-Εργασία 1-

1.1 Rotation

Για την υλοποίηση της περιστροφής μιας εικόνας εισόδου A δημιουργείται η συνάρτηση

```
function G = myImgRotation(A, ang)
```

όπου για κάθε pixel (u,v) της εικόνας εξόδου G πραγματοποιείται ο αντίστροφος μετασχηματισμός περιστροφής για να βρεθεί τον αντίστοιχο pixel (x,y) της εικόνας εισόδου.

Ο ευθύς μετασχηματισμός των pixels $(x, y) \rightarrow (u, v)$ της εικόνας δίνεται από την σχέση

$$\begin{bmatrix} u \\ v \end{bmatrix} = floor(\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}) + \begin{bmatrix} t_u \\ t_v \end{bmatrix}$$

όπου ο αντιστρέψιμος πίνακας $T = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$ είναι ο πίνακας στροφής και τα t_u, t_v είναι κάποια translations που χρησιμοποιούνται για τους σκοπούς της τεχνικής υλοποίησης (για να αποφευχθούν αρνητικοί δείκτες σε πίνακες). Τα τελευταία υπολογίζονται μέσω των μετασχηματισμένων συντεταγμένων των 4 άκρων της εικόνας (των σημείων $[1,1], [1, N2], [N1,1], [N1, N2]$ δηλαδή, όπου N1 και N2 οι διαστάσεις τις εικόνας). Ακολουθώντας την αντίστροφη πορεία μπορεί να βρεθεί και ο αντίστροφος μετασχηματισμός των συντεταγμένων.

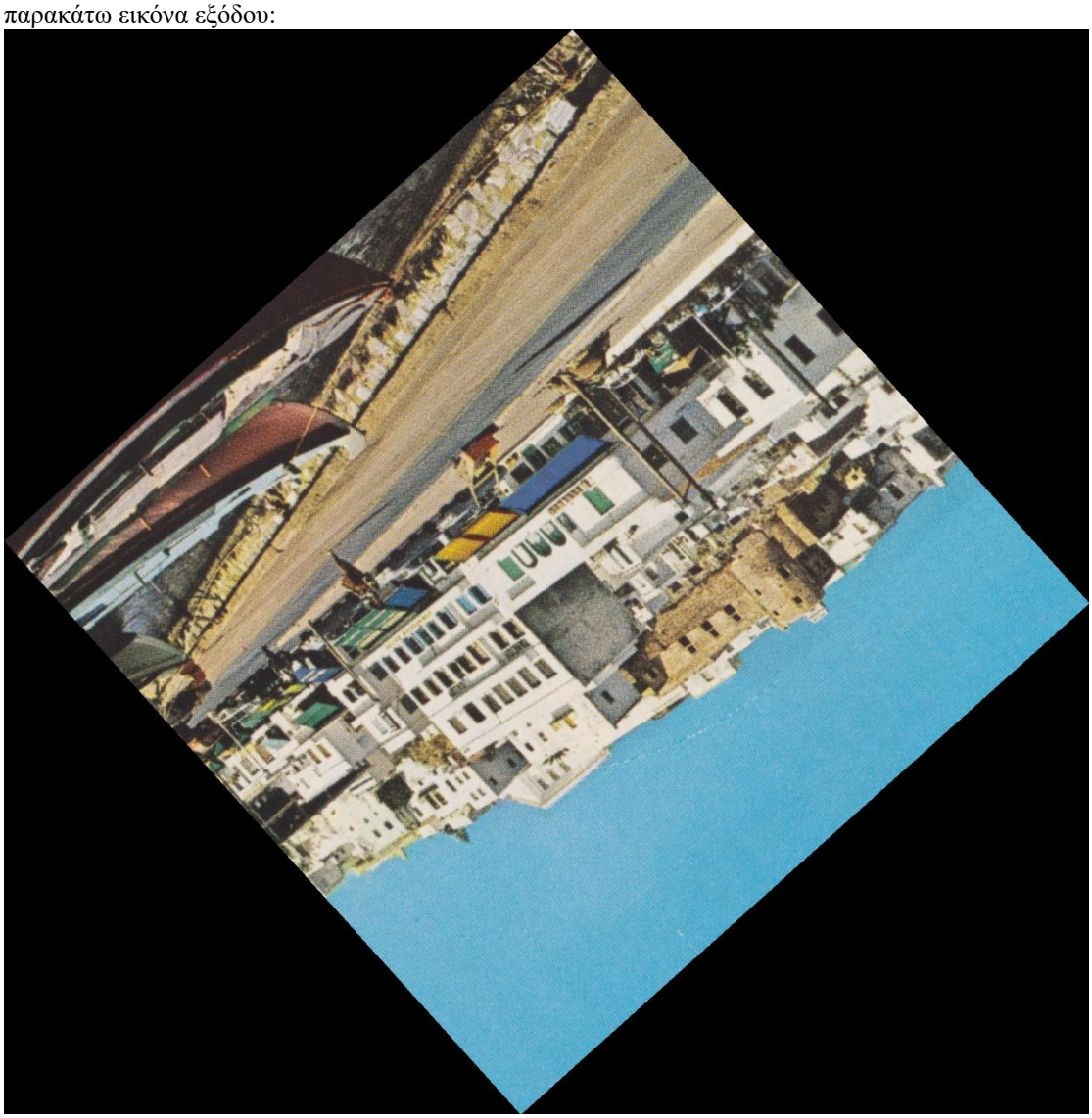
Η τιμή του G(u,v) είναι ο μέσος όρος των τιμών της αρχικής εικόνας για τους 4 πλησιέστερους γείτονες του (x,y) (ή όσους υπάρχουν/αντιστοιχούν από αυτούς).
Επιβάλλεται δηλαδή ενός είδους smoothing στην εικόνα εξόδου.

1.

Χρησιμοποιώντας ως εικόνα εισόδου την TestIm1.png και για στροφή κατά $\theta_1 = 35°$ λαμβάνουμε την παρακάτω εικόνα εξόδου:

2.

