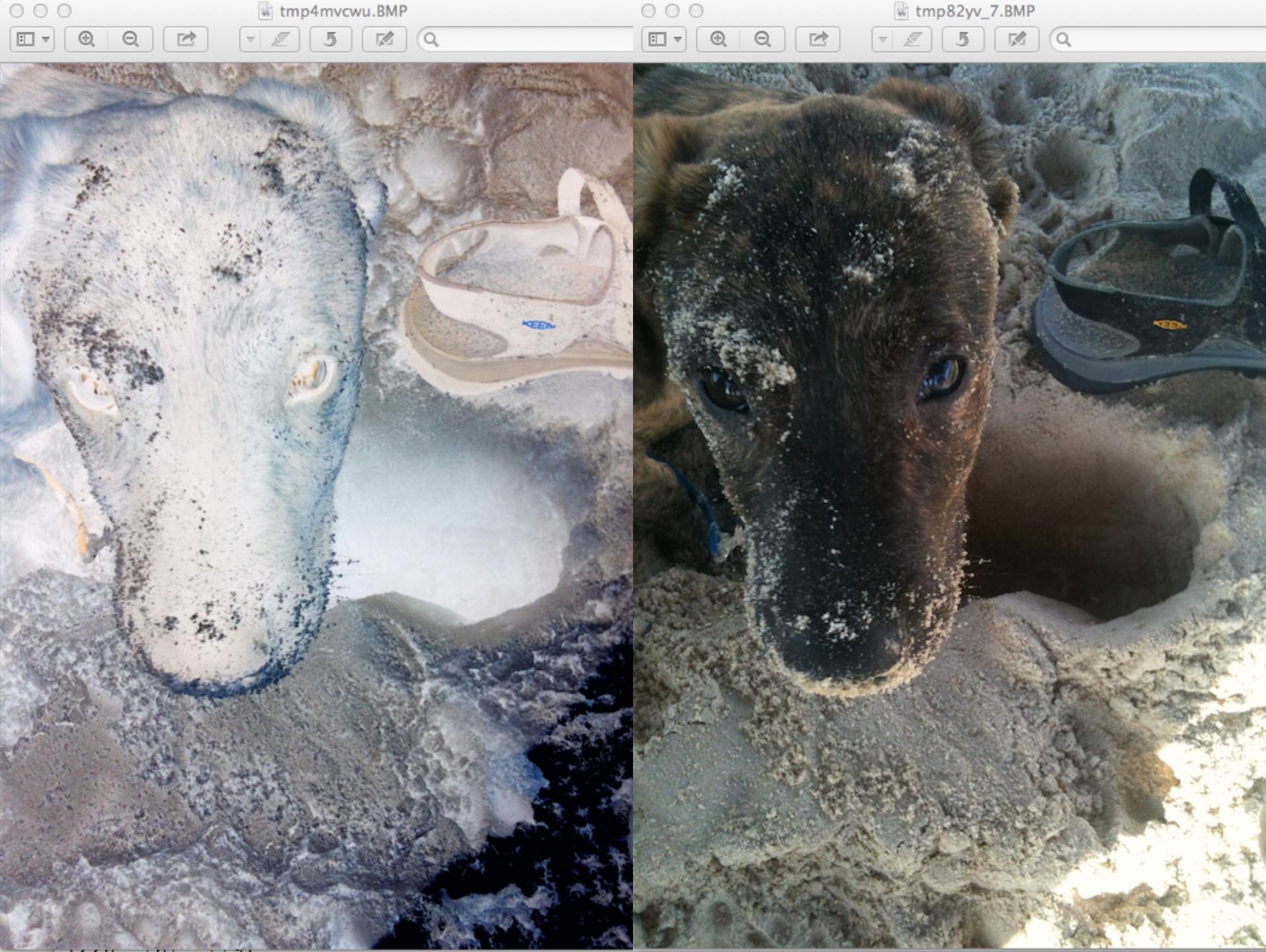
תִּפְעָלָה

```
>>> class Picture:
    def __init__(self, pixels):
        self.pixels = [row.copy() for row in pixels]
        self.nrows = len(pixels)
        self.ncols = len(pixels[0])
    def simple_transform(self, f):
        pixels = [[f(self.pixels[i][j]) for j in range(self.ncols)] for i in range(self.nrows)]
        return Picture(pixels)
    def histogram(self, rgb, nvals = 256):
        hist = [0] * nvals
        for row in ar:
            for pix in row:
                hist[pix[rgb]] += 1
        return hist
    def histeq_transform(self):
        cdf = 3 * [None]
        for rgb in range(3):
            hist = self.histogram(rgb)
            dist = [h / sum(hist) for h in hist]
            cdf[rgb] = list(itertools.accumulate(dist))
        return self.simple_transform(lambda px: tuple(int(255*cdf[px[rgb]][px[rgb]])) for px in range(3)))
```

TIP

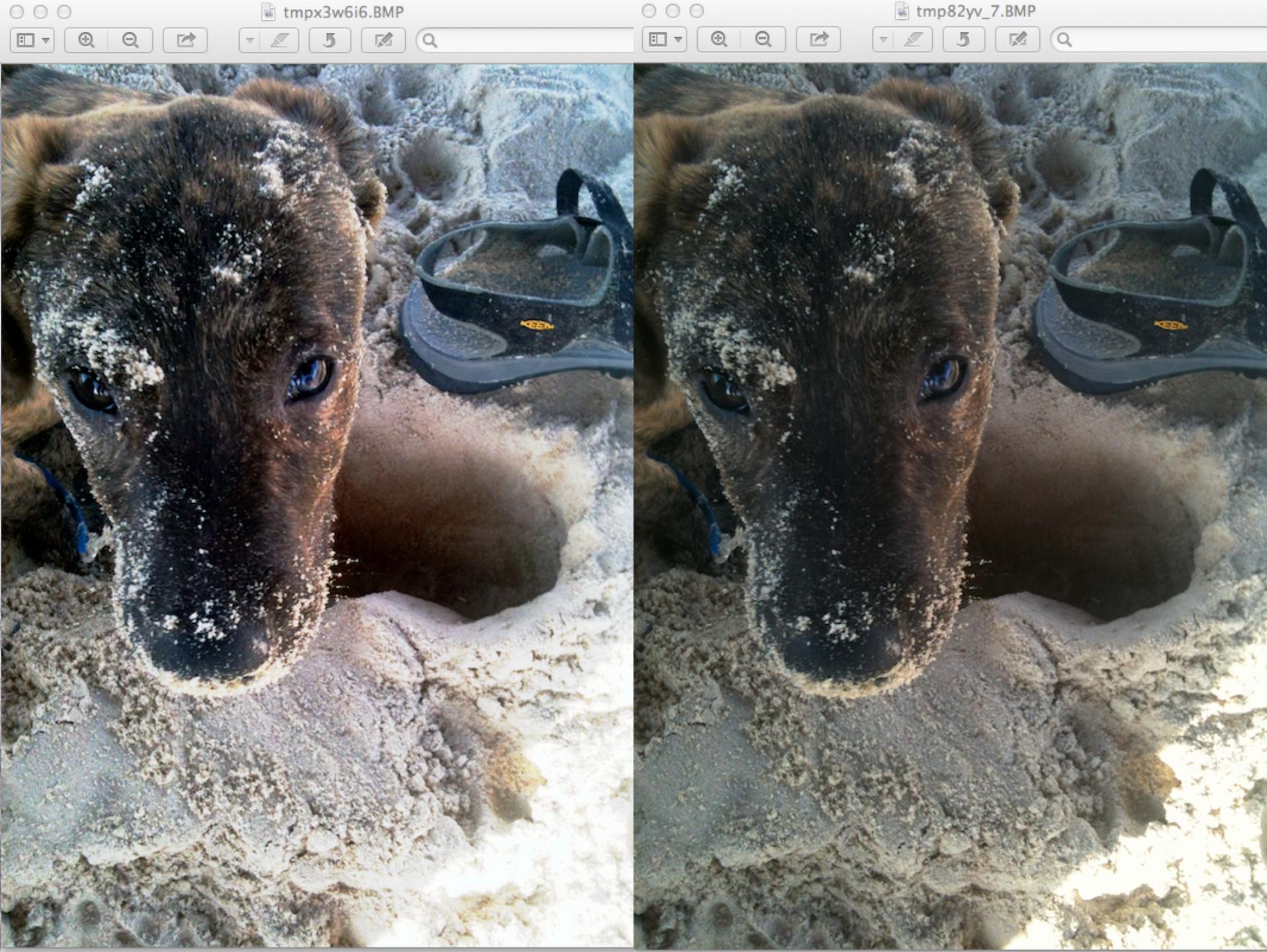
```
>>> class Picture:
    def __init__(self, pixels):
        self.pixels = [row.copy() for row in pixels]
        self.nrows = len(pixels)
        self.ncols = len(pixels[0])
    def simple_transform(self, f):
        pixels = [[f(self.pixels[i][j]) for j in range(self.ncols)] for i in range(self.nrows)]
        return Picture(pixels)
    def histogram(self, rgb, nvals = 256):
        hist = [0] * nvals
        for row in ar:
            for pix in row:
                hist[pix[rgb]] += 1
        return hist
    def histeq_transform(self):
        cdf = 3 * [None]
        for rgb in range(3):
            hist = self.histogram(rgb)
            dist = [h / sum(hist) for h in hist]
            cdf[rgb] = list(itertools.accumulate(dist))
        return self.simple_transform(lambda px:
```





