Πολυτεχνική Σχολή
Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Θέματα Όρασης Υπολογιστών

ΕΡΓΑΣΤΗΡΙΑΚΉ ΆΣΚΗΣΗ 3

Κατσαρός Ανδρέας
Α.Μ. 1084522

Πάτρα, 2024-25

ΕΡΩΤΗΜΑΤΑ

Γενική *Επισκόπηση Αλγόριθμου SIFT*

Ο αλγόριθμος **SIFT**, που προτάθηκε από τον David Lowe το 2004, αποτελεί μια από τις πιο ευρέως χρησιμοποιούμενες μεθόδους για την ανίχνευση και περιγραφή τοπικών χαρακτηριστικών σε ψηφιακές εικόνες.
Η βασική ιδέα του SIFT είναι ότι *μετατρέπει ολόκληρη την εικόνα* σε ένα σύνολο σημείων-κλειδιών (**keypoints**), τοπικών δηλαδή «διακριτών» σημείων με έντονη πληροφορία.

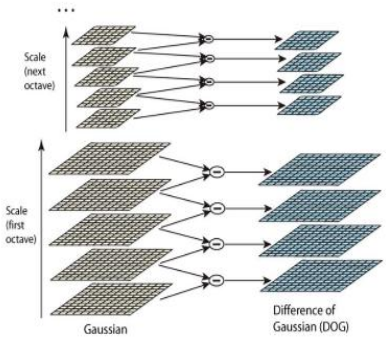
Στη συνέχεια, γύρω από κάθε keypoint εξάγεται ένας πολυδιάστατος περιγραφέας (**descriptor**), ο οποίος περιγράφει μαθηματικά τη γειτονιά του σημείου με όρους κλίσεων (gradients).

Έτσι, αν εντοπιστούν τα ίδια keypoints σε δύο διαφορετικές εικόνες, η σύγκριση των αντίστοιχων descriptors μάς επιτρέπει να αποφασίσουμε εάν οι δύο εικόνες απεικονίζουν την ίδια σκηνή ή/και τα ίδια αντικείμενα (ακόμα και αν υπάρχουν μεταβολές κλίμακας, γωνίας λήψης ή φωτισμού).

---

1. Ανίχνευση και Εντοπισμός Keypoints

Το πρώτο βήμα του SIFT περιλαμβάνει τη δημιουργία ενός **χώρου κλίμακας** (*Scale-Space*). Η εικόνα υφίσταται διαδοχική θόλωση (Gaussian blur) με διαφορετικές τιμές σ(διάφορα επίπεδα), ενώ παράλληλα πραγματοποιείται υποδειγματοληψία (downsampling) για να καλυφθεί ευρύ φάσμα κλιμάκων.



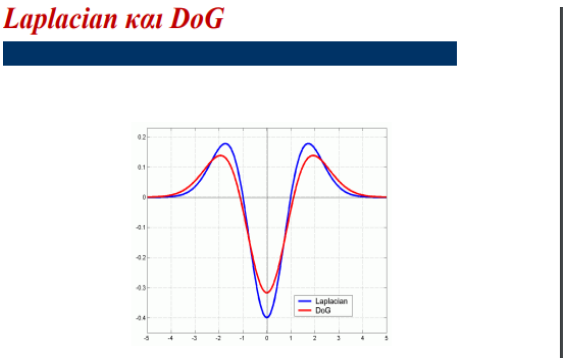
- Η θόλωση επιτυγχάνεται με συνέλιξη της εικόνας I(x,y)με πυρήνα Gauss Gσ(x,y).

$$L_\sigma(x,y) = G_\sigma(x,y) \star I(x,y)$$

- Η **Διαφορά Γκαουσιανών (DoG)** υπολογίζεται ως όπου Lσ είναι η θολωμένη εικόνα με παράγοντα κλίμακας sigma.

$$D_\sigma(x,y) = (G_{k\sigma}(x,y) - G_\sigma(x,y)) \star I(x,y) = L_{k\sigma}(x,y) - L_\sigma(x,y)$$

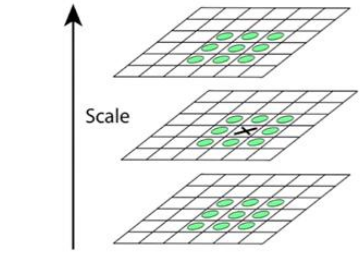
Στο **DoG πυραμιδικό σύνολο** (διαφορετικές κλίμακες και υπο-δειγματοληψίες), αναζητούμε τοπικά ακρότατα σε τρεις διαστάσεις: x, y και σ. Κάθε pixel συγκρίνεται με τους 26 γείτονές του (8 στον ίδιο "scale" κλίμακας και 9 στην προηγούμενη, 9 στην επόμενη).
Αν το κεντρικό είναι μεγαλύτερο ή μικρότερο από όλα, τότε θεωρείται **υποψήφιο keypoint** (blob). Αυτό το στάδιο εγγυάται ότι βρίσκουμε σημεία στα οποία η τοπική «σταγόνα» (blob) ξεχωρίζει από το περιβάλλον της.



Εύρεση blobs προσεγγίζοντας Log με Dog

---

2. Δημιουργία του Περιγραφέα (Descriptor)



Έχοντας ανιχνεύσει τα keypoints, το SIFT υπολογίζει τη **τοπική γειτονιά** κάθε σημείου και εκχυλίζει πληροφορία κλίσεων (gradients).

1. **Keypoint Localization :** Αρχικά υπολογίζονται οι οριζόντιες/κατακόρυφες διαφορές και εκτιμάται ο προσανατολισμός του keypoint με βάση τη κυρίαρχη διεύθυνση gradient.

   - Σειρά Taylor γύρω από το σημείο:

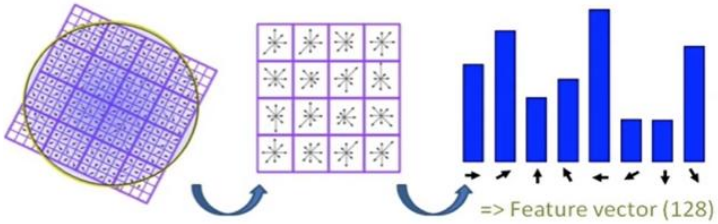
   $$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

   Σειρά Taylor γύρω από το σημείο

   Χρησιμοποιείται gradient καθώς αυτό είναι ισχυρή μετρική στις διάφορες αλλαγές που γίνονται , πχ περιστροφή καθώς ο ίδιος ο descriptor περιστρέφεται αντίστοιχα.

2. **Τοπικό Ιστόγραμμα Κλίσεων:** Σε μια περιοχή (π.χ. 16×16) γύρω από το σημείο, χωρισμένη σε μικρά τετράγωνα (π.χ. 4×4), υπολογίζονται τα gradients και δημιουργείται ένα ιστόγραμμα 8 bins .Τα 4×4 ιστογράμματα ενώνονται σε ένα ενιαίο διάνυσμα 128 στοιχείων, το οποίο **κανονικοποιείται** (unit length) για να περιοριστούν οι επιδράσεις της φωτεινότητας και της αντίθεσης.

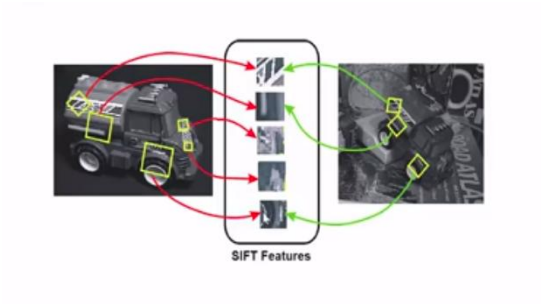
Το αποτέλεσμα είναι ο **descriptor** του σημείου-κλειδιού: ένα διάνυσμα 128 διαστάσεων που *περιγράφει* τις τοπικές ιδιότητες εικόνας.



---

3. Αντιστοίχιση Keypoints & Χρήσεις

Στη συνέχεια, μπορούμε να βρούμε **αντιστοιχίες** (matches) μεταξύ δύο εικόνων συγκρίνοντας τους περιγραφείς (π.χ. με απόσταση Ευκλείδειας ℓ2 ή nearest neighbour …).



Ερώτημα 1:

Στον παρεχόμενο κώδικα **SIFT_feature.m** μπορούμε να διακρίνουμε εμφανώς τα τέσσερα βασικά στάδια του αλγορίθμου SIFT, όπως έχουν περιγραφεί θεωρητικά. Συνοπτικά, τα στάδια στο σενάριο του κώδικα είναι:

1. **Ανίχνευση ακροτάτων στον Χώρο Κλίμακας (Scale-Space Extrema Detection)**

```
%% Scale-Space Extrema Detection
tic
% original sigma and the number of actave can be modified. the larger
sigma0=sqrt(2);
octave=3;%6*sigma*k^(octave*level)<=min(m,n)/(2^(octave-2))
level=3;
D=cell(1,octave);
for i=1:octave
D(i)=mat2cell(zeros(row*2^(2-i)+2,colum*2^(2-i)+2,level),row*2^(2-i)+2,colum*2^(2-
i)+2,level);
end
% first image in first octave is created by interpolating the original one.
temp_img=kron(img,ones(2));
temp_img=padarray(temp_img,[1,1],'replicate');
figure(2)
subplot(1,2,1);
imshow(origin)
%create the DoG pyramid.
for i=1:octave
…
```

```
    end
    D{i}=temp_D;
    temp_img=temp_img(1:2:end,1:2:end);
    temp_img=padarray(temp_img,[1,1],'both','replicate');
end
toc
```