Ομοίως χρησιμοποιώντας την ίδια εικόνα εισόδου και για για στροφή κατά $\theta_2 = 222°$ λαμβάνουμε την παρακάτω εικόνα εξόδου:



1.2 Local Descriptors

Για την υλοποίηση τοπικών περιγραφέων σημείων ενδιαφέροντος σε κάποια εικόνα δημιουργούνται οι συναρτήσεις

```
function d = myLocalDescriptor(I,p,rhom,rhoM,rhostep,N)
function d = myLocalDescriptorUpgrade(I,p,rhom,rhoM,rhostep,N)
```

Οι συναρτήσεις αυτές υπολογίζουν τιμές για τα pixels με συντεταγμένες που βρίσκονται πάνω σε ομόκεντρους κύκλους με κέντρο το σημείο p, ακτίνες από rhom έως rhoM με βήμα rhostep, και για N διαδοχικές ίσες γωνίες (ίσες με $\frac{2\pi}{N}$rad η κάθε μία).

Θεωρώντας ένα πολικό σύστημα συντεταγμένων με κέντρο το σημείο p, για κάθε σημείο (r,θ) όπως τα παραπάνω, υπολογίζεται για την πρώτη συνάρτηση η τιμή της εικόνας στο σημείο αυτό.
Για την δεύτερη συνάρτηση, υπολογίζεται η Λαπλασιανή στο σημείο (r,θ), για τον οποίο τελεστή γνωρίζουμε ότι είναι αναλλοίωτος σε περιστροφές.

Τα παραπάνω ισχύουν σε περίπτωση που κάποιο σημείο υπολογισμού (r,θ) ταυτίζεται με τις συντεταγμένες κάποιου pixel (x,y – σε καρτεσιανές συντεταγμένες) της εικόνας. Σε αντίθετη περίπτωση γίνεται παρεμβολή με τις τιμές των γειτονικών pixel.

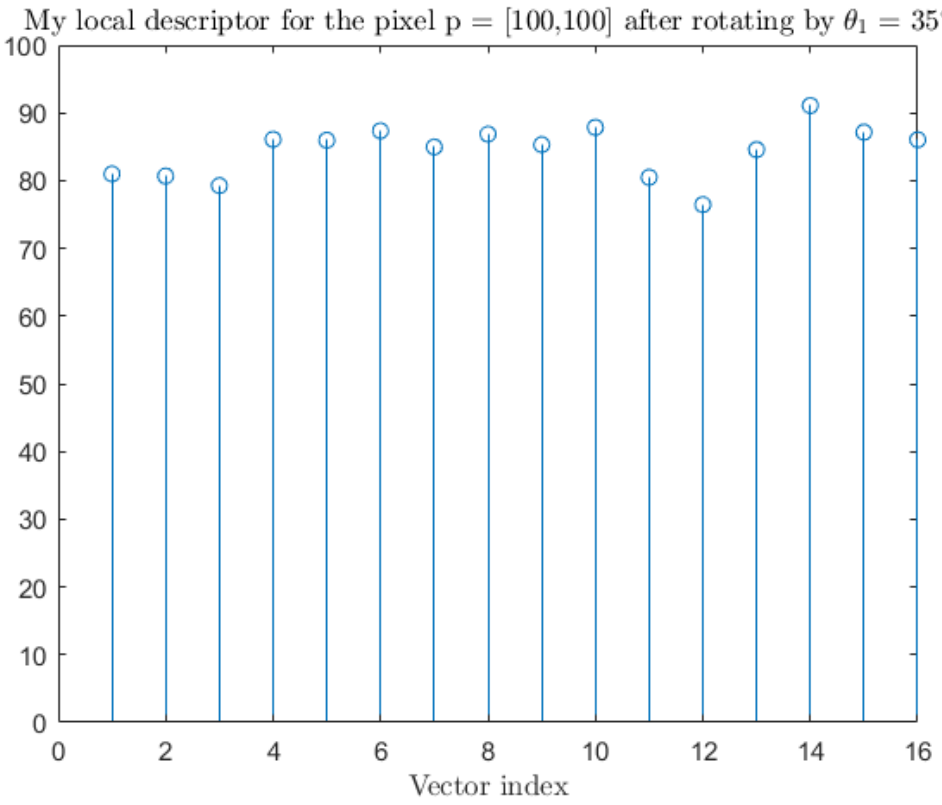
Το διάνυσμα που επιστρέφεται ως περιγραφέας έχει ως στοιχεία του τους μέσους όρους (ως προς τις γωνίες) των υπολογισμένων τιμών για κάθε ακτίνα και είναι ίδιου μήκους για τις δύο συναρτήσεις.

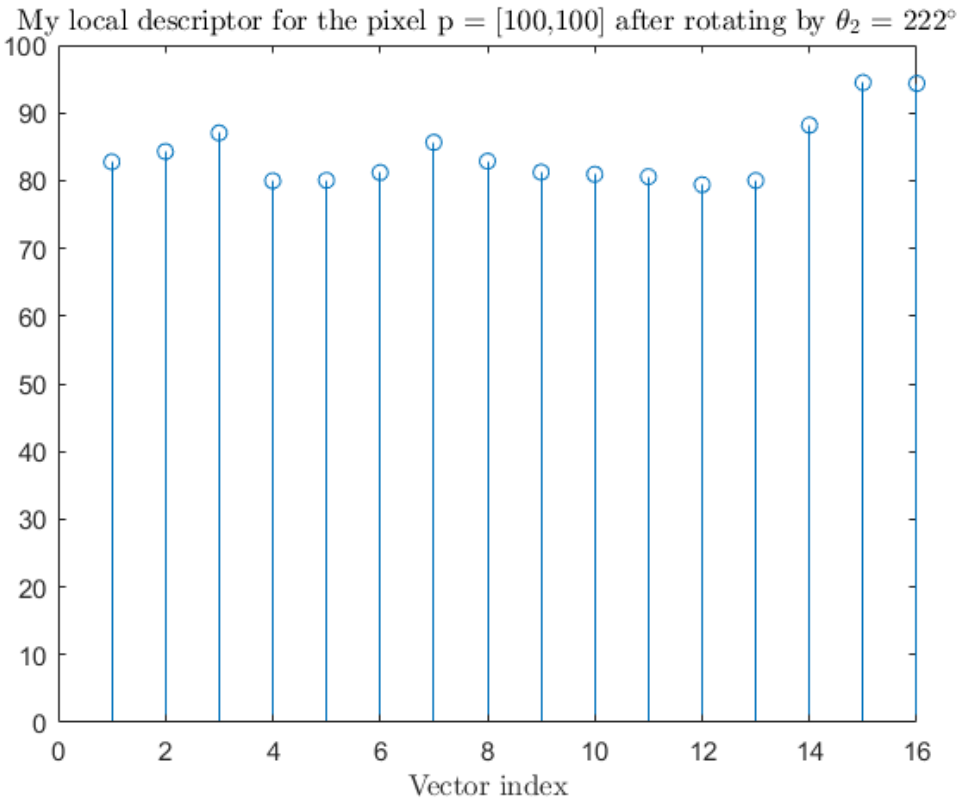
1.