tmp4mvwu.BMP

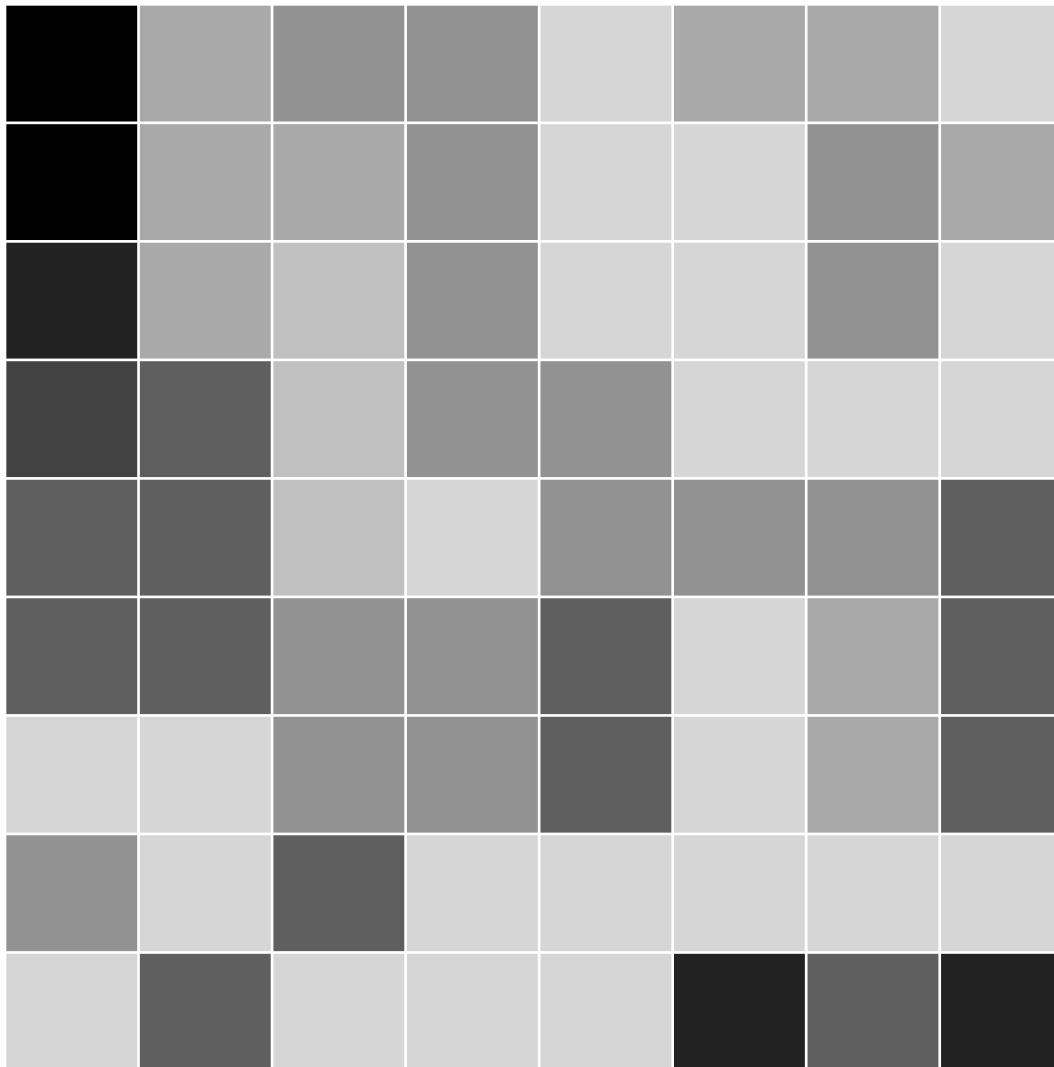
tmp82yv_7.BMP



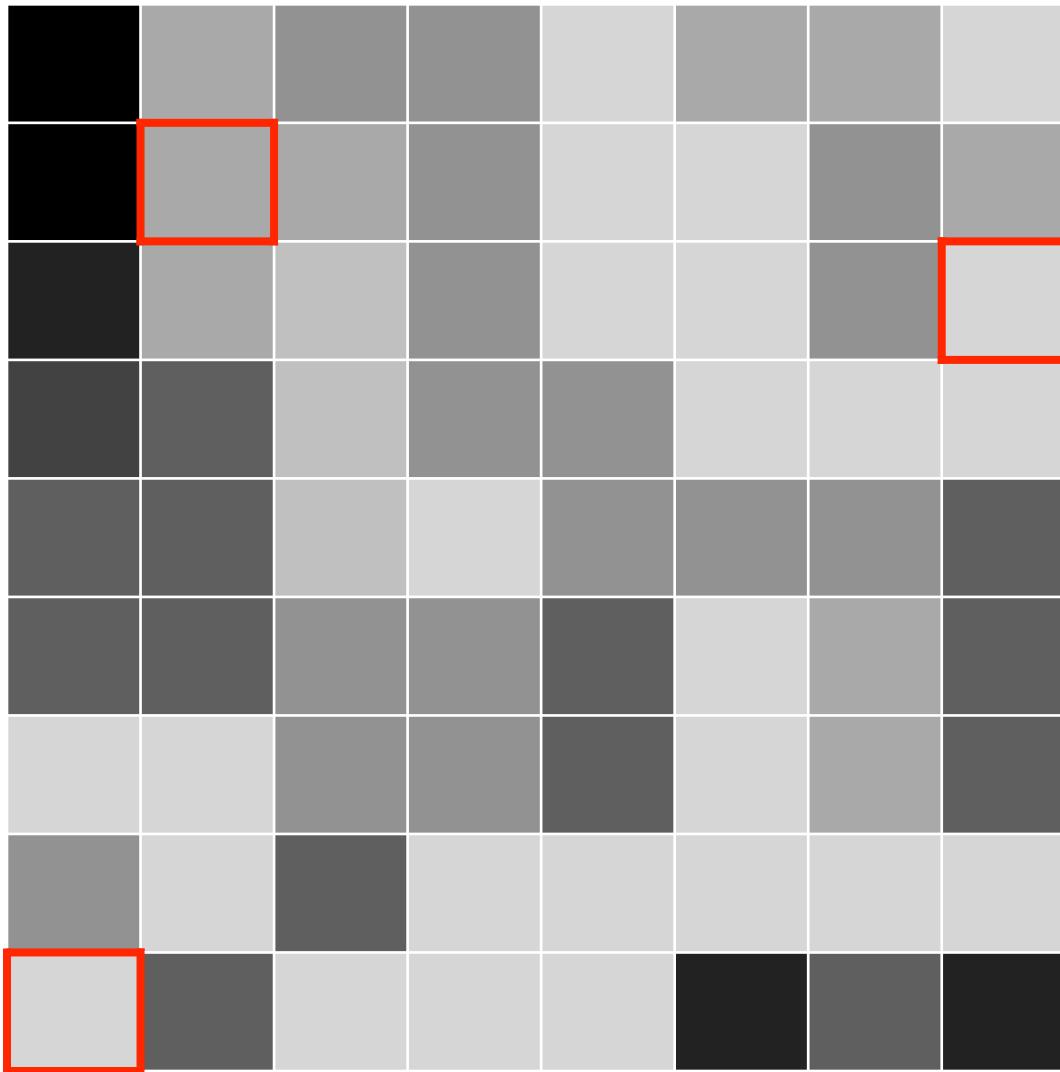
טשטוש תמונה

- על מנת לטשטש, נחליף את הערך של כל פיקסל בmäßig ערך הערכים של הפיקסלים בסביבתו.
- אנחנו ניקח ריבוע של 3×3 שמרכזו בפיקסל ונחליף את ערך הפיקסל (עבור כל צבע) בממוצע הערך בריבוע.
- הפעלה חוזרת של התמרה מושפעת יותר.
- יש לשים לב לקצות התמונה, שם הריבוע שלנו חורג מגבולות התמונה.