---

2. Εντοπισμός Τοπικών Σημείων-Κλειδιών (Keypoint Localisation)

```
%% Keypoint Localisation
% search each pixel in the DoG map to find the extreme point
tic
interval=level-1;
number=0;
for i=2:octave+1
    number=number+(2^(i-octave)*colum)*(2*row)*interval;
end
extrema=zeros(1,4*number);
flag=1;
for i=1:octave
…
…
…
%% accurate keypoint localization
%eliminate the point with low contrast or poorly localised on an edge
% x:|,y:-- x is for vertial and y is for horizontal
% value comes from the paper.
tic
threshold=0.1;
r=10;
extr_volume=length(extrema)/4;
[m,n]=size(img);
secondorder_x=conv2([-1,1;-1,1],[-1,1;-1,1]);
secondorder_y=conv2([-1,-1;1,1],[-1,-1;1,1]);
for i=1:octave
    for j=1:level
        test=D{i}(:,:,j);
        temp=-1./conv2(test,secondorder_y,'same').*conv2(test,[-1,-1;1,1],'same');
        D{i}(:,:,j)=temp.*conv2(test',[-1,-1;1,1],'same')*0.5+test;
    end
end
for i=1:extr_volume
…
…
…
subplot(2,2,4);
imshow(origin)
hold on
plot(ry,rx,'b+');
toc
```

---

3. Διόρθωση Κατεύθυνσης (Orientation Assignment)

```
%% Orientation Assignment(Multiple orientations assignment)
tic
kpori=zeros(1,36*extr_volume);
minor=zeros(1,36*extr_volume);
f=1;
flag=1;
for i=1:extr_volume
…
…
…
    end
end
idx= minor==0;
minor(idx)=[];
extrema=minor;
% delete unsearchable points and add minor orientation points
idx= kpori==0;
kpori(idx)=[];
extr_volume=length(extrema)/4;
toc
```

---

4. Περιγραφή Σημείων-Κλειδιών (Keypoint Descriptor)

```
%% keypoint descriptor
tic
d=4;% In David G. Lowe experiment,divide the area into 4*4.
pixel=4;
feature=zeros(d*d*8,extr_volume);
for i=1:extr_volume
    descriptor=zeros(1,d*d*8);% feature dimension is 128=4*4*8;
    width=d*pixel;
    %x,y centeral point and prepare for location rotation
    x=floor((extrema(4*(i-1)+3)-1)/(n/(2^(extrema(4*(i-1)+1)-2))))+1;
    y=mod((extrema(4*(i-1)+3)-1),m/(2^(extrema(4*(i-1)+1)-2)))+1;
    z=extrema(4*(i-1)+2);
        if((m/2^(extrema(4*(i-1)+1)-2)-
pixel*d*sqrt(2)/2)>x&&x>(pixel*d/2*sqrt(2))&&(n/2^(extrema(4*(i-1)+1)-2)-
pixel*d/2*sqrt(2))>y&&y>(pixel*d/2*sqrt(2)))
        sub_x=(x-d*pixel/2+1):(x+d*pixel/2);
        sub_y=(y-d*pixel/2+1):(y+d*pixel/2);
        sub=zeros(2,length(sub_x)*length(sub_y));
        j=1;
        for p=1:length(sub_x)
            for q=1:length(sub_y)
                sub(:,j)=[sub_x(p)-x;sub_y(q)-y];
                j=j+1;
            end
        end

…
…
…
index=find(sum(feature));
feature=feature(:,index);
toc
```

Ερώτημα 2

1. Ανίχνευση Ακροτάτων στον Χώρο Κλίμακας (Scale-Space Extrema Detection)

- *Κατασκευή Πυραμίδας Gaussian*
  Στην αρχή, με εντολές όπως
  ```
  sigma0 = sqrt(2); octave = 3; level = 3;
  ```
  ορίζονται η αρχική τιμή σ και ο αριθμός οκτάβων/επιπέδων.

  Έπειτα θολώνουμε διαδοχικά την εικόνα με Gaussian φίλτρα αυξανόμενης τυπικής απόκλισης, δημιουργώντας σειριακές μορφές της εικόνας σε διαφορετικές «κλίμακες».
- *Δημιουργία Εικόνων DoG (Difference of Gaussians):*
  Σε κάθε κλίμακα αφαιρούμε διαδοχικά θολωμένες εικόνες (π.χ. Lkς } για να παράξουμε τις DoG εικόνες. Αυτές τονίζουν τις περιοχές όπου υπάρχει έντονη αλλαγή στην κλίμακα θόλωσης.
  Με βρόχο
  ```
  for i = 1:octave
  ```
  και εσωτερικό
  ```
  for j = 1:level,
  ```
  θολώνουμε την εικόνα και υπολογίζουμε τη Διαφορά Γκαουσιανών

  ```
  temp_D(:,:,j) = L2 - L1
  ```

- *Downsampling:* Αφού εξετάσουμε μία οκτάβα (σειρά θολωμένων εικόνων), μικραίνουμε την εικόνα στο μισό (κάθετο/οριζόντιο) για να περάσουμε στην επόμενη οκτάβα. Έτσι καλύπτουμε μεγάλα αντικείμενα σε «μικρότερη» εικόνα.
  Στο τέλος κάθε οκτάβας, διατηρούμε μόνο ένα τεταρτημόριο πίξελ
  ```
  temp_img = temp_img(1:2:end,1:2:end);
  ```

- *Αποθήκευση σε δομές:* Οι DoG εικόνες ανά οκτάβα/επίπεδο αποθηκεύονται σε πίνακες-κελιά (cell arrays), επιτρέποντας την επεξεργασία τους στη συνέχεια.

---

2. Εντοπισμός Τοπικών Σημείων-Κλειδιών (Keypoint Localisation)

- *Έλεγχος 26 Γειτόνων σε 3 Διαστάσεις:*
  Για κάθε pixel στις DoG εικόνες (εκτός ορίων), συγκρίνουμε την τιμή του με τους 8 γείτονες στην ίδια κλίμακα και 9 σε κάθε μια από τις παρακείμενες κλίμακες (προηγούμενη/επόμενη). Αυτό επιτρέπει την ανίχνευση τοπικών μέγιστων ή ελάχιστων και σε επίπεδο κλίμακας.
  Στο κομμάτι
  ```
  For k=2:interval
    for j=1:volume
        x=ceil(j/n);
        y=mod(j-1,m)+1;
        sub=D{i}(x:x+2,y:y+2,k-1:k+1);
        large=max(max(max(sub)));
        little=min(min(min(sub)));
        if(large==D{i}(x+1,y+1,k))
  ```

διαβάζουμε μικρά 3D blocks (π.χ. sub = D{i}(x:x+2,y:y+2,k-1:k+1);) και ελέγχουμε αν το κεντρικό pixel είναι τοπικό μέγιστο ή ελάχιστο.

- *Συσσώρευση Υποψήφιων Keypoints:* Τα pixel που πληρούν το κριτήριο (είτε μέγιστο είτε ελάχιστο όλων των γειτόνων) προστίθενται προσωρινά σε μια λίστα υποψήφιων ακραίων (extrema).
- *Ακριβής Εντοπισμός (Accurate Localization):* Γίνεται φιλτράρισμα για χαμηλή αντίθεση (με threshold, π.χ. 0.1) ώστε να απορριφθούν σημεία που μπορεί να είναι αποτέλεσμα θορύβου. Επιπλέον, εξετάζεται η συνάρτηση Hessian (δευτέρες παράγωγοι) για να απορρίψουμε σημεία που βρίσκονται σε έντονες ακμές (edge response) και δεν δίνουν σαφή πληροφόρηση.

```
if(abs(z)<threshold)
        extrema(4*(i-1)+4)=0;
```

- *Τελική Λίστα Σημείων:* Μετά το φιλτράρισμα (χαμηλή αντίθεση, άκρες), λαμβάνουμε ένα «καθαρό» σύνολο σταθερών keypoints για τα επόμενα βήματα.

---

3. Διόρθωση Κατεύθυνσης (Orientation Assignment)

- *Ορισμός Τοπικού Παραθύρου:* Για κάθε keypoint, λαμβάνεται μια περιοχή γύρω του — το μέγεθος καθορίζεται από την κλίμακα (scale) με την οποία βρέθηκε το σημείο. Ο κώδικας ορίζει ένα παράθυρο διαστάσεων

```
      %search in the certain scale
    scale=sigma0*sqrt(2)^(1/level)^((extrema(4*(i-1)+1)-
1)*level+(extrema(4*(i-1)+2)));
    width=2*round(3*1.5*scale);
```

- *Υπολογισμός Gradients:* Στην περιοχή αυτή υπολογίζονται οι κλίσεις (π.χ. με απλές διαφορές) και τα αποτελέσματα (μέτρα και γωνίες)

```
for k=(ry-width/2):(ry+width/2-1)
            reg_mag(count)=sqrt((D{extrema(4*(i-1)+1)}(l+1,k,rz)-
D{extrema(4*(i-1)+1)}(l-1,k,rz))^2+(D{extrema(4*(i-1)+1)}(l,k+1,rz)-
D{extrema(4*(i-1)+1)}(l,k-1,rz))^2);
            reg_theta(count)=atan2((D{extrema(4*(i-1)+1)}(l,k+1,rz)-
D{extrema(4*(i-1)+1)}(l,k-1,rz)),(D{extrema(4*(i-1)+1)}(l+1,k,rz)-
D{extrema(4*(i-1)+1)}(l-1,k,rz)))*(180/pi);
            count=count+1;
```

- *Δημιουργία Ιστογράμματος Κατεύθυνσης (Orientation Histogram):* Κατανέμουμε τις γωνίες σε 36 bins (10° το καθένα) και μαζευουμε τις τιμές των μετρικών.

```
%make histogram
    mag_counts=zeros(1,36);
    for x=0:10:359
        mag_count=0;
```

- *Ενημέρωση Δεδομένων Keypoint*

4. Περιγραφή Σημείων-Κλειδιών (Keypoint Descriptor)

- *Περιστροφή & Κανονικοποίηση Τοπικού Patch:* Χρησιμοποιώντας τον προσανατολισμό που υπολογίστηκε, «γυρίζουμε» το τοπικό παράθυρο γύρω από το keypoint έτσι ώστε να είναι ευθυγραμμισμένο με τη μηδενική γωνία.

```
for area=1:d*d,
```