Για την εύρεση του αντίστοιχου pixel μετά την περιστροφή της εικόνας δημιουργείται η συνάρτηση

```
function [u1,u2]  = fwd(P,theta,N1,N2)
```

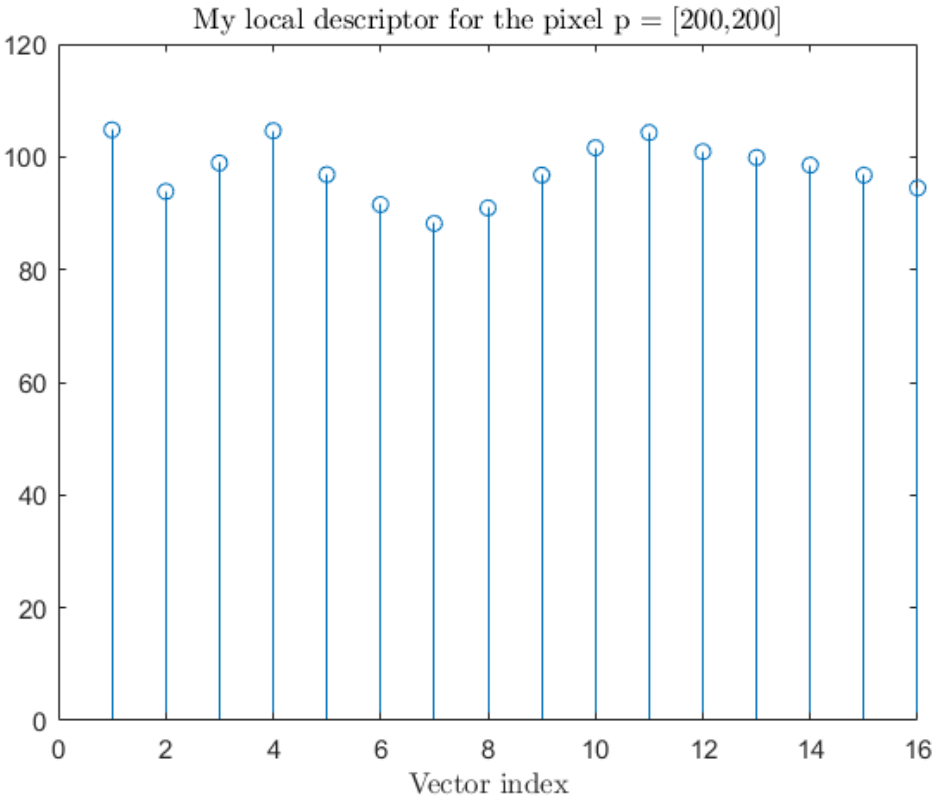
η οποία εφαρμόζει τον ευθύ μετασχηματισμό.
Έτσι για τα αντίστοιχα του pixel p = [100,100] στις περιστραμμένες εικόνες κατά $\theta_1 = 35°, \theta_2 = 222°$ και χρησιμοποιώντας την βασική έκδοση του περιγραφέα με παραμέτρους rhom = 5, rhoM = 20, rhostep = 1, N = 8 παίρνουμε τις παρακάτω τιμές για τα διανύσματα:

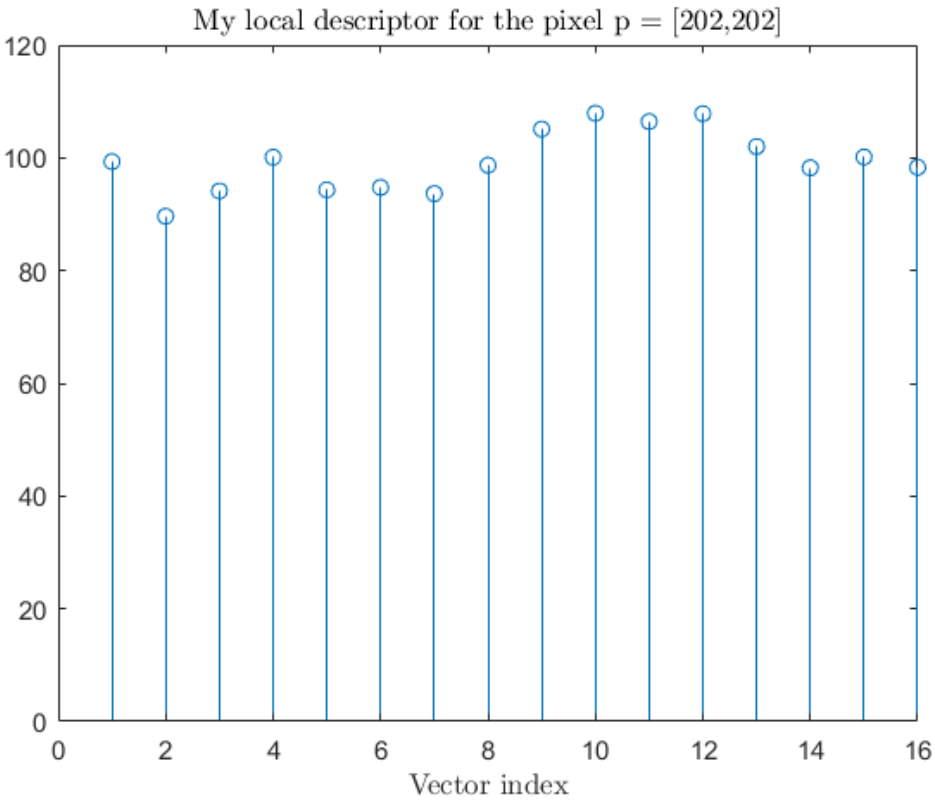




2.