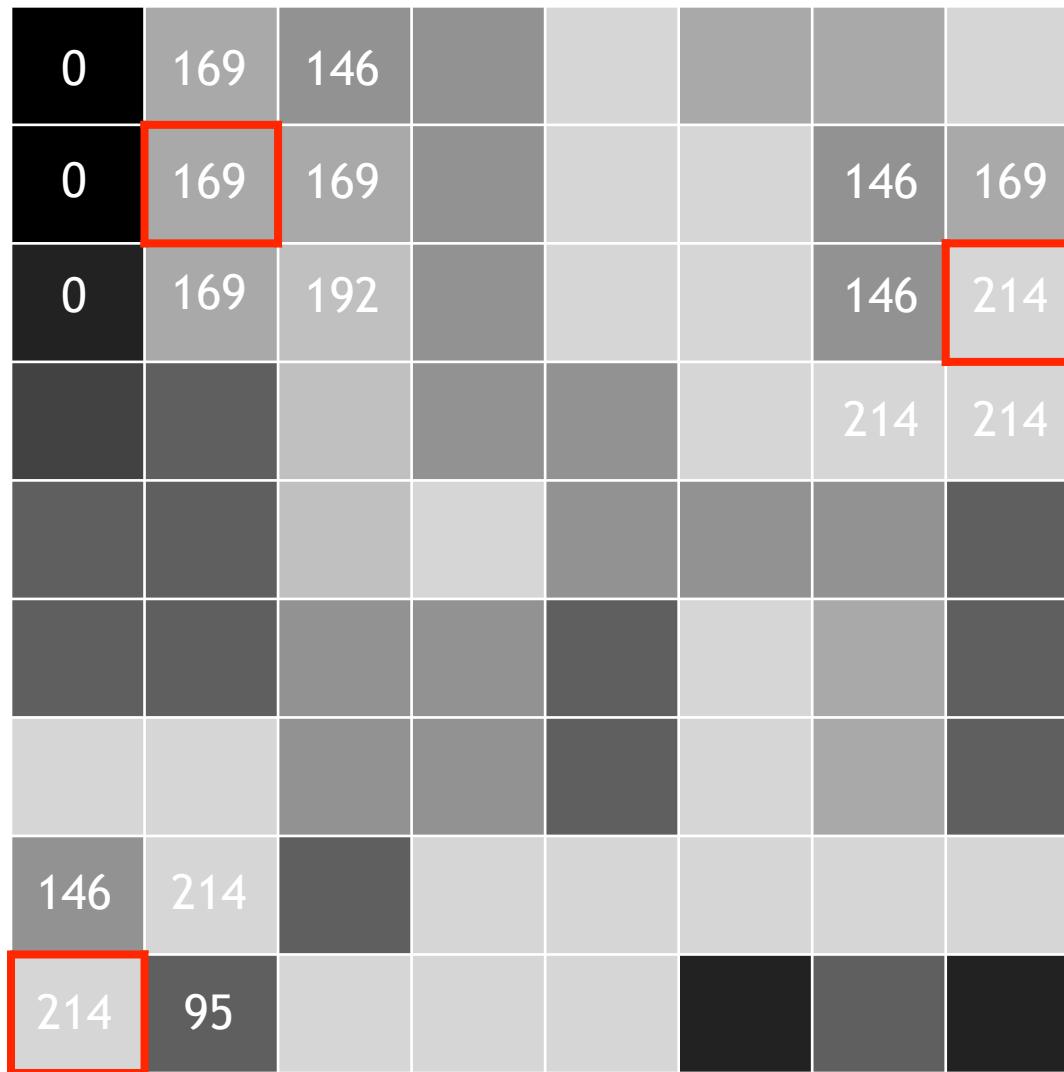
טשטוש תמונה



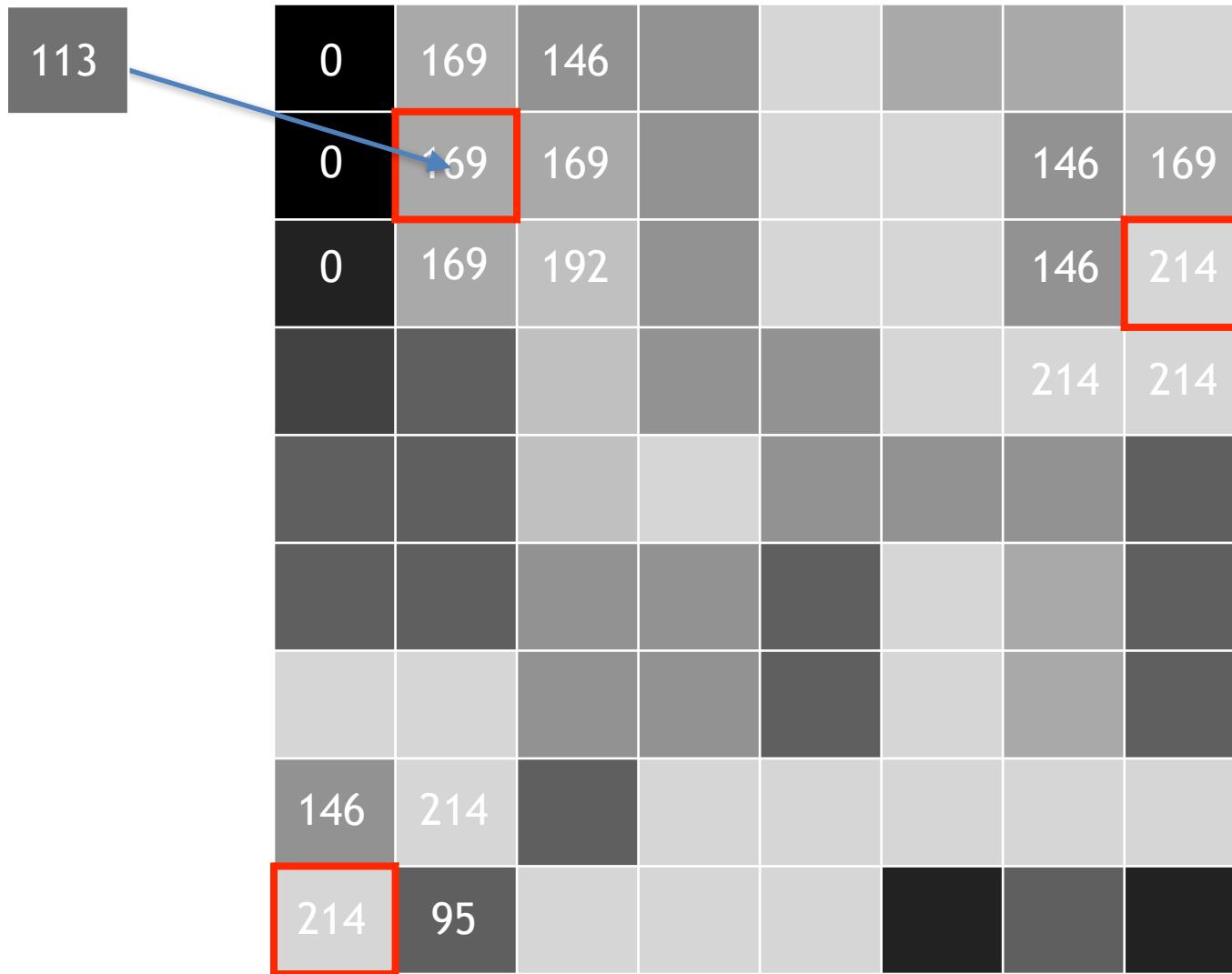
טשטוש תמונה



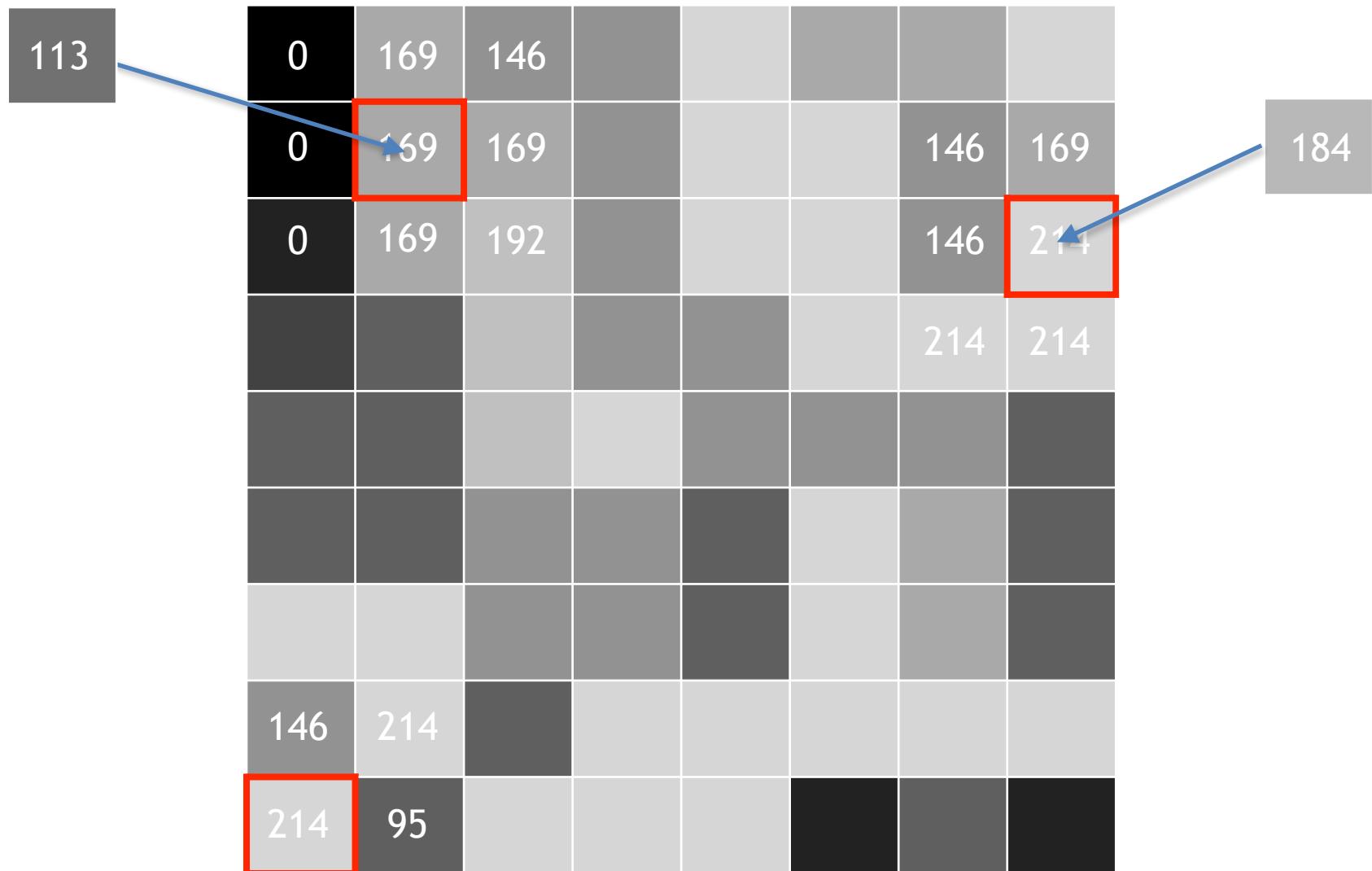
טשטוש תמונה



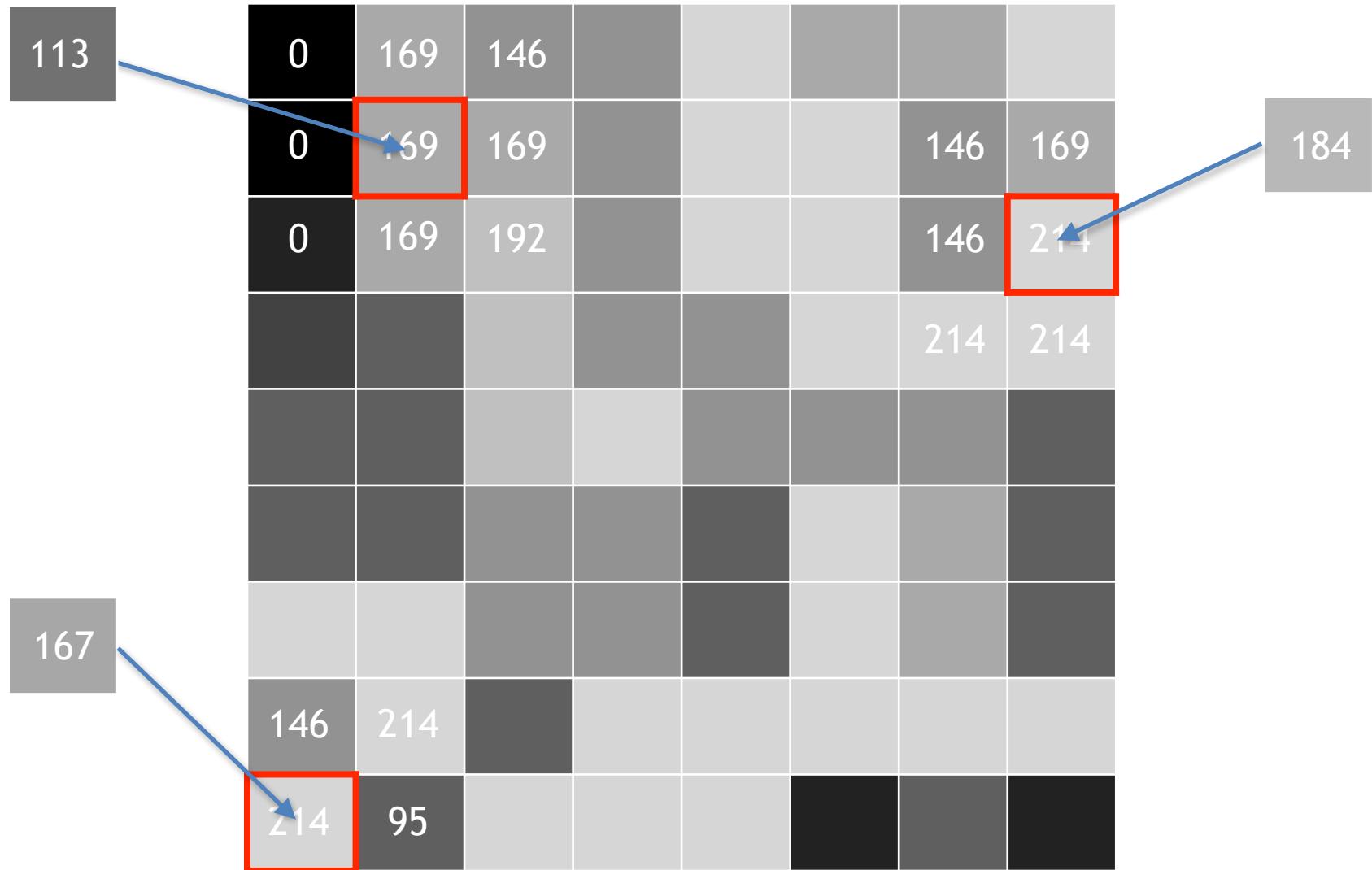
טשטוש תמונה



טשטוש תמונה



טשטוש תמונה



קוד מושך ב-Python

```
class Picture:
    def __init__(self, pixels):
        self.pixels = [row.copy() for row in pixels]
        self.nrows = len(self.pixels)
        self.ncols = len(self.pixels[0])
    def simple_transform(self, f):
        pixels = [[f(self.pixels[i][j]) for j in range(self.ncols)] for i in range(self.nrows)]
        return Picture(pixels)
    def histogram(self, rgb, nvals = 256):
        hist = [0] * nvals
        for row in ar:
            for pix in row:
                hist[pix[rgb]] += 1
        return hist
    def histeq_transform(self):
        cdf = 3 * [None]
        for rgb in range(3):
            hist = self.histogram(rgb)
            dist = [h / sum(hist) for h in hist]
            cdf[rgb] = list(itertools.accumulate(dist))
        return self.simple_transform(lambda px: tuple(int(255*cdf[px[rgb]][px[rgb]])) for px in range(3)))
    def blur(self, env = 1):
        # replace each pixel i,j by the average over i+env,j+env
        pixels = [row.copy() for row in self.pixels]
        for i in range(len(pixels)):
            for j in range(len(pixels[0])):