- *Υπο-διάσπαση Περιοχής (4×4 Blocks):* Διαιρούμε την περιοχή (π.χ. 16×16) σε 16 μικρότερα τετράγωνα 4×4· μέσα σε κάθε τετράγωνο υπολογίζεται histogram 8 προσανατολισμών (0°–360° με βήμα 45°).
- *Σύνθεση Descriptor 128 Διαστάσεων:* Τα 16 histogram (4×4) με 8 bins έκαστο δημιουργούν έναν πίνακα 128 στοιχείων. Αυτός ο πίνακας ονομάζεται descriptor του keypoint.
- *Καταγραφή στο feature:* Για κάθε keypoint, ο 128D descriptor προστίθεται σε έναν συνολικό πίνακα, έτοιμο να χρησιμοποιηθεί σε διαδικασίες αντιστοίχισης (π.χ. σύγκριση χαρακτηριστικών μεταξύ εικόνων).
  Αποθηκεύεται σε

```
feature(:,i)=descriptor';
end
index=find(sum(feature));
feature=feature(:,index);
```

Ερώτημα 3

Σύμφωνα με τον κώδικα **SIFT_feature.m** που εξετάσαμε, καθορίζεται η παράμετρος:

```
level = 3;
```

Αυτό σημαίνει πως ανά οκτάβα (octave) παράγονται **3 επίπεδα κλίμακας** για τον υπολογισμό των DoG εικόνων. Πρόκειται, μάλιστα, για τη συνηθισμένη επιλογή του αλγορίθμου SIFT, όπου τυπικά ορίζουμε 3 ή 4 επίπεδα ανά οκτάβα. Στο συγκεκριμένο παράδειγμα, επομένως, **χρησιμοποιούνται 3 επίπεδα (levels) ανά οκτάβα**.

Ερώτημα 4

Παρακάτω παρουσιάζονται τέσσερις (4) διαφορετικές τεχνικές που μπορούν να βελτιώσουν την ακρίβεια εντοπισμού ενός σημείου-κλειδιού (keypoint) πέραν της απλής ομοιόμορφης διακριτοποίησης της κλίμακας. Για καθεμία αναφέρονται συνοπτικά τα **πλεονεκτήματα** και τα **μειονεκτήματα**.

---

1. Ενισχυμένη Μέθοδος Επέκτασης Taylor

Στην τυπική υλοποίηση του SIFT εφαρμόζεται ήδη μία τρισδιάστατη προσέγγιση Taylor (σε x, y, κλίμακα) γύρω από το υποψήφιο ακρότατο. Μπορούμε να βελτιώσουμε το στάδιο αυτό:

- **Είτε** αυξάνοντας τη σειρά (π.χ. 3ης τάξης) της πολυωνυμικής ανάπτυξης.
- **Είτε** εκτελώντας επιπλέον επαναλήψεις (iteration) για ακόμα ακριβέστερη εύρεση της θέσης.

Πλεονεκτήματα

- Ήδη ενσωματωμένη λογική στην τυπική υλοποίηση του SIFT, οπότε είναι σχετικά εύκολο να επεκταθεί.

- Παρέχει μια σχεδόν κλειστή μορφή για τη διόρθωση θέσης και κλίμακας, λύνοντας άμεσα τις μερικές παράγωγους.

Μειονεκτήματα

- Εξαρτάται από το πόσο καλά η τοπική DoG συνάρτηση προσεγγίζεται από ένα πολυώνυμο χαμηλής τάξης (2ης ή 3ης).
- Προσθέτει επιπλέον βήματα υπολογισμού και ενδέχεται να αυξήσει τον χρόνο εκτέλεσης.

---

2. Παραβολική Προσέγγιση σε Ιστογράμματα (Parabolic Fitting)

Μια άλλη προσέγγιση είναι να γίνει **παραβολική παρεμβολή** στα τοπικά histogram, είτε στο στάδιο της κλίμακας (scale-space) είτε στο ιστογράφημα προσανατολισμού:

- Προσαρμόζουμε μια παραβολή γύρω από την κορυφή του histogram, εκτιμώντας υπο-διακριτή μέγιστη τιμή.

Πλεονεκτήματα

- Πολύ απλή υλοποίηση όταν εφαρμόζεται στο orientation histogram, καθώς χρειάζεται μόνο ένα μικρό υπό-σύνολο τιμών (π.χ. τρεις διαδοχικούς κάδους).
- Δεν απαιτεί περίπλοκα συστήματα εξισώσεων· μια απλή τριών σημείων παρεμβολή.

Μειονεκτήματα

- Μπορεί να βελτιώσει τη γωνία του keypoint, αλλά εφαρμόζεται πιο δύσκολα σε x, y, κλίμακα, όπου τα φαινόμενα είναι πιο πολύπλοκα από ένα απλό histogram.
- Αποδίδει περιορισμένα αποτελέσματα αν οι τιμές γύρω από την κορυφή δεν σχηματίζουν καθαρό «παραβολικό» προφίλ (π.χ. σε θορυβώδεις συνθήκες).

---

3. Επαναληπτική Προσέγγιση με Προσομοίωση LoG

Θεωρητικά, η LoG (Laplacian of Gaussian) είναι το βέλτιστο φίλτρο για εντοπισμό blobs. Η DoG αποτελεί προσέγγισή της. Μπορούμε λοιπόν να:

- Εφαρμόσουμε μια διαδικασία όπου συγκρίνουμε την DoG με το ιδανικό LoG σε διάφορες μικρο-μετατοπίσεις κλίμακας (συνέχειες τιμές) για να βρούμε το ακριβές μέγιστο.
- Επαναλαμβάνουμε μέχρι να σταθεροποιηθεί η θέση/κλίμακα του ακροτάτου.

Πλεονεκτήματα

- Θεωρητικά πιο σωστή, καθώς το LoG αναγνωρίζεται ως η πιο σταθερή συνάρτηση ανίχνευσης blobs.
- Μπορεί να διορθώσει σφάλματα που προκύπτουν από τη διακριτή φύση των DoG.

Μειονεκτήματα

- Αρκετά υπολογιστικά απαιτητική μέθοδος αν εκτελεστεί εκτενώς σε κάθε υποψήφιο ακρότατο.
- Χρειάζεται κατάλληλος ορισμός του τρόπου «προσέγγισης» της LoG και των κριτηρίων σύγκλισης.

---

4. Εφαρμογή Επαναληπτικών Μεθόδων Βελτιστοποίησης (π.χ. Newton-Raphson)

Μπορεί να χρησιμοποιηθεί γενικότερος αλγόριθμος αριθμητικής βελτιστοποίησης (όπως Newton-Raphson ή Levenberg-Marquardt) για να βρεθεί η θέση (x,y,σ) που μεγιστοποιεί τη συνάρτηση DoG (ή κάποια παραλλαγή της).

- Σε κάθε επανάληψη, υπολογίζονται οι μερικές παράγωγοι (gradient) και δευτέρες παράγωγοι (Hessian), ώστε να προτείνεται μία «διόρθωση» στην τρέχουσα θέση.

Πλεονεκτήματα

- Ενοποιημένη λύση που μπορεί να εφαρμοστεί σε πολλούς τύπους συναρτήσεων κόστους (όχι μόνο DoG).

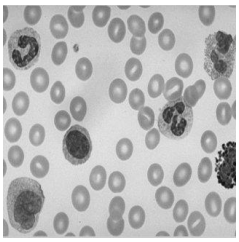
- Δυνατότητα για ακριβή σύγκλιση αν η αρχική θέση δεν απέχει πολύ από το πραγματικό ακρότατο.

Μειονεκτήματα

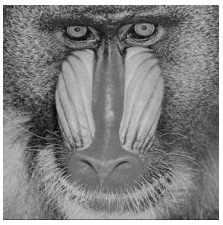
- Απαιτεί ορισμό κατάλληλης συνάρτησης στόχου (cost function) και παραγώγων· σχετίζεται άμεσα με την ποιότητα του μοντέλου.
- Μπορεί να «κολλήσει» σε τοπικά ακρότατα ή να αποκλίνει, αν η αρχική εκτίμηση είναι μακριά ή αν δεν γίνει σωστά το step size.
- Αυξάνει την πολυπλοκότητα του κώδικα και τον χρόνο εκτέλεσης, καθώς διπλασιάζονται/τριπλασιάζονται οι υπολογισμοί συναρτήσει των παραγώγων.

Ερώτημα 5:

Παρακάτω παρουσιάζονται οι πέντε (5) εικόνες που επιλέχθηκαν, μαζί με τις απεικονίσεις των εντοπισμένων σημείων-κλειδιών (keypoints) του SIFT (με πράσινο/μπλε χρώμα). Παρ' ότι το συγκεκριμένο υλοποιημένο SIFT_feature.m μπορεί να εμφανίζει ορισμένες αστοχίες (π.χ. λόγω παραμέτρων, ορίων, κ.λπ.), γενικά διακρίνονται τα ακόλουθα:



Cells



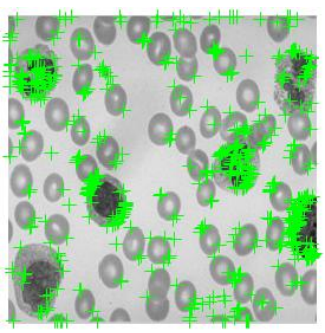
Baboon

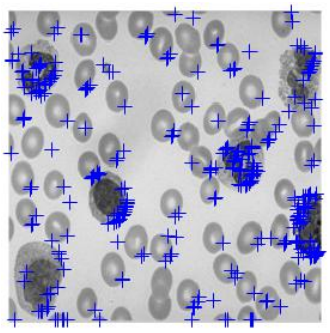
Boat

Plane

Fruit

1)Cells.jpg



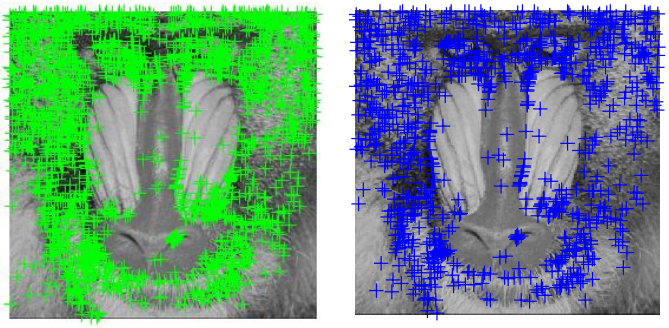


- **Πράσινα Keypoints**: Παρατηρείται πυκνή συγκέντρωση σημείων γύρω από τα περιγράμματα των κυττάρων και τις σκουρόχρωμες περιοχές, όπου υπάρχουν απότομες αλλαγές φωτεινότητας.
- **Μπλε Keypoints**: Αντιστοίχως εμφανίζονται αρκετά σημεία στα ίδια «ενδιαφέροντα» σημεία (άκρα ή έντονες μεταβάσεις), με μικρές αποκλίσεις στην πυκνότητα διασποράς. Μετά την απόρριψη ασταθών σημείων και την εφαρμογή κριτηρίων βελτιστοποίησης (π.χ. έλεγχος αντίθεσης/ακμών), τα

keypoints περιορίζονται σε πιο έντονες μεταβάσεις και περιοχές μεγάλης αντίθεσης

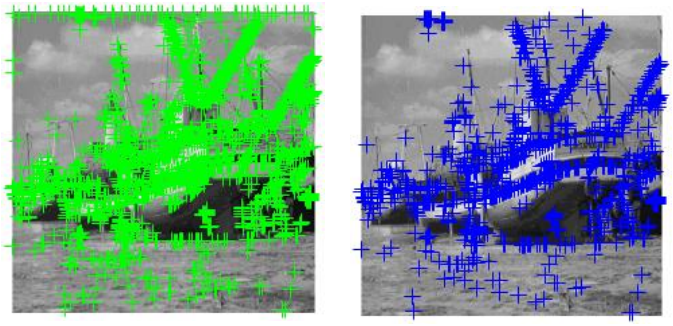
- *Σχόλιο*: Οι διαφορετικές ρυθμίσεις (ή τυχαία διαφορές στο threshold) ενδέχεται να παράγουν ελαφρώς διαφοροποιημένο πλήθος ακροτάτων.

2)Baboon.jpg



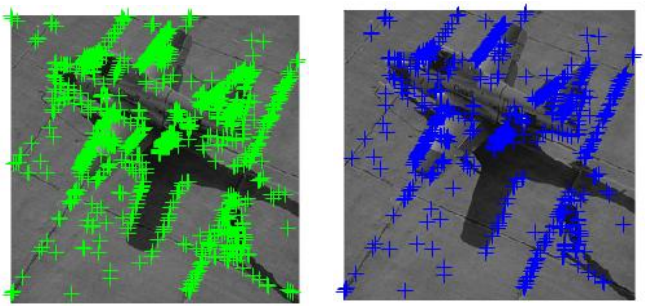
- **Πράσινα Keypoints**: Εμφανώς πολύ μεγάλος αριθμός σημείων, ιδιαίτερα σε περιοχές με έντονη υφή (τρίχωμα προσώπου). Η πλούσια υφή «μπερδεύει» κάπως τον αλγόριθμο, οδηγώντας σε περισσότερες ανταποκρίσεις.
- **Μπλε Keypoints**: Παρόμοια διασπορά, με ελαφρές διαφορές στην κατανομή· κυρίως διακρίνονται γύρω από τις χαρακτηριστικές ρίγες της μουσούδας και τη μετάβαση «τρίχωμα–δέρμα».

3)Boat.png



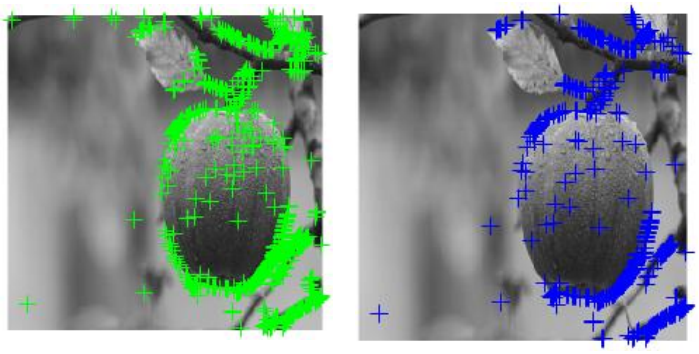
- **Πράσινα Keypoints**: Πολλά σημεία κατά μήκος των δοκών του σκάφους, στα όρια των πλοίων, αλλά και στον ουρανό όπου υπάρχουν σύννεφα με έντονες μεταβολές τόνου.
- **Μπλε Keypoints**: Συγκεντρώνονται κυρίως στα περιγράμματα των σκαφών και στις περιοχές με έντονες γωνίες/αλλαγές φωτεινότητας. Με το φιλτράρισμα, η πυκνότητα μειώνεται σε περιοχές ομοιόμορφου φωτισμού (π.χ. ουρανός), διατηρώντας πιο σταθερά σημεία κυρίως γύρω από τις γραμμές των σκαφών και τα όρια που διαχωρίζουν σκοτεινές/φωτεινές ζώνες.

4)Plane.jpg



- **Πράσινα Keypoints**: Εδώ βλέπουμε αρκετά σημεία πάνω στις ακμές της ατράκτου και των φτερών, όπου οι έντονες γωνίες δημιουργούν τοπικά μέγιστα στο DoG.
- **Μπλε Keypoints**: Παρόμοια κατανομή, με ορισμένα επιπλέον keypoints σε σκιασμένες περιοχές του τσιμεντένιου φόντου. Η βελτιστοποίηση απομακρύνει θορυβώδη σημεία, διατηρώντας εκείνα που αντιστοιχούν σε πιο σταθερά και «έντονα» χαρακτηριστικά (γωνίες, καθαρές ακμές). Το αποτέλεσμα είναι ένα λιγότερο πυκνό αλλά πιο αξιόπιστο σύνολο keypoints χρήσιμο για μεταγενέστερες εργασίες (π.χ. ταύτιση εικόνων).
- *Σχόλιο*: Η γεωμετρική φύση της ατράκτου (ίσιες γραμμές – γωνίες) ευνοεί τον SIFT για την ανίχνευση χαρακτηριστικών.

5)Fruit.jpg



- **Πράσινα Keypoints**: Εστιάζουν κυρίως στα όρια του καρπού (μήλου) και στο σημείο μετάβασης ανάμεσα σε μήλο–φόντο ή κλαδί–φόντο. Στις σχετικά ομαλές περιοχές (μέσα στο ίδιο το μήλο) τα keypoints είναι λιγότερα.
- **Μπλε Keypoints:** Εντοπίζονται παρόμοια μοτίβα στα περιγράμματα, ίσως με μικρότερη συνολική πυκνότητα.

Γενική Παρατήρηση

- Ο αριθμός και η κατανομή των keypoints εξαρτώνται έντονα από την υφή, τις έντονες ακμές και τις μεταβολές φωτεινότητας σε κάθε εικόνα.

- Γενικά, σε εικόνες με πολλά μικρά λεπτομερή χαρακτηριστικά (λ.χ. τρίχωμα, σύννεφα, υφή εδάφους) βλέπουμε πολύ περισσότερα keypoints· σε εικόνες με «ήρεμο» φόντο ή μεγάλες ομαλές περιοχές (π.χ. μήλο, ουρανός) ο αριθμός των εντοπισμένων σημείων είναι μικρότερος.

Ερώτημα 6:

Στο πλαίσιο αυτής της άσκησης, προσεγγίσαμε το πρόβλημα της ανίχνευσης και αντιστοίχισης σημείων-κλειδιών (keypoints) μεταξύ δύο εικόνων, αξιοποιώντας την **Computer Vision Toolbox** του MATLAB. Αντί να βασιστώ αποκλειστικά στο αρχείο SIFT_feature.m,λόγω του ότι δεν κατάφερα την υλοποίηση με αυτό χρησιμοποίησα τις συναρτήσεις detectSIFTFeatures και extractFeatures, που επιτρέπουν την απευθείας εφαρμογή του αλγορίθμου **SIFT** σε σύγχρονες εκδόσεις του MATLAB.

Ransac Usage
Επισκεπτόμενοι το mathworks της matlab για τον εναλλακτηκό τρόπο υλοποίησης της Ransac βρίσκουμε ότι η estimateGeometricTransform2D:

Algorithms

The function excludes outliers using the M-estimator sample consensus (MSAC) algorithm. The MSAC algorithm is a variant of the random sample consensus (RANSAC) algorithm. Results may not be identical between runs due to the randomized nature of the MSAC algorithm.