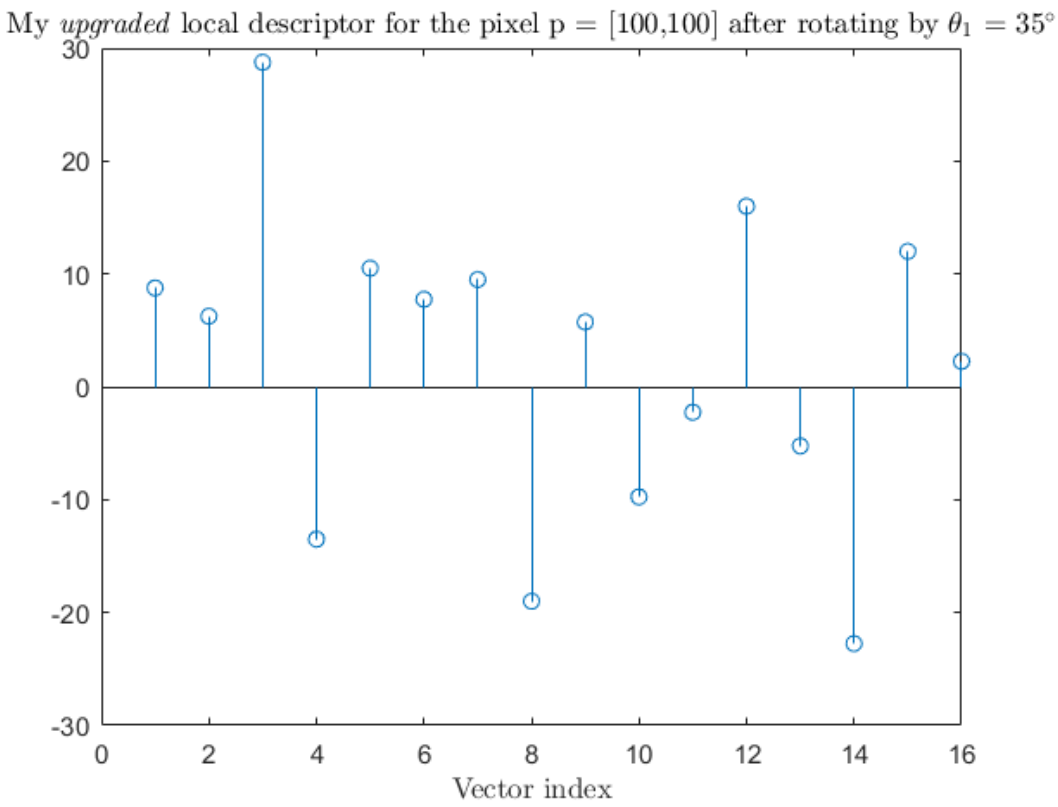
Ομοίως για τα pixel q1 = [200,200], q2 = [202,202] για την αρχική εικόνα (χωρίς περιστροφή) παίρνουμε τις παρακάτω τιμές για τα διανύσματα:

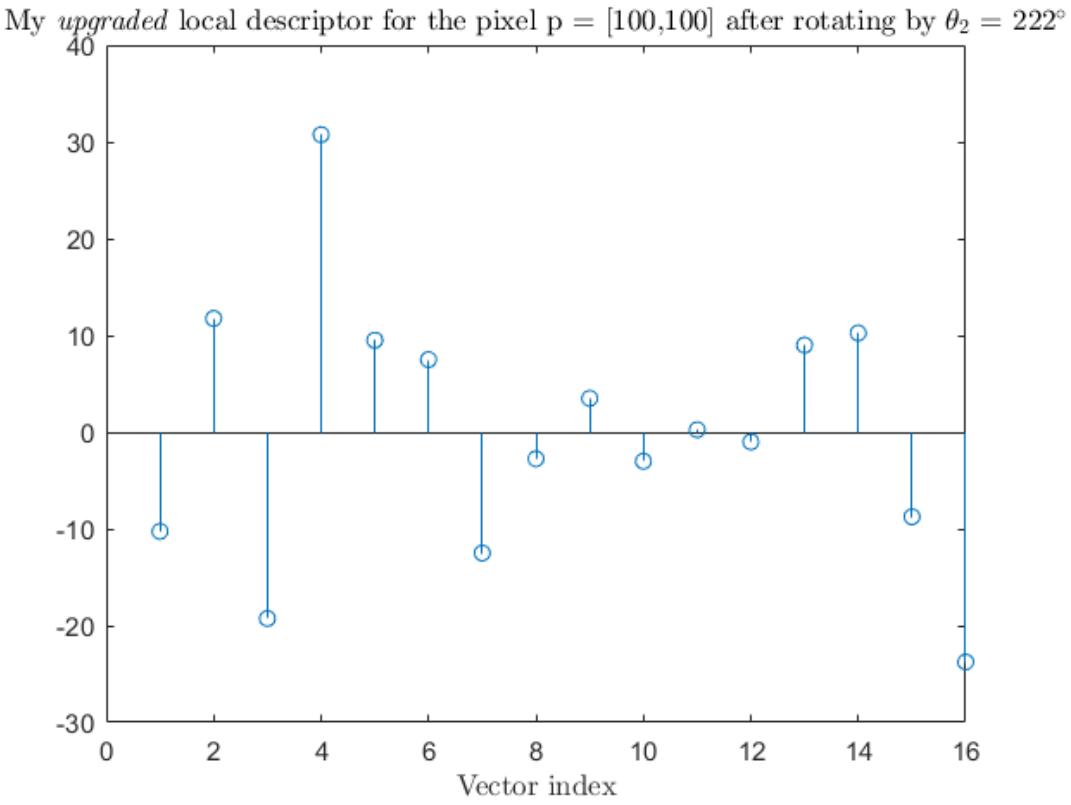




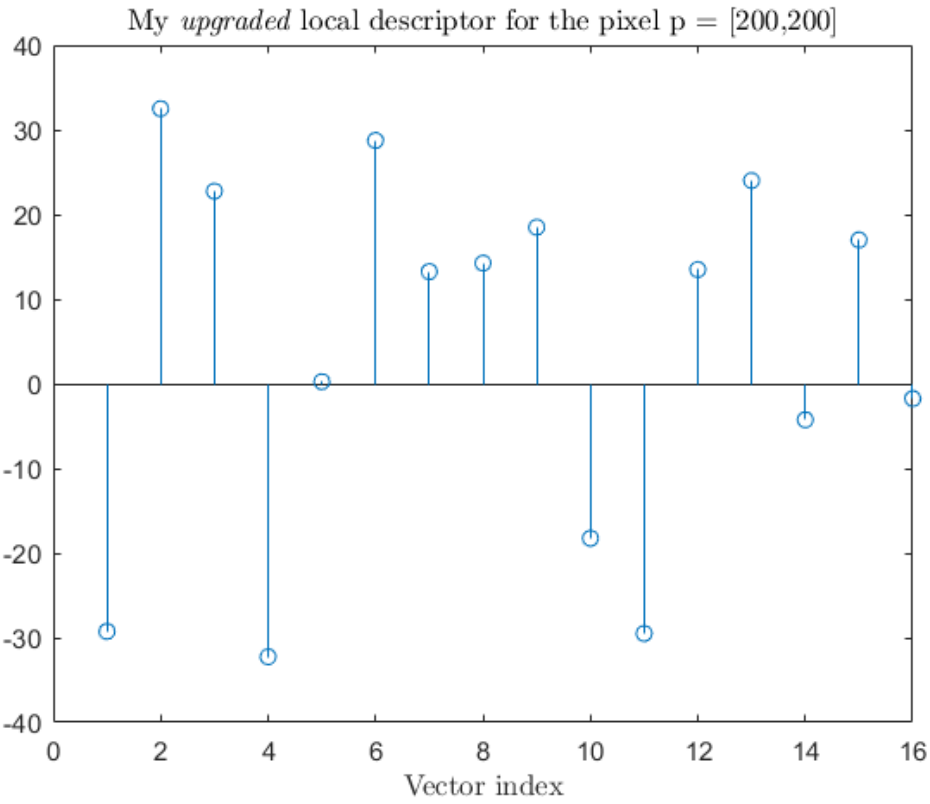
3.