                pixels[i][j] = average_over_env(self.pixels, i, j, env)
        return Picture(pixels)
```

טשטוש פיקסל

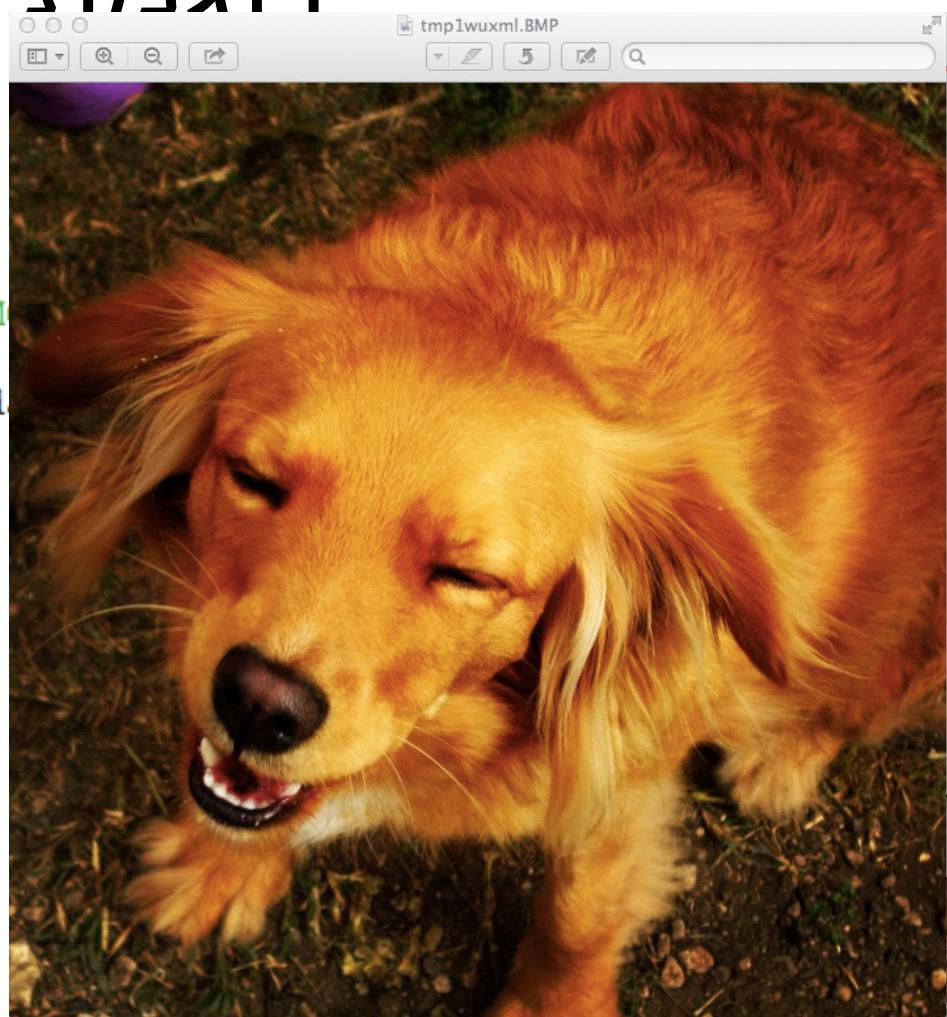
```
> def average_over_env(ar, i, j, env):
    s = [0] * 3
    n = 0
    for k in range(-env, env):
        for l in range(-env, env):
            if i+k >= 0 and i+k < len(ar) and j+l >= 0 and j+l < len(ar[0]):
                n += 1
                for rgb in range(3):
                    s[rgb] += ar[i+k][j+l][rgb]
    return (int(s[0] / n), int(s[1] / n), int(s[2] / n))
```

דוגמת הרצה

```
>>> from PIL import Image
>>> import itertools
>>> img = Image.open("/Users/yrabani/IMG_0564.jpg")
>>> img.show()
>>> pic = Picture(two_dim(list(img.getdata()), img.size[1], img.size[0]))
>>> pic1 = pic.blur(10)
>>> img1 = img.copy()
>>> img1.putdata(one_dim(pic1.pixels))
>>> img1.show()
>>> pic2 = pic1.blur(10)
>>> img2 = img.copy()
>>> img2.putdata(one_dim(pic2.pixels))
>>> img2.show()
>>>
```

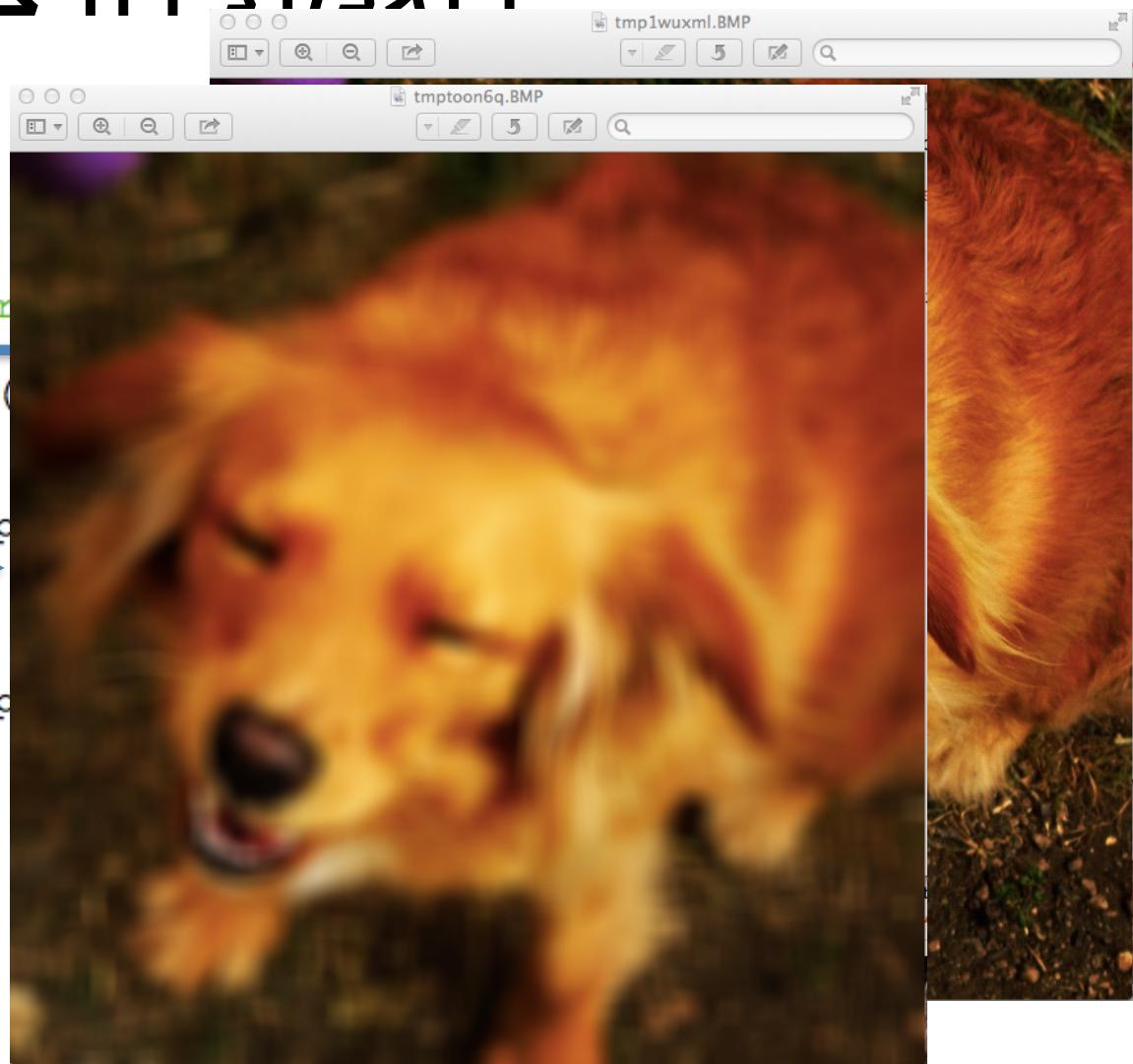
דוגמת הרצה

```
>>> from PIL import Image  
>>> import itertools  
>>> img = Image.open("/Users/yrabani/IM  
>>> img.show() —————→  
>>> pic = Picture(two_dim(list(img.getd  
>>> pic1 = pic.blur(10)  
>>> img1 = img.copy()  
>>> img1.putdata(one_dim(pic1.pixels))  
>>> img1.show()  
>>> pic2 = pic1.blur(10)  
>>> img2 = img.copy()  
>>> img2.putdata(one_dim(pic2.pixels))  
>>> img2.show()  
>>>
```



