



Exercise 6

(due June 12th 2008, 23:59)

Question 1: Transformation

Implement an affine transformation of a geometric figure.

- a) (3 points) Write a function `affine` which builds a 3x3 affine matrix given translations in x,y, rotation `angle1`, rotation `angle2`, scale change `s1`, scale change `s2` (See slides for Local Features and Interest Points I). Convert the angles to radians before computing sin and cos ($angle_{rad} = angle^{\circ} \cdot \pi/180$). The function should display the matrices `R(angle1)`, `S`, and `R(angle2)` and output affine matrix.

```
function A = affine(x,y,angle1,angle2,s1,s2)
...
...
end
```

Generate a matrix for $x = 0, y = 0, angle1 = 10, angle2 = 60, s1 = 5, s2 = 2$.

Apply Singular Value Decomposition to the matrix:

Call `[u d v]=svd(A(1:2,1:2))`.

Compare `u`, `d` and `v` with `R(angle1)`, `S`, and `R(angle2)`.

Change the parameters and compare again.

- b) (3 points) Write a function `transform` which takes as an input coordinates of a figure (list of `px1` and `py1`) and affine parameters. The function will output the transformed points.

```
function [px2 py2] = transform(px1,py1,x,y,angle1,angle2,s1,s2)
...
...
end
```

For example square figure can be defined:

```
px1=[-10 10 10 -10 -10];
py1=[-10 -10 10 10 -10];
plot(px1,py1);axis([-20 20 -20 20]);
```

In the function `transform`, first build an array of points in homogeneous coordinates. The first row will contain coordinates `px`, the second coordinates `py` and the third row ones:

```
p1=[px1;py1;ones(1,number_of_values_in_px)]
```

Transformation of points can be simply done by

$$p2 = A \cdot p1$$

Display the transformed figure.

- c) Play with the parameters. Start by varying the translation by i.e., $x=10, y=15$ pixels, $angles=0$, $scales=1$, and display the figure. Then keep the translation 0,0 and vary the angles, and so on with the scales such that you see how each parameter changes the transformation.

Question 2: DoG Detector

- a) (6 points) Implement scale invariant Difference of Gaussian Detector.

```
function [px py ps]=dog_detector(img,thres)
...
...
end
```

Implement a function which computes scale space representation of an image.

1. define initial $\sigma = \sqrt{2}$
2. run `gaussianfilter.m` on the image for n , where n is 1, 2, 3, 4, ..., 11 and collect the blurred images in array `ss`:

```
ss=zeros(size(img,1),size(img,2),11);\\
ss(:,:,1)=gaussianfilter(img,sigma);\\
ss(:,:,2)=gaussianfilter(img,sigma^2);\\
:\\
ss(:,:,11)=gaussianfilter(img,sigma^11;
```

3. Compute a difference between all the neighboring levels.

```
dog=zeros(size(img,1),size(img,2),10);
dog(:,:,1)=ss(:,:,1)-ss(:,:,2);
dog(:,:,2)=ss(:,:,2)-ss(:,:,3);
:
dog(:,:,10)=ss(:,:,10)-ss(:,:,11);
```

Write a function which detects local maxima in 3D space of `dog`. Take an approach similar to `nonmaxsup2d` in the Exercise 4 or in the slides of Interest Points II: Scale Invariance).

- Start with detecting local maxima at the second level $n = 2$.
- Increment point position (i, j) . 3. If the value at `dog(i, j, n = 2)` is larger than any neighboring pixel $(i - 1, i, i + 1, j - 1, j, j + 1)$, than go to step 3, otherwise go to step 2.
- Verify that the value `dog(i, j, n)` is also larger than corresponding values at lower `dog(i, j, n - 1)` and higher `dog(i, j, n + 1)` scale level.
- If `dog(i, j, n)` is the local maximum higher than `thres` than remember the position (i, j) and scale n of the point in an list `[px py ps]`. Go to step 2
- Repeat the detection of local maximum at each level (`dog(i, j, 4), dog(i, j, 4), ..., dog(i, j, 7)`).

- b) Write a function `drawpoints` which takes an input image and lists `px, py, ps` of interest point coordinates and displays the regions in the image

```
function D = drawpoints3( img, px, py, ps )
[h w] = size(img);
imagesc(img);colormap gray;hold on;
for i=1:length(px)
    rad = round(4*(ps(i)+1);
    minx = max(px(i)-rad,1);
    maxx = min(px(i)+rad,w);
    miny = max(py(i)-rad,1);
    maxy = min(py(i)+rad,h);
    plot([minx maxx maxx minx minx ],[miny miny maxy maxy miny]);
end;
hold off;
```

- c) Detect and display interest points in image `boat1.png, boat2.png, boat3.png, boat4.png`, compare the displayed features.
- d) (2 points) Modify the function `descriptors_maglap` which computes a *mag, lap* histogram around each interest point so that size of the image patch is adapted to the scale of interest point (Note: experiment a little bit with an example image to select the histogram range appropriately). The size of the patch will be $(4ps(i) + 1)$

```
function D = descriptors_maglap(img, px, py, ps, sigma, bins)
...
...
end
```

Question 3: Matching

- a) (3 points) Find the corresponding DoG interest points between image pairs `boat1.png/boat2.png` and `boat1.png/boat4.png` using the scale-adaptive version of `descriptors_maglap`. Visualize the matches using function `displaymatches` provided in the solution to the exercise sheet 5.
- b) (3 points) Go to <http://lear.inrialpes.fr/people/Mikolajczyk/ScaleSelection/index.html>. Follow the instructions on the web page and experiment with different criteria for selection of scale at interest points. Write a short summary of your observations.

Please turn in your solution by sending an email to Micha Andriluka (andriluka@mis.informatik.tu-darmstadt.de) including all relevant m-files before Thursday, June 12th, 23:59