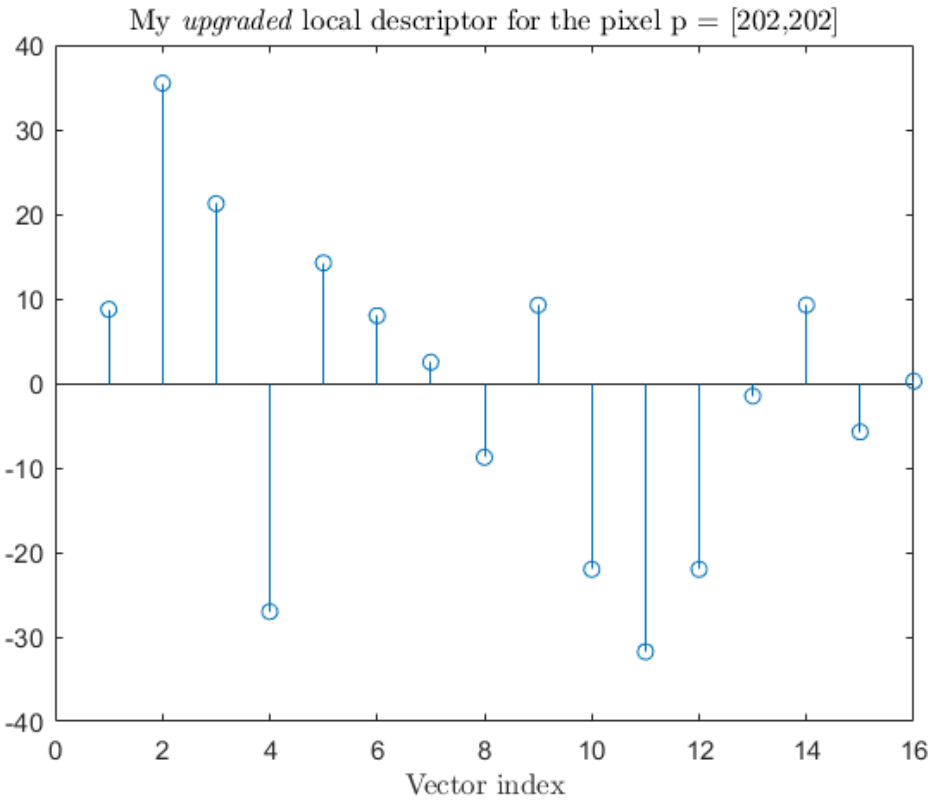
Επαναλαμβάνοντας τα παραπάνω, χρησιμοποιώντας την εναλλακτική υλοποίηση του περιγραφέα, παίρνουμε για τα μετασχηματισμένο pixel p = [100,100] κατά $\theta_1 = 35°, \theta_2 = 222°$ με τις ίδιες επιλογές παραμέτρων:





Και για τα pixel q1 = [200,200], q2 = [202,202] για την αρχική εικόνα:





Παρατηρούμε ότι παίρνουμε παρόμοιες τιμές του διανύσματος των περιγραφέων τόσο για τα αντίστοιχα pixels μετά από στροφή (pixel p), αλλά και για κοντινά μεταξύ τους pixels που βρίσκονται σε σχετικά ομαλές περιοχές της εικόνας (pixels q1,q2).

1.3 Harris Corner Detection

Για την αναγνώριση ιδιαζόντων σημείων μιας εικόνας εισόδου (με σχετικά γρήγορη υλοποίηση) δημιουργείται η συνάρτηση

```
function corners = myDetectHarrisFeatures(I,k = 0.05,sigma = 5/3,
Rthres = 100, windowsize = 5)
```

Κατασκευάζεται αρχικά το Gaussian window function/matrix $W(x_1, x_2;\text{sigma}) = \exp\{-\frac{(x_1^2+x_2^2)}{2sigma^2}\}$ για τα $[x_1, x_2]$ σε ένα τετραγωνικό παράθυρο κέντρου (0,0) και πλευράς windowsize. Υπολογίζονται οι μερικές παράγωγοι $I_1 = \frac{\partial I}{\partial v_1}, I_2 = \frac{\partial I}{\partial v_2}$ της εικόνας I (όπου ορίζονται) κατά την κατακόρυφη και οριζόντια διεύθυνση σύμφωνα με τον τελεστή Sobel.

Για κάθε στοιχείο της εικόνας για το οποίο ορίζονται οι μερικές παράγωγοι των γειτόνων του σε ένα τετραγωνικό παράθυρο με κέντρο το ίδιο και πλευράς windowsize εξετάζεται αν αποτελεί γωνία, μέσω της συνάρτησης

```
function c = isCorner(k,Rthres,W,I1,I2)
```

όπου λαμβάνονται ως είσοδοι το παράθυρο W και οι μερικές παράγωγοι εντός του παραθύρου. Υπολογίζοντας τις ιδιοτιμές λ_1, λ_2 του πίνακα

$$M(p_1,p_2) = \sum_{u_1}\sum_{u_2} w(u_1,u_2)\begin{bmatrix} I_1(p_1+u_1,p_2+u_2)^2 & I_1(p_1+u_1,p_2+u_2)I_2(p_1+u_1,p_2+u_2) \\ I_1(p_1+u_1,p_2+u_2)I_2(p_1+u_1,p_2+u_2) & I_2(p_1+u_1,p_2+u_2)^2 \end{bmatrix},$$

και με βάση την παράμετρο k της συνάρτησης υπολογίζεται για το σημείο p η μετρική $R(p_1,p_2) = \lambda_1\lambda_2 - k(\lambda_1+\lambda_2)^2$, η οποία όταν είναι μεγαλύτερη από την παράμετρο Rthres, υποδεικνύεται ότι εμφανίζεται γωνία στο εκάστοτε εξεταζόμενο σημείο (επιστρέφοντας c=1 και c=0 σε άλλη περίπτωση).