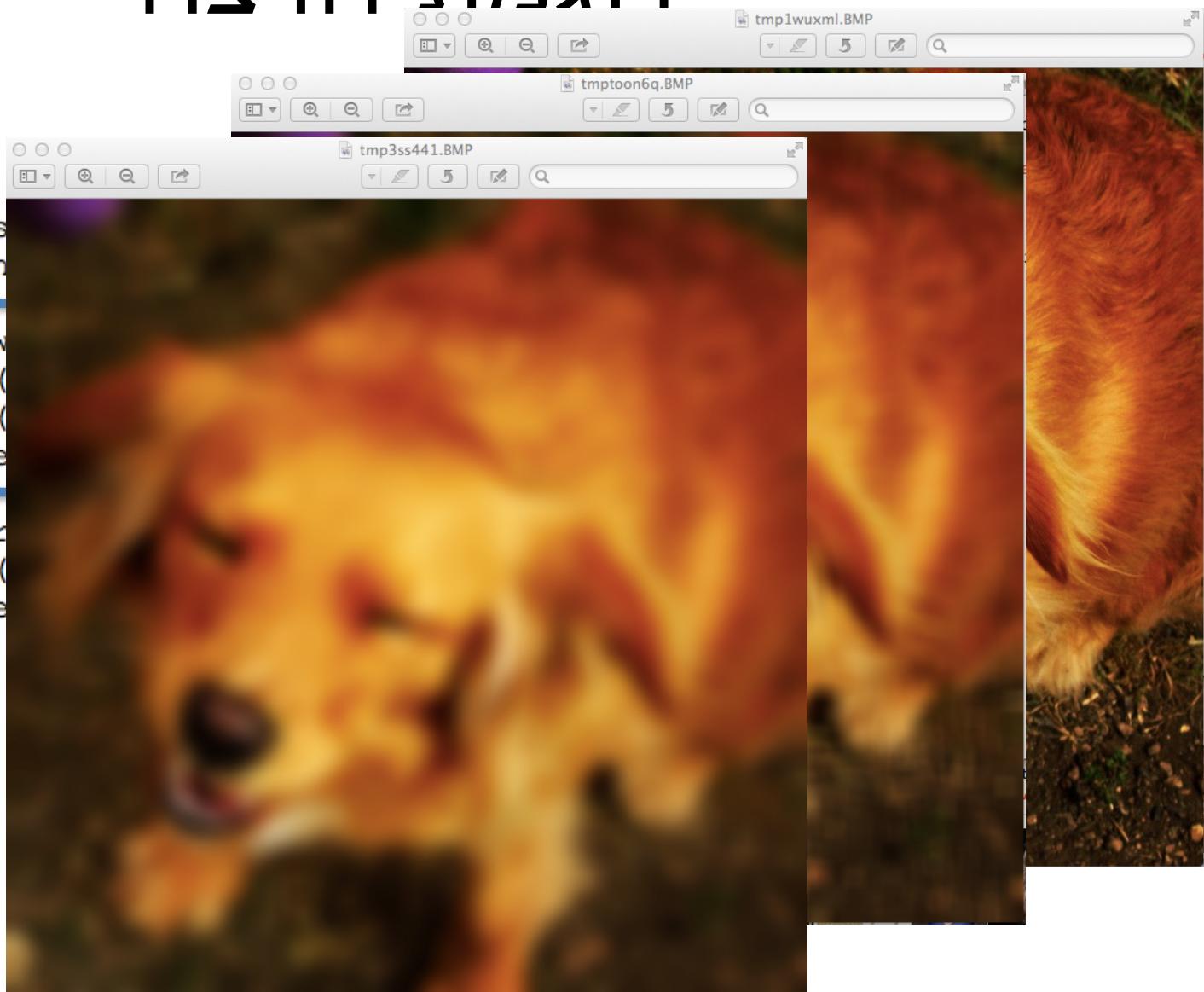
דוגמת הרצה

```
>>> from PIL import Image  
>>> import itertools  
>>> img = Image.open("/Users/yr...  
>>> img.show() ——————  
>>> pic = Picture(two_dim(list(...  
>>> pic1 = pic.blur(10)  
>>> img1 = img.copy()  
>>> img1.putdata(one_dim(pic1.p...  
>>> img1.show() ——————→  
>>> pic2 = pic1.blur(10)  
>>> img2 = img.copy()  
>>> img2.putdata(one_dim(pic2.p...  
>>> img2.show()  
>>>
```



דוגמת הרצה

```
>>> from PIL import  
>>> import itertools  
>>> img = Image.open  
>>> img.show() —————  
>>> pic = Picture(tw  
>>> pic1 = pic.blur()  
>>> img1 = img.copy()  
>>> img1.putdata(one  
>>> img1.show() —————  
>>> pic2 = pic1.blur()  
>>> img2 = img.copy()  
>>> img2.putdata(one  
>>> img2.show() —————  
>>>
```



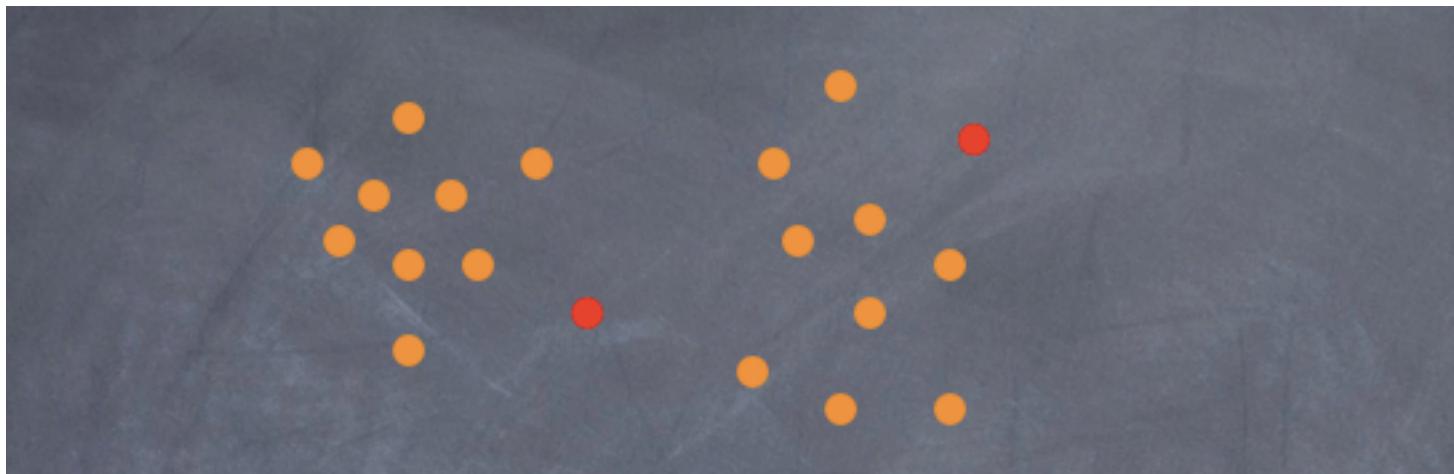
k-מוצעים

- לעתים צריך לחלק אוסף גדול של נתונים לקבוצות (אשכולות) של נתונים דומים.
- נניח שכל נתון מיוצג על ידי וקטור, ומספר הקבוצות הנדרש ידוע (נסמננו ב-k).
- נמדד דמיון באופנו הבא:
 - מחיר קבוצה: סכום ריבועי המרחקים למרכז הבודד (הממוצע) של הקבוצה.
 - מחיר חלוקה: סכום מחירי הקבוצות.
 - נפש חלוקה שמחירה מינימלי.

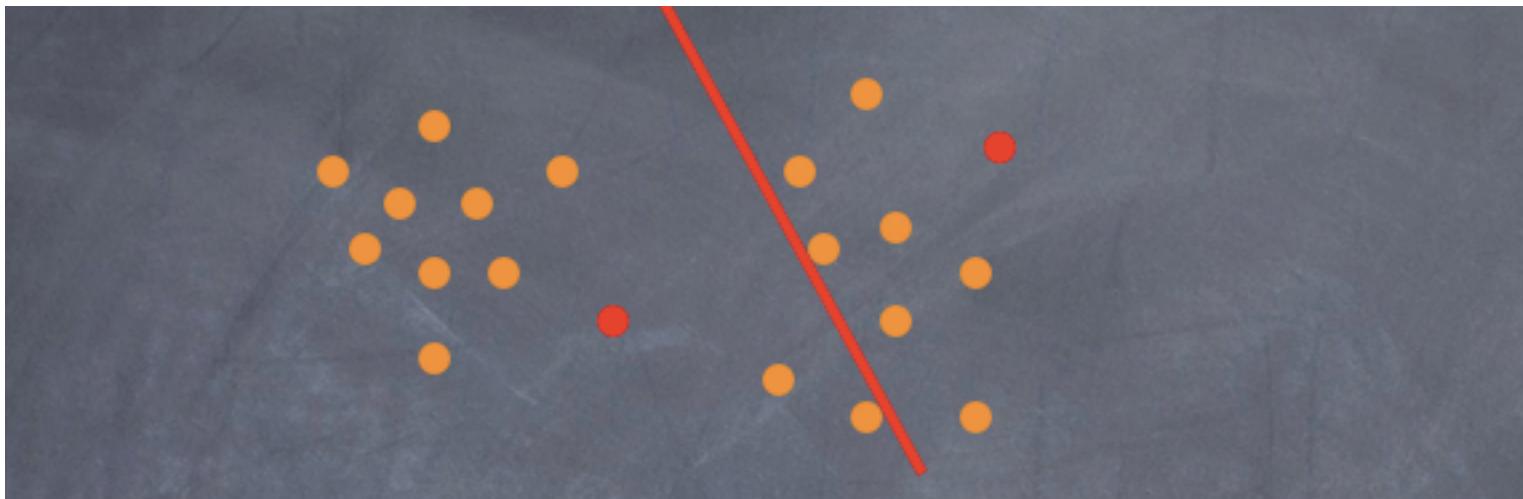
האיטרציה של Lloyd

- במיד גובה זו בעיה קשה, לא טריוויאלית גם במיד נמוך.
- איטרצית Lloyd מתכונסת למינימום מקומי (או דוקא פיתרון אופטימלי) והיא מאוד פופולרית.
- מתחילה מ-k מרכזים כלשהם.
- בכל איטרציה משיכים כל ווקטור למרכז הקרוב ביותר לו ואז מזיכים כל מרכז לموقع של הווקטורים שעויינו לו.