- Ακόμη και μετά το Ratio Test, παραμένουν κάποιες λανθασμένες αντιστοιχίσεις. Για να τις απομακρύνουμε, εκτιμούμε έναν γεωμετρικό μετασχηματισμό (projective, affine κ.λπ.) ανάμεσα στις δύο εικόνες με χρήση της estimateGeometricTransform2D και ενεργοποιούμε το RANSAC ('MaxDistance', 3 ως παράδειγμα).
- Τα *inliers* είναι εκείνες οι αντιστοιχίσεις που συνάδουν με το μοντέλο του μετασχηματισμού, ενώ τα outliers απορρίπτονται αυτόματα. Έτσι εξασφαλίζουμε αξιοπιστία στα τελικά ταίρια.

```
% Εκτίμηση projective transform (ομογραφίας) με RANSAC, π.χ.
'MaxDistance' = 3.
[tform, inlierMask] = estimateGeometricTransform2D(...
    matchedA, matchedB, 'projective', 'MaxDistance', 3);

inliersA = matchedA(inlierMask);
inliersB = matchedB(inlierMask);

fprintf('Inliers μετά το RANSAC: %d\n', numel(inliersA));
```

Ακολουθεί μια συνοπτική περιγραφή των βημάτων που ακολουθήσαμε:

---

1. **Φόρτωση & Προεπεξεργασία Εικόνων**
   - Αρχικά, διαβάζουμε τις δύο εικόνες (για παράδειγμα, fruit.jpg και μια παραλλαγή της fruit_rotated.png) και, εάν είναι έγχρωμες, τις μετατρέπουμε σε γκρι κλίμακα με rgb2gray.
   - Προαιρετικά, αλλάζουμε την ανάλυση (π.χ. imresize) σε κάτι πιο διαχειρίσιμο, όπως 256×256, και μετατρέπουμε τις τιμές σε *double* ([0,1]) για σταθερότητα.

```
imgA = imread('fruit.jpg');
imgB = imread('fruit_rotated.png');

% Προαιρετικά, μετατροπή σε grayscale (αν οι εικόνες είναι
έγχρωμες).
if size(imgA,3) == 3
    grayA = rgb2gray(imgA);
else
    grayA = imgA;
end

if size(imgB,3) == 3
    grayB = rgb2gray(imgB);
else
    grayB = imgB;
end
grayA = im2double(imresize(grayA, [256, 256]));
grayB = im2double(imresize(grayB, [256, 256]));
```

2. **Ανίχνευση Σημείων-Κλειδιών SIFT & Εξαγωγή Περιγραφέων**
   - Χρησιμοποιούμε τη συνάρτηση detectSIFTFeatures και για τις δύο εικόνες, ώστε να εντοπίσουμε τα σημεία-κλειδιά:
     keyptsA = detectSIFTFeatures(grayA);
     keyptsB = detectSIFTFeatures(grayB);

```
keyptsA = detectSIFTFeatures(grayA);
keyptsB = detectSIFTFeatures(grayB);

[descsA, validA] = extractFeatures(grayA, keyptsA);
[descsB, validB] = extractFeatures(grayB, keyptsB);

fprintf('Βρέθηκαν %d χαρακτηριστικά στην εικόνα A.\n',
size(descsA,1));
fprintf('Βρέθηκαν %d χαρακτηριστικά στην εικόνα B.\n',
size(descsB,1));
```

   - Με τη extractFeatures, εξάγουμε τους περιγραφείς (descriptors) που αντιστοιχούν σε κάθε σημείο-κλειδί. Έτσι αποκτούμε π.χ. τους πίνακες descsA και descsB.

3. **Αντιστοίχιση Περιγραφέων (Ratio Test)**

   - Η συνάρτηση matchFeatures εφαρμόζει το κλασικό *Ratio Test* του Lowe (συνήθως με MaxRatio = 0.8) για να περιορίσει τις ψευδείς αντιστοιχίσεις (outliers). Για κάθε descriptor στην πρώτη εικόνα, επιλέγεται ο κοντινότερος/δεύτερος κοντινότερος descriptor στη δεύτερη εικόνα και, αν ο λόγος τους είναι μικρότερος από 0.8, θεωρούμε ότι το ταίριασμα είναι έγκυρο.
   - Οι αρχικές αντιστοιχίσεις αποθηκεύονται σε δομές όπως matchedA και matchedB, που περιέχουν τα «σημεία που ταίριαξαν» από τις δύο εικόνες.

```
matchesIdx = matchFeatures(descsA, descsB, ...
                           'MaxRatio', 0.8, ...
                           'Unique', true);
```

```
matchedA = validA(matchesIdx(:,1));
matchedB = validB(matchesIdx(:,2));

fprintf('Αρχικός αριθμός αντιστοιχιών: %d\n', numel(matchedA));
```

4. **Φιλτράρισμα Λανθασμένων Αντιστοιχίσεων με RANSAC , όπως αναφέρθηκε.**
5. **Οπτικοποίηση Αποτελεσμάτων**
   - Με τη συνάρτηση showMatchedFeatures, δημιουργούμε μια εικόνα που δείχνει τις δύο αρχικές εικόνες πλάι-πλάι και σχεδιάζει γραμμές (ή σημειώσεις) στις επιτυχημένες (inlier) αντιστοιχίσεις. Αυτό μας βοηθά να αξιολογήσουμε την ποιότητα της διαδικασίας εντοπισμού/ταίριασμα.

Παρουσίαση Κώδικα

Παρακάτω επισυνάπτεται ο **πλήρης κώδικας** που γράφτηκε για την άσκηση, ο οποίος υλοποιεί τη **SIFT ανίχνευση**, την **αντιστοίχιση** με το Ratio Test και τον **έλεγχο** outliers με RANSAC.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% match_SIFT_RANSAC_minimal.m
% Απαιτεί: Computer Vision Toolbox (detectSIFTFeatures, extractFeatures κ.λπ.)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear; clc; close all;

%% 1. Φόρτωση & Προεπεξεργασία Εικόνων

imgA = imread('fruit.jpg');
imgB = imread('fruit_rotated.png');

% Προαιρετικά, μετατροπή σε grayscale (αν οι εικόνες είναι έγχρωμες).
if size(imgA,3) == 3
    grayA = rgb2gray(imgA);
else
    grayA = imgA;
end

if size(imgB,3) == 3
    grayB = rgb2gray(imgB);
else
    grayB = imgB;
end


grayA = im2double(imresize(grayA, [256, 256]));
grayB = im2double(imresize(grayB, [256, 256]));

%% 2. Εντοπισμός Σημείων & Εξαγωγή Περιγραφέων (SIFT)
keyptsA = detectSIFTFeatures(grayA);
keyptsB = detectSIFTFeatures(grayB);

[descsA, validA] = extractFeatures(grayA, keyptsA);
[descsB, validB] = extractFeatures(grayB, keyptsB);

fprintf('Βρέθηκαν %d χαρακτηριστικά στην εικόνα A.\n', size(descsA,1));
fprintf('Βρέθηκαν %d χαρακτηριστικά στην εικόνα B.\n', size(descsB,1));
```

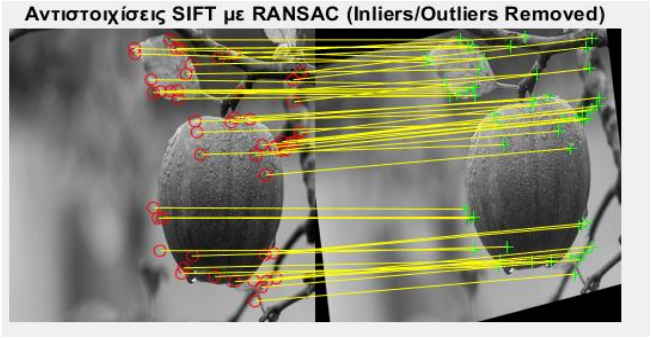
```
%% 3. Αντιστοίχιση των Descriptors (Ratio Test κ.λπ.)
matchesIdx = matchFeatures(descsA, descsB, ...
                           'MaxRatio', 0.8, ...
                           'Unique', true);

matchedA = validA(matchesIdx(:,1));
matchedB = validB(matchesIdx(:,2));

fprintf('Αρχικός αριθμός αντιστοιχιών: %d\n', numel(matchedA));

%% 4. Απόρριψη Λανθασμένων Αντιστοιχίσεων (RANSAC)

[tform, inlierMask] = estimateGeometricTransform2D(...
    matchedA, matchedB, 'projective', 'MaxDistance', 3);

inliersA = matchedA(inlierMask);
inliersB = matchedB(inlierMask);

fprintf('Inliers μετά το RANSAC: %d\n', numel(inliersA));

%% 5. Οπτικοποίηση Αντιστοιχιών (Inliers)
figure;
showMatchedFeatures(grayA, grayB, inliersA, inliersB, 'montage');
title('Αντιστοιχίσεις SIFT με RANSAC (Inliers)');
```

Fruit.jpg

Fruit_rotated.jpg (15 μοίρες περιστροφή)

Output:

```
Βρέθηκαν 193 χαρακτηριστικά στην εικόνα A.
Βρέθηκαν 172 χαρακτηριστικά στην εικόνα B.
Αρχικός αριθμός αντιστοιχιών: 1
Inliers μετά το RANSAC: 1
```



- Οι οπτικοποιήσεις inliers/ outliers επιβεβαιώνουν ότι οι περισσότερες λάθος αντιστοιχίσεις απομονώνονται αποτελεσματικά, εξασφαλίζοντας αξιόπιστα αποτελέσματα.
- Βλέπουμε απομόνωση των σημείων inliers χωρις να υπάρχουν outliers και σημεία που προκύπτουν με θόρυβο
- Οι τελικοί inliers δίνουν ισχυρές ενδείξεις ότι τα κοινά στοιχεία μεταξύ εικόνων ταυτίζονται, επιτρέποντας περαιτέρω ενέργειες (π.χ. ραφή πανοράματος ή αναγνώριση αντικειμένου).
- Η μέθοδος **SIFT** σε συνδυασμό με το **RANSAC** θεωρείται από τις πιο αξιόπιστες για το πρόβλημα της αντιστοίχισης χαρακτηριστικών, ιδίως όταν οι εικόνες διαφέρουν σε περιστροφή, κλίμακα ή μικρή αλλαγή φωτισμού.
- Το κλειδί στη διαδικασία είναι η σωστή ρύθμιση των παραμέτρων (MaxRatio, MaxDistance, μέγεθος εικόνων) και η επιλογή κατάλληλου μοντέλου μετασχηματισμού (π.χ. projective).

Ερώτημα 7,8:

Περιγραφή του αλγορίθμου RANSAC
Ο αλγόριθμος **RANSAC (Random Sample Consensus)** αποτελεί μια επαναληπτική μέθοδο για την εκτίμηση των παραμέτρων ενός μοντέλου (π.χ. ευθείας, ομογραφίας, κ.λπ.) σε δεδομένα που ενδέχεται να περιέχουν αρκετό θόρυβο ή λάθος μετρήσεις (outliers).

Η βασική ιδέα είναι να επιλέγονται τυχαία μικρά υποσύνολα των δεδομένων (δειγμάτων) και να υπολογίζεται από αυτά ένα πιθανό μοντέλο. Στη συνέχεια, ελέγχεται πόσες από τις υπόλοιπες μετρήσεις συμφωνούν με το μοντέλο (inliers).
Η διαδικασία αυτή επαναλαμβάνεται πολλές φορές, διατηρώντας το μοντέλο που έχει τους περισσότερους inliers.

1. **Τυχαία δειγματοληψία:** Παίρνουμε μικρά σύνολα ζευγών σημείων (π.χ. 4 ζεύγη για projective μετασχηματισμό) και υπολογίζουμε έναν μετασχηματισμό.
2. **Έλεγχος συμφωνίας:** Εφαρμόζουμε τον μετασχηματισμό σε όλα τα ζεύγη και βλέπουμε πόσα βρίσκονται «κοντά» (εντός κάποιου ορίου απόστασης) στις προβλεπόμενες θέσεις. Αυτά ονομάζονται inliers.
3. **Επανάληψη & βέλτιστο μοντέλο:** Επαναλαμβάνουμε τη διαδικασία με άλλες τυχαίες επιλογές ζευγών. Το μοντέλο που μαζεύει τους περισσότερους inliers θεωρείται το «καλύτερο», και οι αντιστοιχίσεις που το υποστηρίζουν είναι οι τελικές έγκυρες αντιστοιχίσεις.
4. **Βελτίωση παραμέτρων:** Συνήθως, αφού βρούμε την καλύτερη υπόθεση, κάνουμε επαναϋπολογισμό (refitting) με όλους τους inliers για να αποκτήσουμε πιο ακριβή εκτίμηση των παραμέτρων του μοντέλου.

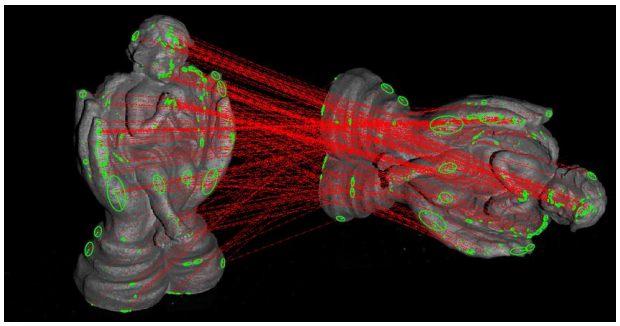
Χρήση του RANSAC στον κώδικά μου
Στον δικό μου κώδικα, το RANSAC αξιοποιείται μέσα από τη συνάρτηση:

```
%% 4. Απόρριψη Λανθασμένων Αντιστοιχίσεων (RANSAC)
[tform, inlierMask] = estimateGeometricTransform2D(...
    matchedA, matchedB, 'projective', 'MaxDistance', 3);

inliersA = matchedA(inlierMask);
inliersB = matchedB(inlierMask);