1.

Έτσι χρησιμοποιώντας τις παραπάνω συναρτήσεις μέσω της συνάρτησης

```
function corners = part_1_3(I,k,sigma,Rthres,windowsize)
```

για default τιμές παραμέτρων, σχηματίζοντας ένα κόκκινο 5x5 τετράγωνο για κάθε γωνιακό σημείο και για εικόνες εισόδου τις grayscale (channel #2) των TestIm1.png και TestIm2.png παίρνουμε αντίστοιχα:

My harris corner detector

My harris corner detector

Σε γενικές γραμμές εντοπίζονται σημεία που αντιστοιχούν όντως σε ακμές και αρκετά από αυτά είναι κοινά για τις δύο εικόνες.

2. Το πρόβλημα

Για την ένωση των δύο εικόνων σε μία δημιουργούνται οι συναρτήσεις

```
function imgnew = myStitch(I1c,I2c,corners1,corners2,npoints)
function T = myTransformEstimation
(I1,I2,corners1,corners2,npoints = 20)
```

Η τελευταία λαμβάνει ως είσοδο τις δύο grayscale εικόνες (I1, I2) και τα ιδιάζοντα σημεία που προσδιορίστηκαν προηγουμένως (corners1, corners2).
Χρησιμοποιεί την βασική έκδοση του περιγραφέα για κάθε ιδιάζον σημείο των εικόνων για να εξάγει διανύσματα χαρακτηριστικών.
Για κάθε διάνυσμα χαρακτηριστικών ιδιάζοντος σημείου της 2ης εικόνας υπολογίζεται το αντίστοιχο της 1ης εικόνας που απέχει την μικρότερη ευκλείδεια απόσταση.
Γίνεται αύξουσα ταξινόμηση των ζευγών που προκύπτουν με βάση την απόσταση και επιλέγονται τα πρώτα npoints (≥ 2) ζεύγη σημείων για να αποτελέσουν τα δεδομένα δύο συνόλων (A για την 1η εικόνα και B για την 2η).

Σχηματίζεται ο πίνακας διασπορών των δύο συνόλων $H = (B - E[B])^T(A - E[A])$ και γίνεται ανάλυση των ιδιάζουσων τιμών του πίνακα H.
Από τους πίνακες των δεξιών και αριστερών ιδιάζοντων διανυσμάτων (U και V αντίστοιχα) προκύπτει η εκτίμηση για τον πίνακα στροφής ως $R = VU^T$.
Εφόσον έχει εκτιμηθεί ο πίνακας στροφής, η εκτίμηση για το διάνυσμα των translations προκύπτει ως $t = E[A] - R \cdot E[B]$.
[Kabsch algorithm]
[Using SVD for some fitting]
[Least-squares fitting of two 3-D point sets]

Ως έξοδος προκύπτει η εκτίμηση του πίνακα μετασχηματισμού $T = \begin{bmatrix} R & \vec{0} \\ t^T & 1 \end{bmatrix}$.

Η συνάρτηση myStitch αφού καλέσει την δεύτερη, εξάγει την εκτίμηση ($\hat{\theta}$) της γωνίας θ με βάση τον πίνακα μετασχηματισμού και περιστρέφει κατά αυτή τη γωνία την έγχρωμη της δεύτερης εικόνας (χρησιμοποιώντας την συνάρτηση myImgRotation).
Υπολογίζει με βάση το διάνυσμα του translation και την γωνία θ το διάνυσμα μετατόπισης $\Delta u = [\Delta u_1, \Delta u_2]$ μεταξύ των pixels [1,1] της εικόνας 1 και της περιστραμμένης εικόνας 2.
Με βάση αυτό καλείται η συνάρτηση

```
imgnew = fillImg(I1c,II2c,deltau1,deltau2)
```

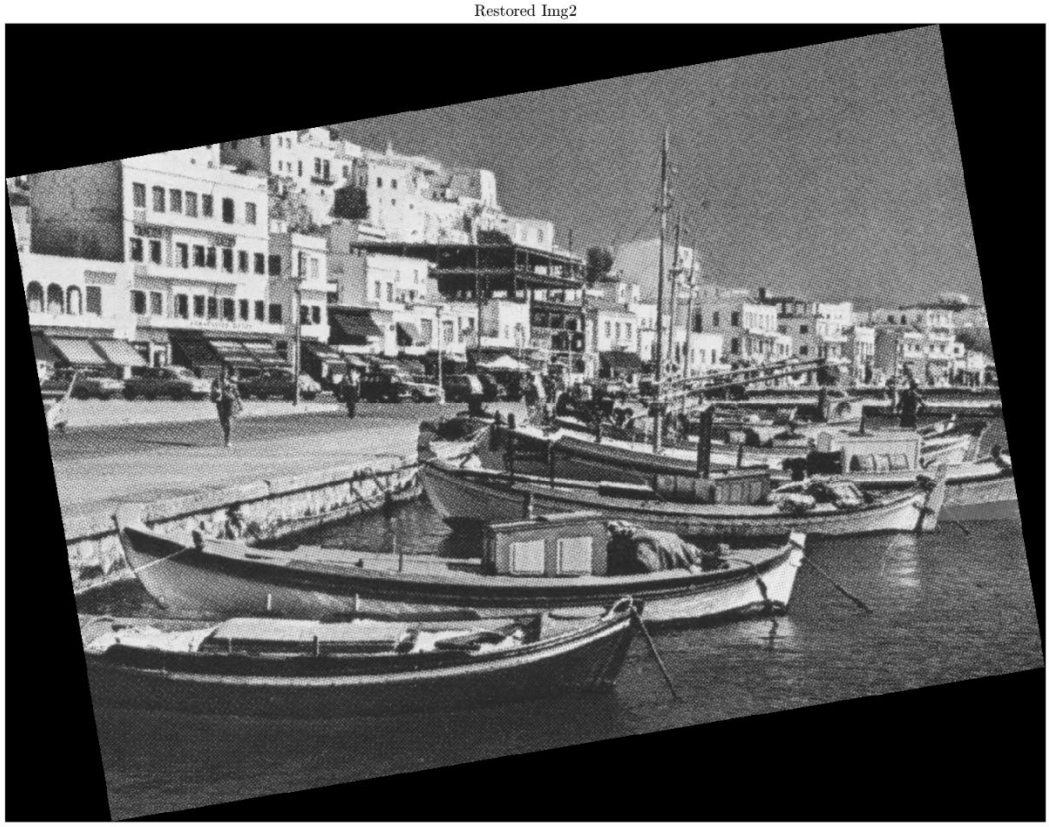
η οποία συμπληρώνει την τελική εικόνα με τα περιεχόμενα και των δύο εικόνων, ενώνοντας τες.