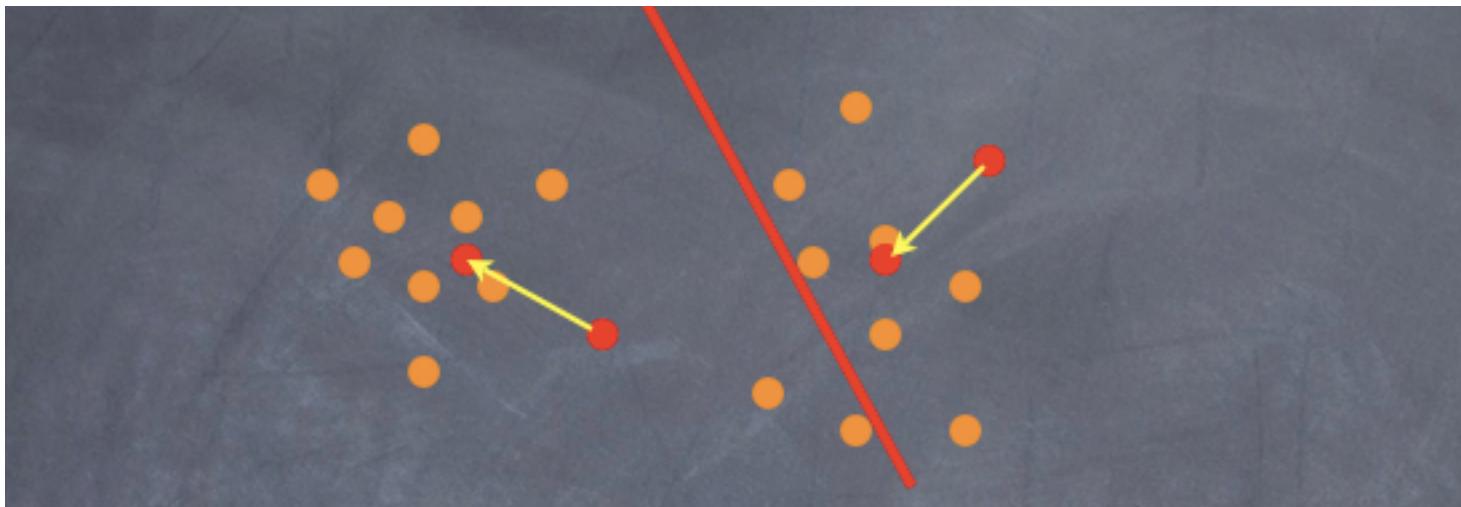
דוגמה



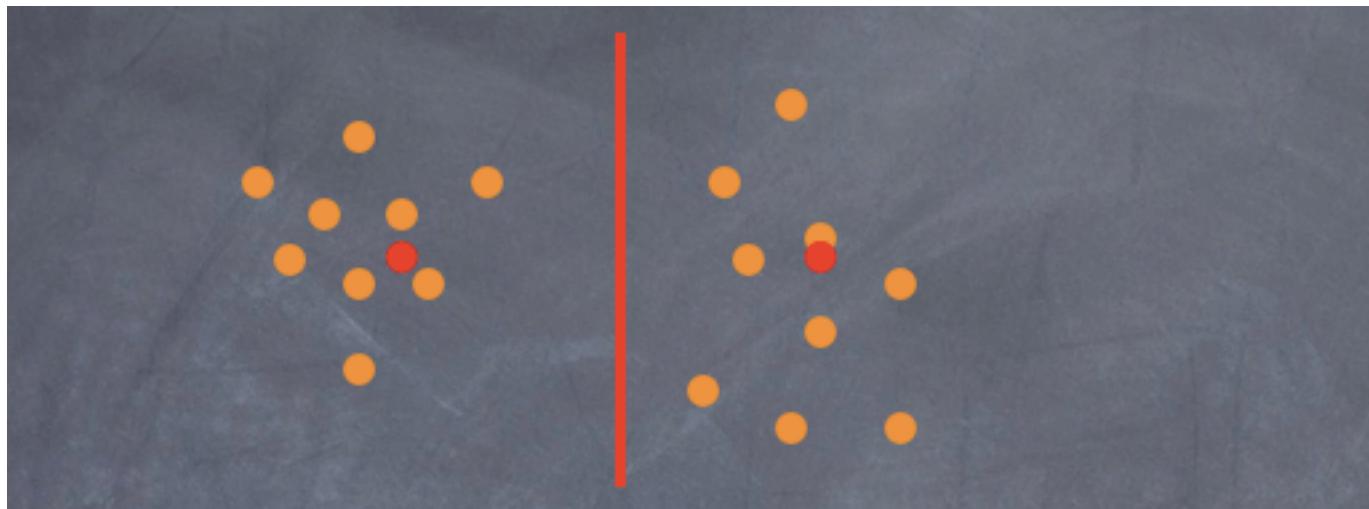
דוגמה



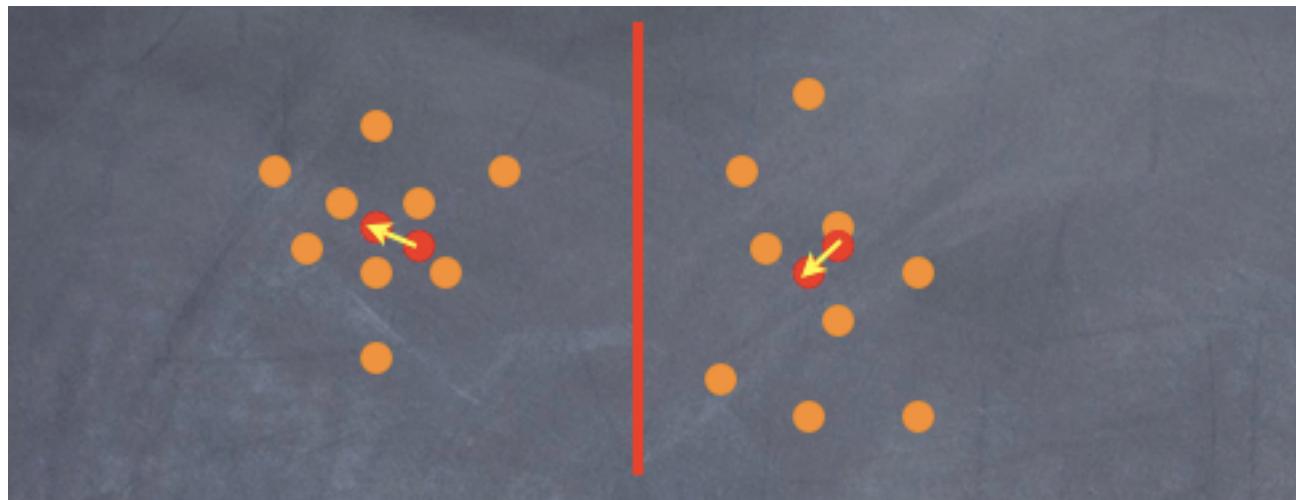
דוגמה



דוגמה



דוגמה



TIP

```
from PIL import Image
import math

class kMeans:
    def __init__(self, k, points):
        self.k = k
        self.points = points.copy()
    def lloyd_step(self, centers):
        # recalculate cluster centers
        clusters = self.cluster(centers)
        return list([self.center(clusters[i]) for i in range(self.k)])
    def cluster(self, centers):
        # cluster according to centers
        clusters = [[] for i in range(self.k)]
        for pt in range(len(self.points)):
            c = 0
            d = dist(self.points[pt], centers[0])
            for i in range(1, self.k):
                if dist(self.points[pt], centers[i]) < d:
                    c = i
                    d = dist(self.points[pt], centers[i])
            clusters[c].append(pt)
        return clusters
    def center(self, cluster):
        s = [sum([self.points[pt][rgb] for pt in cluster]) for rgb in range(3)]
        return tuple(int(s[rgb] / len(cluster)) for rgb in range(3))
    def lloyd_iter(self, initial_centers):
        centers = initial_centers.copy()
        while (True):
            new_centers = self.lloyd_step(centers)
            if sum([pt_neq(centers[i],new_centers[i]) for i in range(len(centers))]) == False:
                break
            yield new_centers
            centers = new_centers
```

```

from PIL import Image
import math

class kMeans:
    def __init__(self, k, points):
        self.k = k
        self.points = points.copy()
    def lloyd_step(self, centers):
        # recalculate cluster centers
        clusters = self.cluster(centers)
        return list([self.center(clusters[i]) for i in range(self.k)])
    def cluster(self, centers):
        # cluster according to centers
        clusters = [[] for i in range(self.k)]
        for pt in range(len(self.points)):
            c = 0
            d = dist(self.points[pt], centers[0])
            for i in range(1, self.k):
                if dist(self.points[pt], centers[i]) < d:
                    c = i
                    d = dist(self.points[pt], centers[i])
            clusters[c].append(pt)
        return clusters
    def center(self, cluster):
        s = [sum([self.points[pt][rgb] for pt in cluster]) for rgb in range(3)]
        return tuple(int(s[rgb] / len(cluster)) for rgb in range(3))
    def lloyd_iter(self, initial_centers):
        centers = initial_centers.copy()
        while (True):
            new_centers = self.lloyd_step(centers)
            if sum([pt_neq(centers[i],new_centers[i]) for i in range(len(centers))]) == False:
                break
            yield new_centers
            centers = new_centers

```

tip

מה יקרה אם נחליף שורה זו ב-

clusters = [[] * self.k]

```

clusters = [[ ] for i in range(self.k)]

```

פונקציות עזר

```
def dist(vec1, vec2):
    assert len(vec1) == len(vec2)
    d = 0
    for i in range(len(vec1)):
        d += abs(vec1[i] - vec2[i])
    return math.sqrt(d)

def pt_neq(vec1, vec2):
    assert len(vec1) == len(vec2)
    for i in range(len(vec1)):
        if (vec1[i] != vec2[i]):
            return True
    return False
```