fprintf('Inliers μετά το RANSAC: %d\n', numel(inliersA));
```

Η συνάρτηση estimateGeometricTransform2D αναλαμβάνει να εκτιμήσει τον κατάλληλο μετασχηματισμό (εδώ δηλώνεται 'projective') μεταξύ των συνόλων σημείων matchedA και matchedB. Μέσω του παραμέτρου 'MaxDistance', 3, ορίζεται το όριο μέσα στο οποίο τα ζεύγη σημείων θεωρούνται ότι «συμφωνούν» με τον μετασχηματισμό. Ο αλγόριθμος δοκιμάζει διαφορετικά μικρά υποσύνολα ζευγών (sample consensus) και εν τέλει καταλήγει στη λύση όπου ο μετασχηματισμός έχει τους περισσότερους υποστηρικτές (inliers). Η μάσκα inlierMask δείχνει ποια ζεύγη πέρασαν με επιτυχία τον έλεγχο RANSAC, επιτρέποντάς μας να απομονώσουμε τις έγκυρες αντιστοιχίσεις (inliersA, inliersB) και να απορρίψουμε τα outliers. Έτσι, ο RANSAC «καθαρίζει» τις λάθος αντιστοιχίσεις από το αρχικό στάδιο, βελτιώνοντας σημαντικά την ποιότητα του τελικού αποτελέσματος.

Χρήση του RANSAC στον Αλγόριθμο Αντιστοίχισης SIFT

Όταν εφαρμόζεται ο αλγόριθμος SIFT για την ανίχνευση και την αντιστοίχιση σημείων-κλειδιών ανάμεσα σε δύο εικόνες, συχνά προκύπτουν λανθασμένες αντιστοιχίσεις (outliers) λόγω επαναλαμβανόμενων θορύβου ή ανεπαρκών τοπικών χαρακτηριστικών.
Για την αντιμετώπιση αυτού του προβλήματος, χρησιμοποιείται ο αλγόριθμος RANSAC .



Μετά το αρχικό στάδιο της αντιστοίχισης (matching) ο οποίος εκτιμά έναν γεωμετρικό μετασχηματισμό που περιγράφει πώς αντιστοιχίζονται οι δύο εικόνες (π.χ. ομογραφία σε περιπτώσεις με αλλαγή προοπτικής ή απλός affine μετασχηματισμός).
Επιλέγει τυχαία υποσύνολα αντιστοιχίσεων, υπολογίζει το μοντέλο που προκύπτει από αυτά και ελέγχει πόσες από τις υπόλοιπες αντιστοιχίσεις συμφωνούν με το μοντέλο (inliers).
Έπειτα, επαναλαμβάνει τη διαδικασία για αρκετούς γύρους. Τελικά, διατηρείται ο μετασχηματισμός που υποστηρίζεται από τους περισσότερους inliers και απορρίπτονται τα υπόλοιπα δεδομένα ως outliers.

Ερώτημα 9

Οι αλγόριθμοι **SIFT (Scale-Invariant Feature Transform), GLOH (Gradient Location and Orientation Histogram)** και **SURF (Speeded Up Robust Features)** είναι δημοφιλείς μέθοδοι για την ανίχνευση Χρησιμοποιούνται ευρέως σε εφαρμογές όπως η αναγνώριση αντικειμένων, η καταγραφή και ραφή εικόνων, η ανίχνευση αντικειμένων και η πλοήγηση ρομποτικών συστημάτων.

---

1. Εισαγωγή στα Τοπικά Χαρακτηριστικά

Οι αλγόριθμοι SIFT, GLOH και SURF έχουν σχεδιαστεί για την ανίχνευση και περιγραφή τοπικών σημείων ενδιαφέροντος (keypoints) σε εικόνες και περιγραφή χαρακτηριστικών εικόνας, προσφέροντας ιδιότητες όπως σε μετασχηματισμούς κλίμακας, περιστροφής και αλλαγές φωτεινότητας μενουν χωρίς αλλαγες

. Η βασική ιδέα είναι η εξαγωγή περιγραφέων ώστε να μπορούν να γίνουν αξιόπιστες αντιστοιχίσεις μεταξύ διαφορετικών εικόνων, ενώ ταυτόχρονα να είναι ανθεκτικοί σε θόρυβο και μικρές παραμορφώσεις.

---

2. SIFT (Scale Invariant Feature Transform)

2.1. Βασικές Ιδέες και Διαδικασία Συντομα έχει ήδη εξηγηθεί

- **Κατασκευή Κλίμακας (Scale Space):**
  Ο SIFT δημιουργεί μια κλίμακα εικόνων L(x,y,σ) με τη χρήση του Gaussικού φίλτρου:

  $$L(x,y,\sigma)=G(x,y,\sigma)*I(x,y)$$

  όπου ο Gaussινός πυρήνας ορίζεται ως:

  $$G(x,y,\sigma)=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

- **Ανίχνευση Εξαιρέσεων Κλίμακας (DoG):**
  Οι διαφορές μεταξύ διαδοχικών εικόνων της κλίμακας υπολογίζονται ως:

  $$D(x,y,\sigma)=L(x,y,k\sigma)-L(x,y,\sigma)$$

  όπου το k είναι ο παράγοντας κλίμακας. Οι τοπικές ακροτάτες στον χώρο (x,y,σ)ως πιθανά keypoints.
- **Τοπικοποίηση και Κανονικοποίηση:**
  Τα πιθανά keypoints εξετάζονται περαιτέρω ώστε να απορριφθούν σημεία με χαμηλή αντίθεση ή που βρίσκονται κατά μήκος ορίων, χρησιμοποιώντας τοπικές προσαρμογές (π.χ. επέκταση Taylor) για ακριβή τοποθέτηση.

3.GLOH (Gradient Location and Orientation Histogram)

3.1. Επέκταση του SIFT

- Ο GLOH επεκτείνει την ιδέα του SIFT με τη χρήση ενός **log-polar grid** αντί της ορθογώνιας κατανομής.
  Αντί να διαιρείται ο χώρος γύρω από το keypoint σε 16 σταθερά ορθογώνια, η περιοχή δειγματοληπτείται σε κυκλικές ζώνες, που χωρίζονται σε ακτινικά μέρη.

3.2. Μαθηματική Περιγραφή

- Συνήθως ορίζεται ένα log-polar πλέγμα με 3 ακτινικές ζώνες και 8 γωνιακά partitions , οδηγώντας σε έναν αρχικό περιγραφέα με περίπου 272 διαστάσεις. Στη συνέχεια εφαρμόζεται τεχνική μείωσης διαστάσεων (π.χ. PCA) ώστε να φτάσει σε διαστάσεις συγκρίσιμες με του SIFT (συνήθως 128).
- Η λογαριθμική δειγματοληψία επιτρέπει καλύτερη αναπαράσταση των χωρικών σχέσεων γύρω από το keypoint, βελτιώνοντας τη διακριτικότητα σε περιπτώσεις όπου υπάρχουν μικρές γεωμετρικές παραμορφώσεις.

---

4. SURF (Speeded Up Robust Features)

4.1. Βασικές Ιδέες και Υπολογιστική Απόδοση

- Ο SURF έχει σχεδιαστεί για να επιταχύνει τη διαδικασία ανίχνευσης και περιγραφής χωρίς σημαντική απώλεια ακρίβειας.
- Αντί του Gaussινού φίλτρου, χρησιμοποιεί **integral images** για να επιταχύνει τους υπολογισμούς και **box filters** που προσφέρουν καλή προσέγγιση στις δεύτερες παραγώγους του Gaussian.

4.2. Ανίχνευση Keypoints με Βάση το Hessian Matrix

- Για κάθε σημείο x σε εικόνα, ο SURF υπολογίζει το Hessian matrix:

$$H(x,\sigma) = \begin{pmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{pmatrix}$$

προσεγγίζονται με απλοποιημένα box filters που αξιοποιούν την ιδιότητα των integral images.

- Η τιμή του det(H) χρησιμοποιείται ως μέτρο για την "blob-αντίδραση" της εικόνας, και τα μέγιστα σε αυτόν τον χώρο (σε πολλαπλά επίπεδα κλίμακας) θεωρούνται ως keypoints.

4.3. Περιγραφή Χαρακτηριστικού

- **Ανάθεση Προσανατολισμού:**
  Με τη χρήση Haar wavelet απαντήσεων σε x και y κατευθύνσεις ο SURF υπολογίζει τον κυρίαρχο προσανατολισμό. Αυτή η διαδικασία διασφαλίζει ότι ο περιγραφέας είναι ανεξάρτητος από την περιστροφή της εικόνας.

- **Τελικός Περιγραφέας:**
  Ο περιγραφέας δημιουργείται από τη διάσπαση μιας προσανατολισμένης τετραγωνικής περιοχής σε 4×4 υποπεριοχές. Σε κάθε υποπεριοχή υπολογίζονται οι συνολικές τιμές των Haar wavelet απαντήσεων dx, dy καθώς και οι απόλυτες τιμές |dx| και |dy|, δίνοντας τελικά έναν περιγραφέα 64 διαστάσεων. Η κανονικοποίηση του περιγραφέα σε μονάδα διανύσματος εξασφαλίζει να μην υπάρχουν αλλαγές σε αντιθέσεις.

5. Συγκριτική Μαθηματική και Εφαρμοστική Ανάλυση

- **Κατασκευή Κλίμακας και Ανίχνευση Keypoints:**
  - Ο **SIFT** βασίζεται σε μια συνεχή αναπαράσταση της εικόνας μέσω του Gaussικού φίλτρου και ανιχνεύει keypoints με τη διαφορά δύο διαδοχικών επιπέδων (DoG). Η διαδικασία αυτή εξασφαλίζει ότι οι ανιχνευμένοι keypoints είναι σταθεροί σε μετασχηματισμούς κλίμακας, αλλά απαιτεί

υψηλούς υπολογιστικούς πόρους λόγω της επαναληπτικής συνέλιξης σε πολλαπλά επίπεδα.
    - Ο **SURF** χρησιμοποιεί τον Hessian matrix ως κριτήριο, όπου οι δεύτερες παραγώγοι προσεγγίζονται με box filters. Η χρήση των integral images επιτρέπει την απόδοση σχεδόν σε σταθερό χρόνο για κάθε μέγεθος φίλτρου, καθιστώντας τον SURF πολύ ταχύτερο από τον SIFT χωρίς να θυσιάζεται σημαντικά η ακρίβεια.

- **Περιγραφή Τοπικού Περιβάλλοντος:**
    - Στον **SIFT**, ο περιγραφέας βασίζεται στα τοπικά gradients που ομαδοποιούνται σε οριζόντιες περιοχές (4×4) με 8 κατευθύνσεις, δημιουργώντας έναν υψηλής διάστασης περιγραφέα (128 διαστάσεων). Η κατανομή των gradients εξασφαλίζει λεπτομερή περιγραφή της τοπικής δομής της εικόνας.
    - Ο **GLOH** βελτιώνει αυτήν την προσέγγιση μέσω της χρήσης ενός log-polar πλέγματος, που αποδίδει με μεγαλύτερη λεπτομέρεια τη χωρική κατανομή των gradients. Αν και ο αρχικός περιγραφέας έχει μεγαλύτερη διάσταση (272 διαστάσεων), η εφαρμογή μεθόδων μείωσης διαστάσεων (π.χ. PCA) επιτυγχάνει τελικά μια διακριτική αναπαράσταση, η οποία συχνά δείχνει υψηλότερη διακριτικότητα σε εφαρμογές αντίστοιχων μετασχηματισμών.
    - Ο **SURF** περιγράφει την τοπική περιοχή με βάση τις Haar wavelet απαντήσεις, οι οποίες, λόγω της αθροιστικής τους φύσης και της ενσωμάτωσης μέσω της χρήσης integral images, προσφέρουν έναν συμπαγή (συνήθως 64 διαστάσεων) και υπολογιστικά αποδοτικό περιγραφέα. Η χρήση της απλοποιημένης μαθηματικής προσέγγισης καθιστά τον SURF ιδανικό για εφαρμογές όπου ο χρόνος επεξεργασίας είναι κρίσιμος.

Ερώτημα 10

2.2 Περιγραφή Χαρακτηριστικών

- **SIFT**:
    - Δημιουργεί **128-διάστατους descriptors** βασισμένους σε τοπικές βαθμίδες (gradient orientations).
    - Χρησιμοποιεί **4×4 πλέγμα κελιών** και **8 ιστογράμματα προσανατολισμών** για κάθε κελί.
- **GLOH**:
    - Παρόμοια με τον SIFT, αλλά με **περισσότερα διανύσματα** (272 αρχικά, μειωμένα σε 128 με PCA).
    - Αποδίδει καλύτερα από τον SIFT σε ανίχνευση αντικειμένων, αλλά είναι πιο ακριβός υπολογιστικά.
- **SURF**:
    - Περιγραφέας **64 διαστάσεων** (ή 128 στην εκτεταμένη εκδοχή του).
    - Βασίζεται σε **Haar wavelet responses** .

Σύγκριση Ανάμεσα στους Αλγορίθμους
SIFT:

- **Υπολογιστικό Κόστος:**
  - Η κατασκευή της Gaussian πυραμίδας και ο υπολογισμός των DoG απαιτούν πολλαπλές convolutions με κόστος περίπου O(k2·M2)για κάθε επίπεδο, ενώ ο πλήρης χρόνος επεξεργασίας μπορεί να κυμαίνεται από 1 έως 2 δευτερόλεπτα για εικόνες με μέγεθος περίπου 800×640 (σε παλαιότερα συστήματα).
- **Σημεία:**
  - Παράγει περιγραφείς 128 διαστάσεων, που οδηγούν σε μεγαλύτερο αριθμό πράξεων κατά την αντιστοίχιση (matching).

GLOH:

- **Υπολογιστικό Κόστος:**
  - Αρχικά παράγει περιγραφείς με 272 διαστάσεις, οι οποίοι στη συνέχεια μειώνονται μέσω PCA σε περίπου 128 διαστάσεων. Η εφαρμογή της PCA προσθέτει επιπλέον υπολογιστικό φόρτο.
- **Σημεία:**
  - Ενώ ο GLOH προσφέρει υψηλότερη διακριτικότητα λόγω του log-polar grid, ο επιπρόσθετος υπολογιστικός φόρτος του οδήγησε σε υψηλότερο κόστος σε σχέση με τον SIFT.

SURF:

- **Υπολογιστικό Κόστος:**