Η τελευταία συνάρτηση με βάση το διάνυσμα Δu λαμβάνει υπόψιν τις σχετικές θέσεις στις οποίες πρέπει να μετακινηθούν οι εικόνες 1 και 2 για κάθε περίπτωση (όχι μόνο την συγκεκριμένη για τις TestIm1 και TestIm2) εφόσον αυτές έχουν κοινά σημεία.

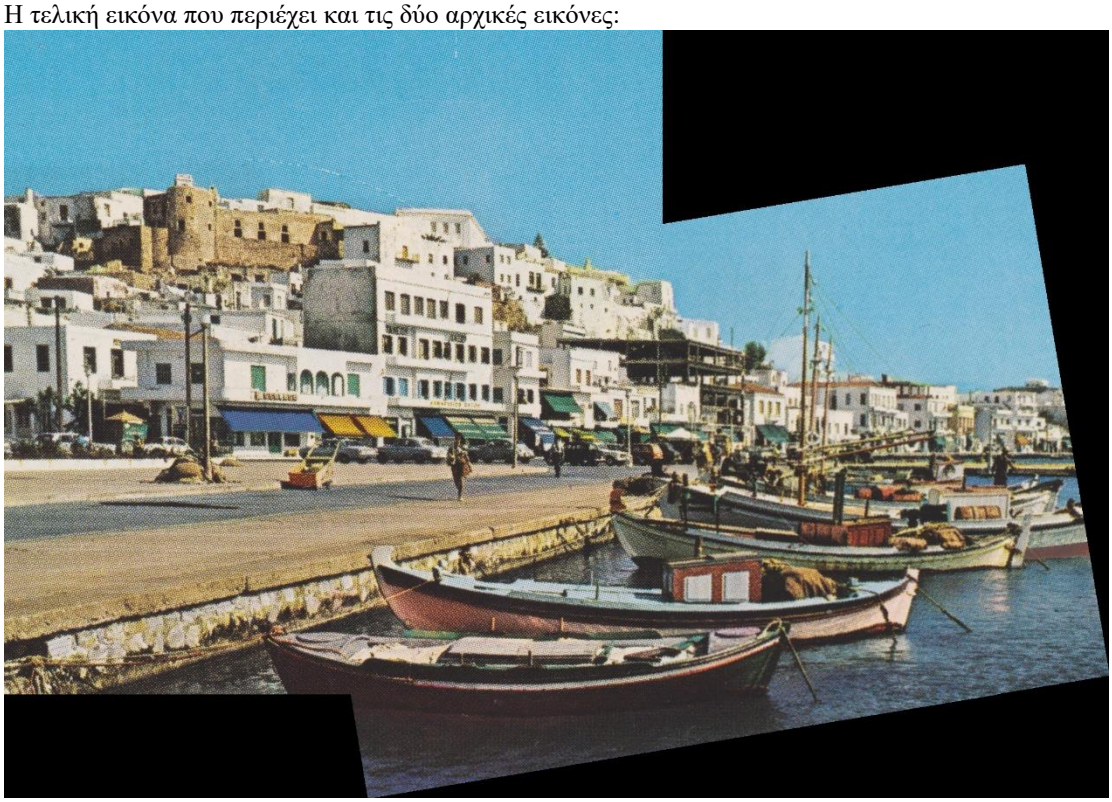
Τα αποτελέσματα του αλγόριθμου παρουσιάζονται παρακάτω.

H restored – reoriented TestIm2.png:

Restored Img2



Η τελική εικόνα που περιέχει και τις δύο αρχικές εικόνες:



Extra testing για διαφορετικούς τρόπους με τους οποίους πρέπει να ενωθούν οι εικόνες

Original sample image:

I1:

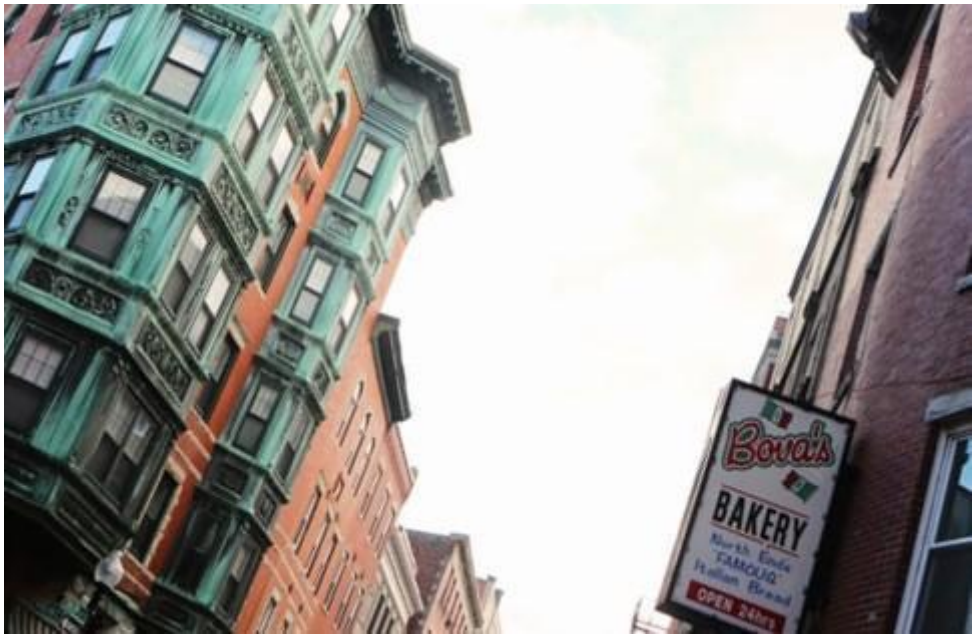
I2:

Sitched:

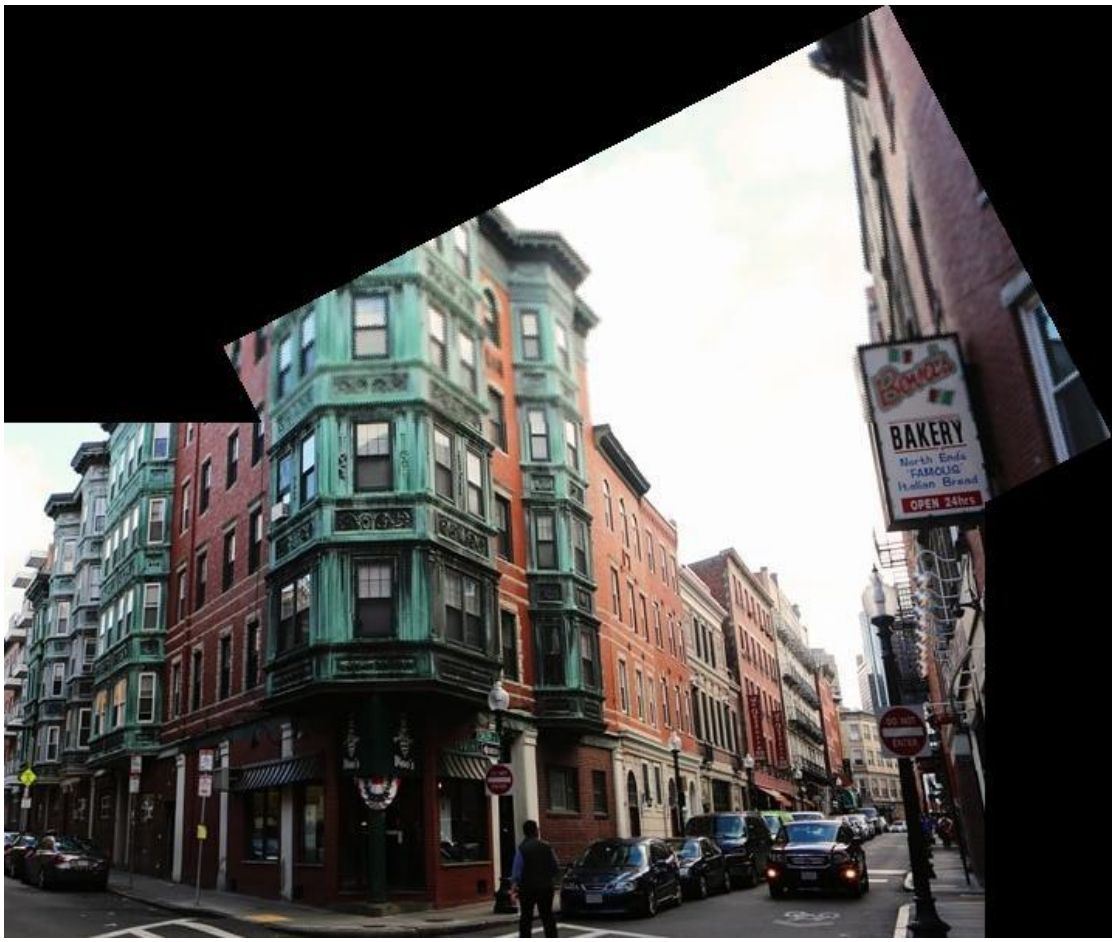
I1:


BAKERY
North Ends
"FAMOUS"
Italian Bread
OPEN 24hrs

I2:


Bova's
BAKERY
North Ends
"FAMOUS"
Italian Bread
OPEN 24hrs

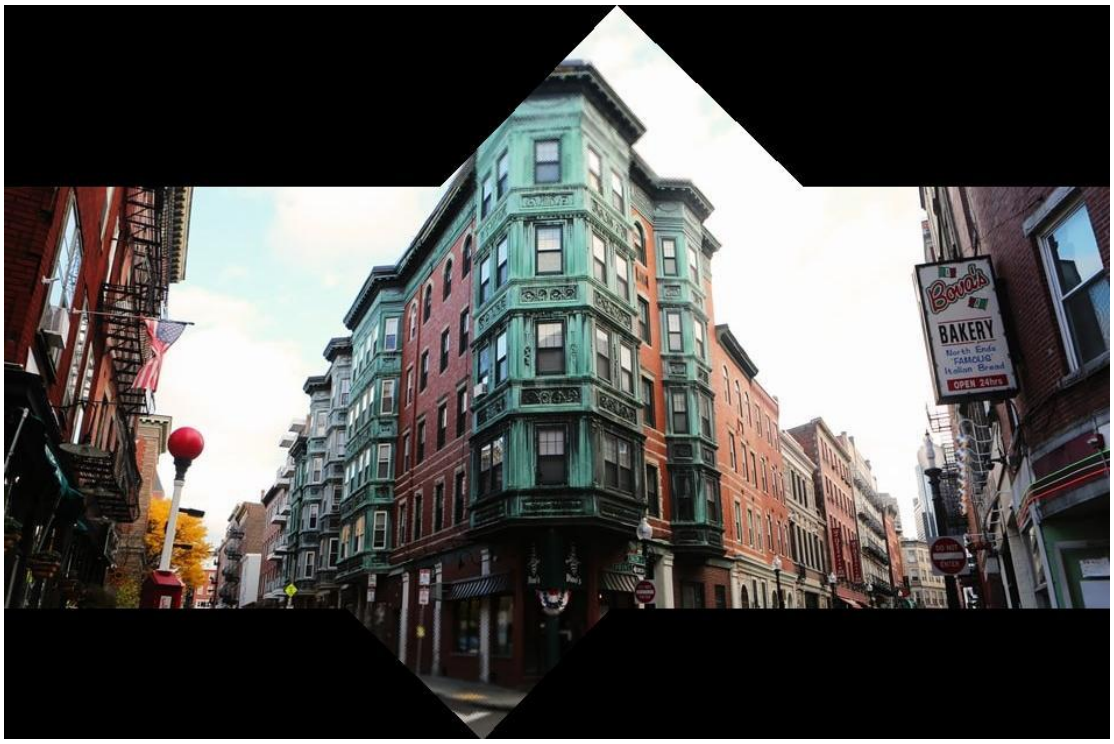
Stitched:



I1:

I2:

Stitched:


Bonci's
BAKERY
North Ends
TARALLINI
Italian Bread
OPEN 24hrs