סגמנטציה של תМОנות

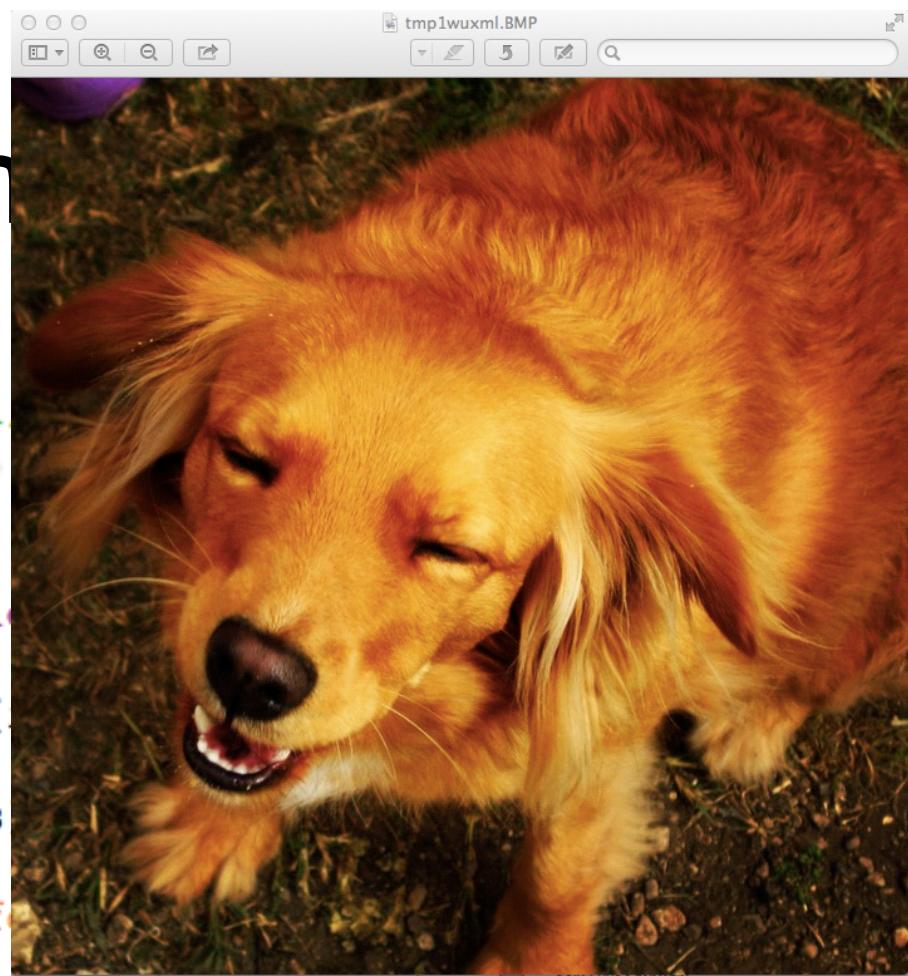
- בדרך כלל משתמשים באיטרציה Lloyd כדי לחלק אוספים של תМОנות לאשכולות של תМОנות דומות.
- אנחנו נדגים את האיטרציה עבור חלוקה של פיקסלים של תМОנה לאשכולות של פיקסלים דומים.
- זו דרך פרימיטיבית לבצע סגמנטציה - פירוק תМОנה למרכיבים.
- בדרך כלל סגמנטציה נעזרת גם בתנוני קרבה בתМОנה בין פיקסלים.

הרצאה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_0564.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [pix[0], pix[len(pix)-1]]
    print(initial_centers)
    for centers in clust.lloyd_iter(initial_centers):
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for i in range(len(pix))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```

לצת

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani")
    img.show() →
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [pix[0], pix[1]]
    print(initial_centers)
    for centers in clust.lloyd_iter():
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for _ in range(len(pix))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```



הרצאה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_0564.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [pix[0], pix[len(pix)-1]]
    print(initial_centers)
    for centers in clust.lloyd_iter(initial_centers):
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for i in range(len(pix))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```

הרצה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/○○○")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [pix[0], pix[1]]
    print(initial_centers)
    for centers in clust.lloyd_iter():
        print(centers) →
clusters = clust.cluster(centers)
for cluster in clusters:
    new_pix = [(255, 255, 255) for
    for pt in cluster:
        new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```

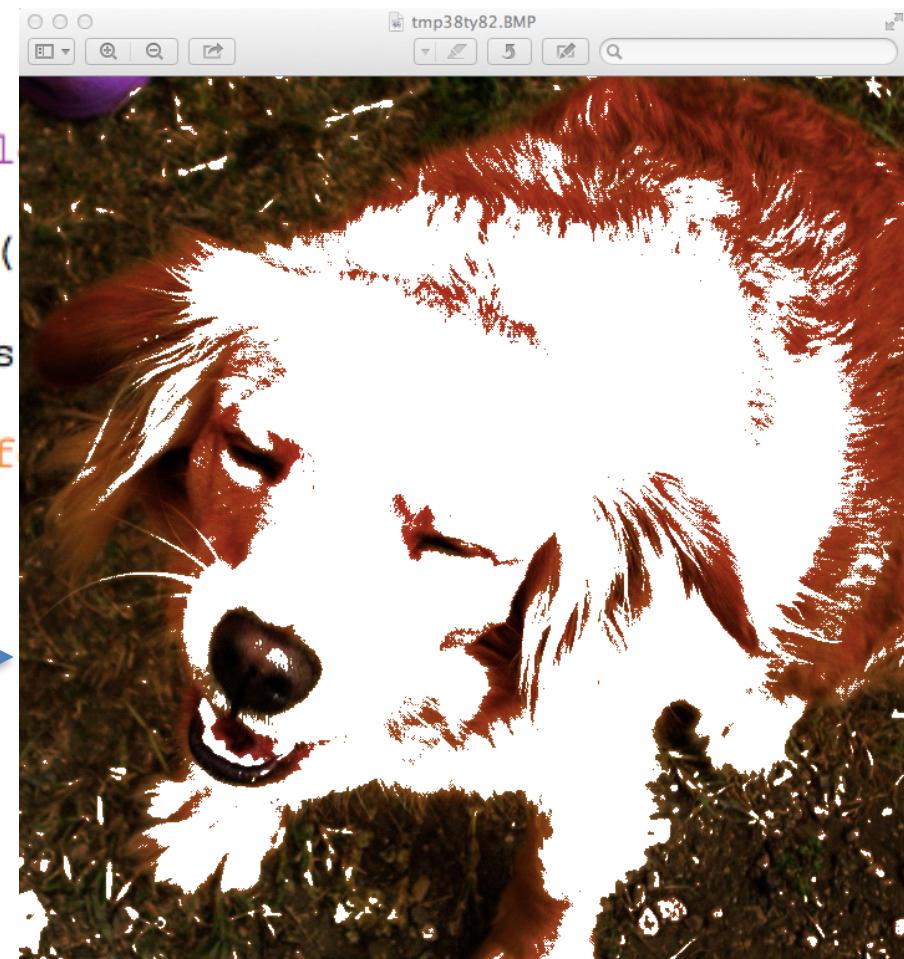
Python 3.3.3 (v3.3.3:c3896275c0f6, Nov
[GCC 4.2.1 (Apple Inc. build 5666) (dot
Type "copyright", "credits" or "license"
>>> ===== RE
>>>
[(23, 5, 17), (33, 16, 8)]
[(15, 6, 3), (136, 77, 25)]
[(46, 26, 9), (172, 97, 31)]
[(61, 32, 11), (189, 109, 35)]
[(70, 35, 12), (196, 116, 37)]
[(75, 36, 12), (200, 120, 39)]
[(78, 37, 12), (202, 122, 40)]
[(79, 38, 12), (203, 124, 40)]
[(80, 38, 12), (203, 125, 41)]
[(81, 38, 12), (204, 125, 41)]
>>>

הרצאה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_0564.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [pix[0], pix[len(pix)-1]]
    print(initial_centers)
    for centers in clust.lloyd_iter(initial_centers):
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for i in range(len(pix))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```

הרצה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_0564.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [pix[0], pix[1]]
    print(initial_centers)
    for centers in clust.lloyd_iter():
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for _ in range(len(cluster))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```

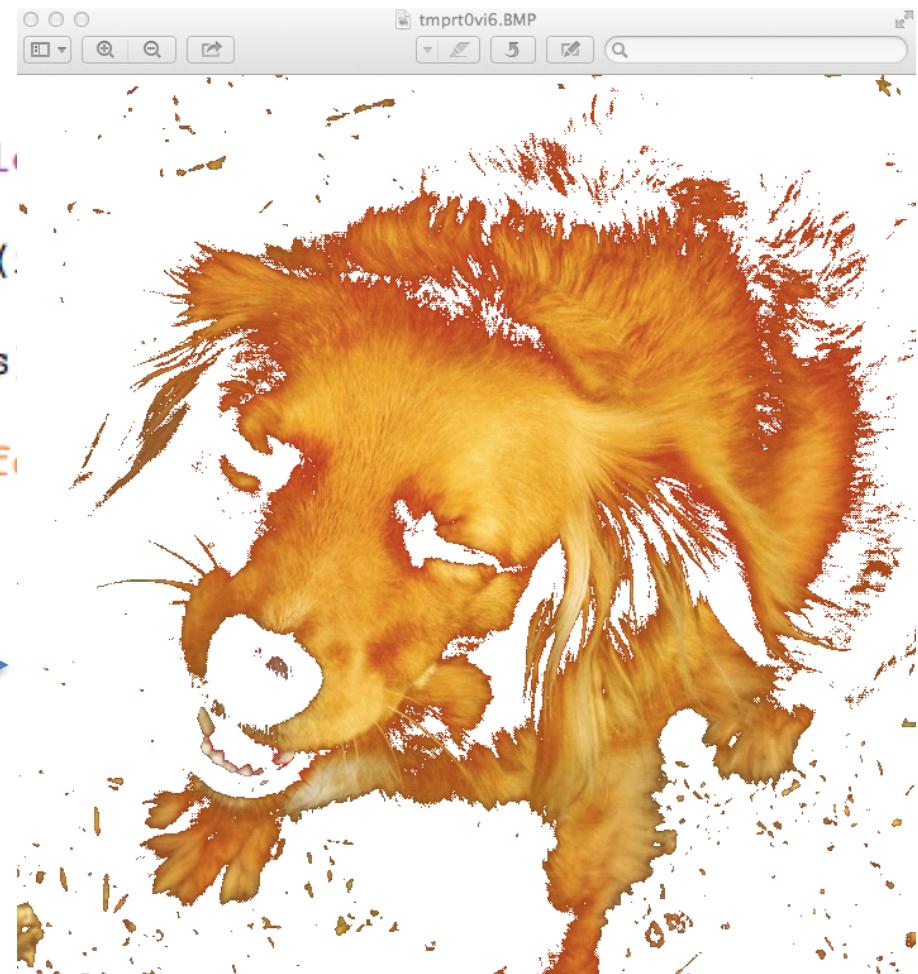


הרצאה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_0564.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [pix[0], pix[len(pix)-1]]
    print(initial_centers)
    for centers in clust.lloyd_iter(initial_centers):
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for i in range(len(pix))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```

הרצה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_0564.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [pix[0], pix[1]]
    print(initial_centers)
    for centers in clust.lloyd_iter():
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for i in range(len(cluster))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```