  - Παράγει περιγραφείς 64 διαστάσεων, που καθιστούν την αντιστοίχιση (matching) πολύ ταχύτερη. Σύμφωνα με πειραματικές μελέτες (π.χ. [SIFT and SURF Performance Evaluation against Various Image Deformations on Benchmark Dataset]), ο SURF μπορεί να είναι περίπου 6–8 φορές ταχύτερος από τον SIFT, χωρίς να θυσιάζει σημαντικά την ακρίβεια.

1) Πιθανη πολυπλοκότητα:

Σύνθετη Μαθηματική Διατύπωση (όπως στην αναφορά του LSU thesis)
Σε μία από τις εργασίες που αναφέρεται (η διδακτορική διατριβή του Phaneendra Vinukonda, η οποία μπορείτε να βρείτε μέσω του LSU Digital Commons ή της αντίστοιχης σελίδας), παρουσιάζεται μια σύνθετη έκφραση για το χρόνο εκτέλεσης ενός αλγορίθμου που επεξεργάζεται εικόνα n×n ως εξής για μονοπύρηνα(αντίστοιχος τύπος για πολυπύρηνα αναφέρεται στον συνδεσμο) :

$$[\Theta\left(\frac{(n+x)^2}{p_i}\Gamma_0 + \alpha\beta N^2 x^2 \Gamma_1 + \frac{(\alpha\beta+\gamma)n^2 \log x}{p_o}\Gamma_2\right)]$$

όπου:

- x αναπαριστά το μέγεθος της γειτονιάς (ή του “tile”) που χρησιμοποιείται στην επεξεργασία,
- pi και po είναι οι αριθμοί των “εισόδων” και “εξόδων” (input/output pins) σε έναν επεξεργαστή ή chip,
- α, β και γ είναι παράγοντες που χαρακτηρίζουν τα ποσοστά επεξεργασίας ή το μερίδιο των χαρακτηριστικών στα διάφορα στάδια,
- και Γ0, Γ1, Γ2 αντιπροσωπεύουν τους κύκλους (clock cycles) για την είσοδο, την επεξεργασία και την έξοδο αντίστοιχα.

We develop a tile based template for running SIFT that facilitates the analysis while abstracting away lower-level details. We formalize the computational pipeline and the time to execute any algorithm on it based on the relative times taken by the pipeline stages. In the context of the SIFT algorithm, this reduces the time to that of running the entire image through a bottlenecked stage and the time to run either the first or last tile through the remaining stages. Through an experimental study of the SIFT algorithm on a broad collection of test images, we determined image feature fraction values, that relate the sizes of the image extracts as it the computation proceeds through the stages of the SIFT algorithm.

We show that for a single chip uniprocessor pipeline, the computational stage is the bottleneck. Specifically we show that for an $N \times N$ image with $n \times n$ tiles the overall time complexity is $\Theta\left(\frac{(n+x)^2}{p_i}\Gamma_0 + \alpha\beta N^2 x^2 \Gamma_1 + \frac{(\alpha\beta+\gamma)\, n^2 \log x}{p_o}\Gamma_2\right)$; here x is the neigborhood of the tile, p_i, p_o are the number of input, output pins of the chip, α, β, γ are the feature fractions, and $\Gamma_0, \Gamma_1, \Gamma_2$ are the input, compute, output clocks. The three terms in the expression represents the time complexities of input, compute and output stages. The input and output stages can be slowed down substantially without appreciate degradation of the overall performance. This slowdown can be traded off for lower power and higher signal quantity.

Επίσης αναφέρει:

Table 2.1: The time complexity and the number of operations required by the different phases of the SIFT algorithm for N^2 pixels

Phase	Complexity	Number of operations
Gaussian Blurring	$\Theta\left(N^2 w^2 s\right)$	$4N^2w^2s$
Difference of Gaussian	$\Theta\left(sN^2\right)$	$4N^2s$
Scale-space Extrema Detection	$\Theta\left(sN^2\right)$	$104sN^2$
Keypoint Detection	$\Theta\left(\alpha s N^2\right)$	$100s\alpha N^2$
Orientation Assignment	$\Theta\left(sN^2(1-\alpha\beta)\right)$	$48sN^2$
Keypoint Descriptor Generation	$\Theta\left(x^2N^2(\alpha\beta+\gamma)\right)$	$1520x^2(\alpha\beta+\gamma)N^2$

For multicore chips, we show that for an $N \times N$ image on a P-core chip, the overall time complexity to process the image is $\Theta\left(\frac{N^2}{p_i}\Gamma_0 + \frac{(n^2w^2 + \alpha\beta n^2 x^2)}{P}\Gamma_1 + \frac{(\alpha\beta+\gamma)\, n^2 \log x}{p_o}\Gamma_2\right)$; in addition to the quantities described earlier w is the window size used for the Gaussian blurring. Overall we establish that without improvements in the input bandwidth, the power of multicore processing cannot be used efficiently for SIFT.

2. Χρησιμοποιηθεικε επίσης για το παρακάτω ερώτημα η εξης τόσο για τις διαφορές οσο και για τα κόστη (είναι και στις πηγές): SIFT, SURF GLOH descriptors https://www.micc.unifi.it/delbimbo/wp-content/uploads/2011/03/slide_corso/A33%20SIFT-GLOH-SURF.pdf#:~:text=Computation%20of%20SIFT%20descriptor%20%E2%80%A2,keypoint

Υπολογιστικό Κόστος

Περιγραφέας	**Ενδεικτικός χρόνος(για εικόνα ~800×640)**	**Διάσταση descriptor**	**Σχόλια / Παρατηρήσεις**
SIFT	~1000–1100 ms(π.χ. 1036 ms στον πίνακα)	128 στοιχεία	- Πολύ διαδεδομένος, υψηλή διακριτική ικανότητα.- Σχετικά αργός σε σχέση με SURF.- Αρχικά (~6s) σε μεγάλες εικόνες (1280×768).
GLOH	Δεν δίνεται ακριβές νούμερο στον πίνακα,αλλά γενικά υψηλότερο από SIFT	272 στοιχεία (πριν την PCA)συνήθως μειώνονται με PCA	- Παρόμοιος με SIFT αλλά με πολυπλοκότερη δομή (log-polar grid).- Η υψηλή διάσταση (272) οδηγεί σε αυξημένο υπολογιστικό κόστος, αν και συχνά γίνεται μείωση διάστασης με PCA.
SURF	350–400 ms(π.χ. 354 ms ή 391 msγια SURF-128)	64 στοιχεία (βασική έκδοση)128 στοιχεία (SURF-128)	- Πολύ πιο γρήγορος από SIFT, χάρη στη χρήση Integral Images και box filters για τις παραγώγους (Fast Hessian).- Σε “U-SURF” πέφτει ακόμη περισσότερο ο χρόνος (~255 ms).

Computational cost

- SURF computational cost (detector and descriptor) against the most-used detectors and the SIFT descriptor:

Table 1. Thresholds, number of detected points and calculation time for the detectors in our comparison. (First image of Graffiti scene, 800 × 640).

detector	threshold	nb of points	comp. time (msec)
Fast-Hessian	600	1418	120
Hessian-Laplace	1000	1979	650
Harris-Laplace	2500	1664	1800
DoG	default	1520	400

Table 2. Computation times for the joint detector - descriptor implementations, tested on the first image of the Graffiti sequence. The thresholds are adapted in order to detect the same number of interest points for all methods. These relative speeds are also representative for other images.

	U-SURF	SURF	SURF-128	SIFT
time (ms):	255	354	391	1036

Εξτρα Συνοπτική Σύγκριση SIFT – GLOH – SURF

- **SIFT**
  - Χρησιμοποιεί Gaussian πυραμίδες και Difference of Gaussians (DoG) για πολλαπλές κλίμακες.
  - Ο εντοπισμός των ενδιαφερόντων σημείων και ο υπολογισμός του descriptor (128 διαστάσεις) είναι σχετικά αργός (περίπου 1s σε εικόνες 800×640, έως ~6s σε 1280×768).
- **GLOH**
  - «Παραλλαγή» του SIFT με log-polar πλέγμα, αρχική διάσταση descriptor 272 (συνήθως μειώνεται με PCA).
  - Λόγω υψηλότερης διάστασης είναι ακριβότερο υπολογιστικά (και στο matching) από τον κλασικό SIFT.
- **SURF**
  - Εντοπισμός σημείων βάσει Hessian και χρήση Integral Images + box filters αντί Gaussians, για μεγάλη επιτάχυνση.
  - Ο descriptor έχει 64 διαστάσεις (βασική έκδοση) ή 128 (SURF-128).
  - Υπολογιστικά 2–3 φορές ταχύτερος από SIFT (~350–400ms), ενώ στην έκδοση U-SURF (χωρίς ανίχνευση προσανατολισμού) φτάνει ~255ms.

4. Συμπεράσματα

- **SIFT**: Πολύ ακριβής, αλλά αργός. Χρήσιμος για υψηλή αξιοπιστία αντιστοιχίσεων.
- **GLOH**: Καλύτερη διακριτότητα από τον SIFT, αλλά ακόμη πιο αργός.
- **SURF**: Καλός συμβιβασμός μεταξύ ταχύτητας και ακρίβειας. Ιδανικός για πραγματικό χρόνο.

Ο **SURF είναι η πιο αποδοτική επιλογή**, αλλά αν η ακρίβεια είναι κρίσιμη, ο **SIFT/GLOH είναι καλύτεροι**.
Η τελική επιλογή εξαρτάται από την εφαρμογή και την ισορροπία μεταξύ **ακρίβειας και ταχύτητας**.

<u>3.Επιπλέον επιβεβαίωση ταχύτητας μεσω IEEE Paper για SIFT και SURF</u>:
Σύμφωνα με την εργασία "SIFT and SURF Performance Evaluation against Various Image Deformations on Benchmark Dataset", οι μετρήσεις επιβεβαιώνουν πως ο SURF έχει σημαντικά χαμηλότερο χρόνο εκτέλεσης σε σύγκριση με τον SIFT, ειδικά σε συνθήκες μετασχηματισμών (π.χ. περιστροφή, κλίμακα). Επιπλέον, αναδεικνύεται ότι το χαμηλότερο dimensional descriptor του SURF (64 διαστάσεων) συνεπάγεται ταχύτερη αντιστοίχιση, κάτι που είναι κρίσιμο για εφαρμογές σε πραγματικό χρόνο.

ΒΙΒΛΙΟΓΡΑΦΙΑ:

- Lowe, D. G. (2004). **Distinctive image features from scale-invariant keypoints**.: https://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94

- Image-Based Alignment of 3D Scans: https://www.researchgate.net/publication/354597620_Image-Based_Alignment_of_3D_Scans

- Image Matching for Geomorphic Measurement Based on SIFT and RANSAC Methods: https://ieeexplore.ieee.org/abstract/document/4722061?casa_token=VvB2rYEUVAMAAAAA:7jC8JctWgDNlefS8QThhiLtrvOxZqDPbw7HRp_6LCUVePL9Q_tNm8dXBdq_A4NXbirIlywrsvCA

- Bay, H., Tuytelaars, T., & Van Gool, L. (2006). **SURF: Speeded Up Robust Features**.: https://link.springer.com/chapter/10.1007/11744023_32

- SIFT and SURF Performance Evaluation against Various Image Deformations on Benchmark Dataset : https://ieeexplore.ieee.org/document/6128710

- A study of the scale-inv A study of the scale-invariant f ariant feature transform on a par ansform on a parallel pipeline Phaneendra Vinukonda https://repository.lsu.edu/cgi/viewcontent.cgi?article=3720&context=gradschool_theses

- SIFT, SURF GLOH descriptors: https://www.micc.unifi.it/delbimbo/wp-content/uploads/2011/03/slide_corso/A33%20SIFT-GLOH-SURF.pdf#:~:text=Computation%20of%20SIFT%20descriptor%20%E2%80%A2,keypoint

- https://en.wikipedia.org/wiki/Scale-invariant_feature_transform
- https://en.wikipedia.org/wiki/Random_sample_consensus
- mathworks της matlab για τον εναλλακτηκό τρόπο υλοποίησης της Ransac: https://www.mathworks.com/help/vision/ref/estimategeometrictransform2d.html