הרצאה

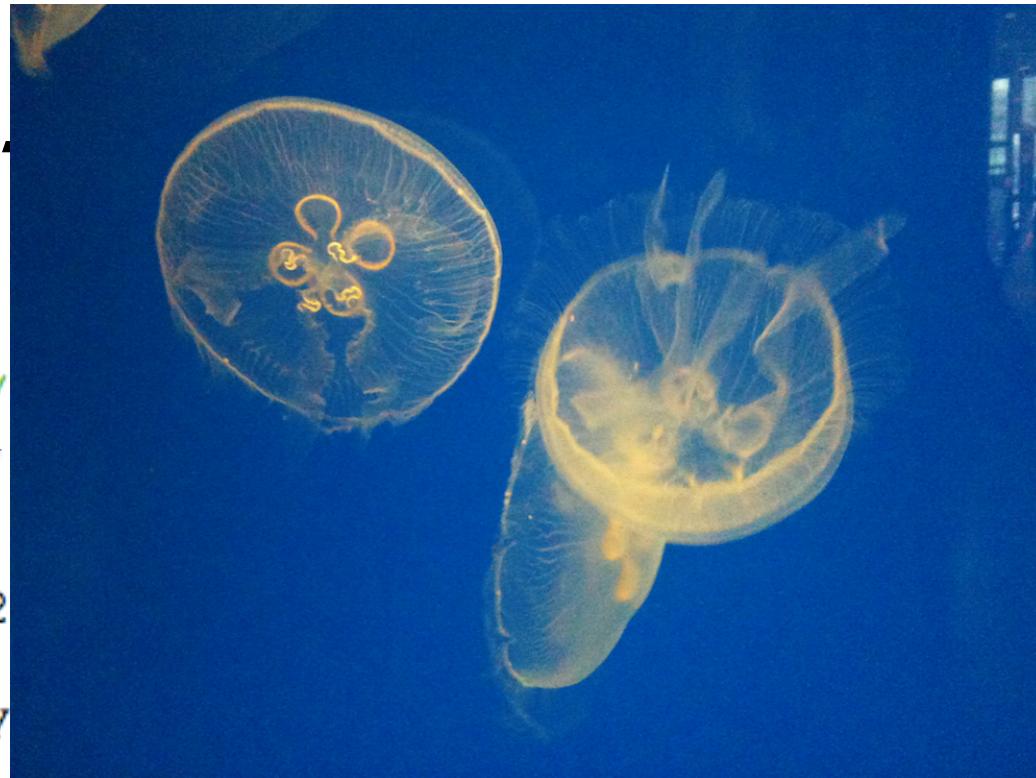
```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_0564.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [pix[0], pix[len(pix)-1]]
    print(initial_centers)
    for centers in clust.lloyd_iter(initial_centers):
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for i in range(len(pix))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```

עד הרצה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_1352.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [(0,0,255), (255, 255, 0)]
    print(initial_centers)
    for centers in clust.lloyd_iter(initial_centers):
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for i in range(len(pix))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```

אזה

```
if __name__ == "__main__":
    img = Image.open("/Users/
    img.show() →
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [(0,0,2
    print(initial_centers)
    for centers in clust.lloy
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for i in range(len(pix))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```



עד הרצה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_1352.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [(0,0,255), (255, 255, 0)]
    print(initial_centers)
    for centers in clust.lloyd_iter(initial_centers):
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for i in range(len(pix))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```

עד הרצה

```
if __name__ == "__main__":
    img = Image.open("/Users/○○○")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [(0,0,2)]
    print(initial_centers)
    for centers in clust.lloyd():
        print(centers) →
    clusters = clust.cluster()
    for cluster in clusters:
        new_pix = [(255, 255, 255)]
        for pt in cluster:
            new_pix[pt] = pix[pt]
        img.putdata(new_pix)
    img.show()
```

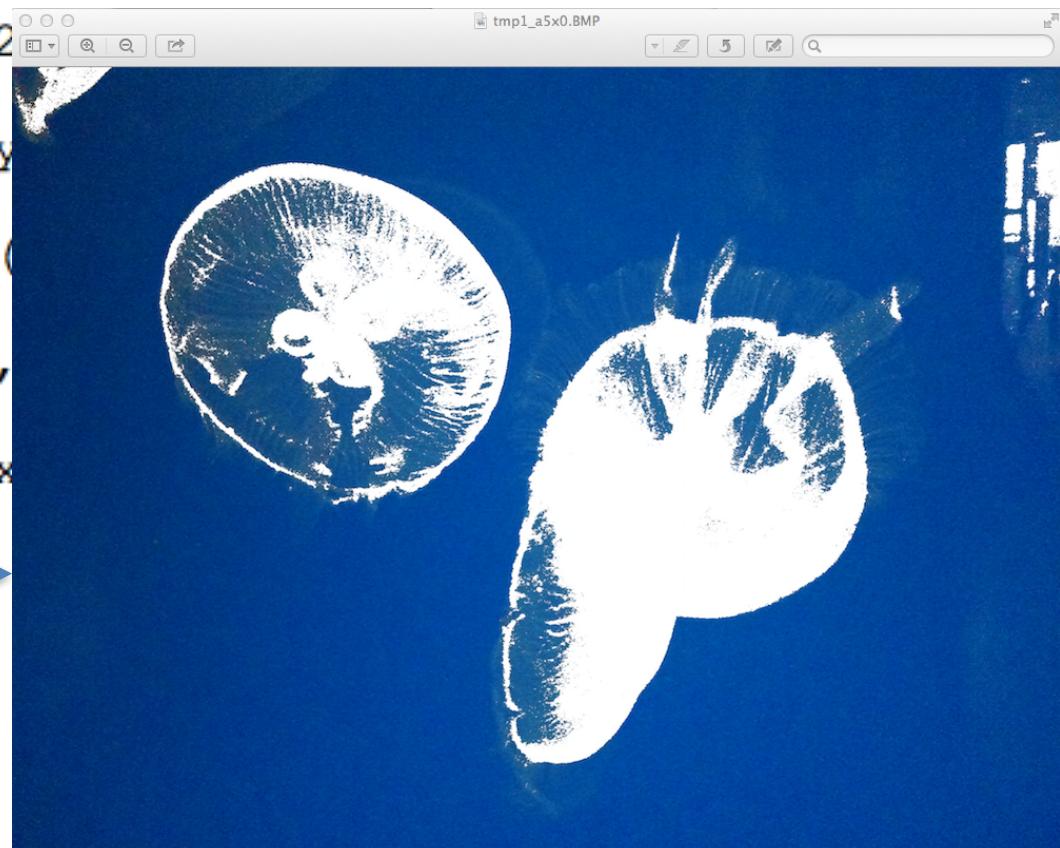
Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 16 2011, [GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] o
 Type "copyright", "credits" or "license()" for more information.

עד הרצה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_1352.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [(0,0,255), (255, 255, 0)]
    print(initial_centers)
    for centers in clust.lloyd_iter(initial_centers):
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for i in range(len(pix))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```

עד הרצה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_1352.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [(0,0,255), (255,255,255)]
    print(initial_centers)
    for centers in clust.lloyd():
        print(centers)
    clusters = clust.cluster()
    for cluster in clusters:
        new_pix = [(255, 255, 255)] * len(pix)
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```

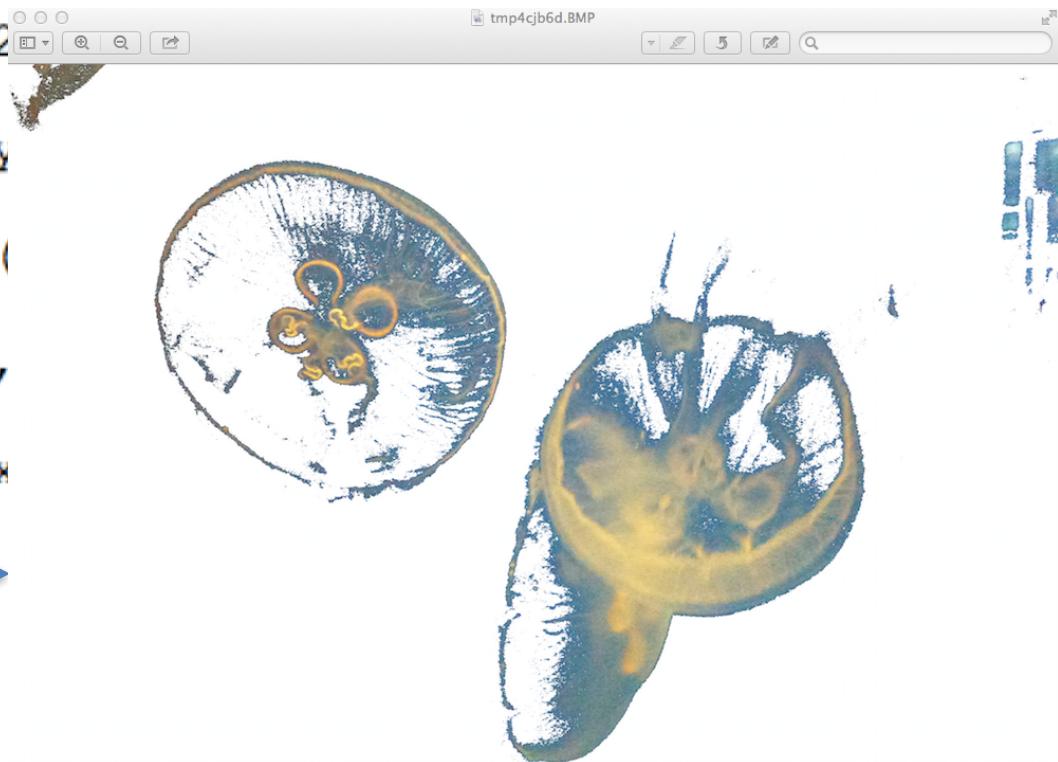


עד הרצה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_1352.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [(0,0,255), (255, 255, 0)]
    print(initial_centers)
    for centers in clust.lloyd_iter(initial_centers):
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for i in range(len(pix))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```

עד הרצה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_1352.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [(0,0,255), (255,255,255)]
    print(initial_centers)
    for centers in clust.lloyd
        print(centers)
    clusters = clust.cluster()
    for cluster in clusters:
        new_pix = [(255, 255, 255)]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```



עד הרצה

```
if __name__ == "__main__":
    img = Image.open("/Users/yrabani/IMG_1352.jpg")
    img.show()
    pix = list(img.getdata())
    clust = kMeans(2, pix)
    initial_centers = [(0,0,255), (255, 255, 0)]
    print(initial_centers)
    for centers in clust.lloyd_iter(initial_centers):
        print(centers)
    clusters = clust.cluster(centers)
    for cluster in clusters:
        new_pix = [(255, 255, 255) for i in range(len(pix))]
        for pt in cluster:
            new_pix[pt] = pix[pt]
    img.putdata(new_pix)
    img.show()
```