Image Analysis And Computer Vision
Exercises

Christian Rossi

Academic Year 2023-2024

Abstract

The topics of the course are:

- Introduction.
- Camera sensors: transduction, optics, geometry, distortion
- Basics on Projective geometry: modelling basic primitives (points, lines, planes, conic sections, quadric surfaces) and projective spatial transformations and projections.
- Camera geometry, and single view analysis: calibration, image rectification, localization of 3D models.
- Multi-view analysis: 3D shape reconstruction, self-calibration, 3D scene understanding.
- Linear filters and convolutions, space-invariant filters, Fourier Transform, sampling and aliasing.
- Nonlinear filters: image morphology and morphology operators (dilate, erode, open, close), median filters.
- Edge detection and feature detection techniques. Feature matching and feature tracking along image sequences.
- Inferring parametric models from noisy data (including outliers), contour segmentation, clustering, Hough Transform, Ransac (random sample consensus).
- Applications: object tracking, object recognition, classification.

Contents

Chapter 1

Introduction to MATLAB

1.1 Main MATLAB operators

To print some string it is possible to use;

```
% Print the string
disp('string');
% Print the string with C-like syntax
fprintf('string\n%s %d','string',number);
```

The variables are created as follows:

```
% Variables are created by assignements
v=3
c='k'
% Data types can be checked
whos v
% Casting to 8-bit integer
v=uint8(v)
```

The main data types used on MATLAB: double, uint8, and logical. The arrays are defined in the following ways:

```
% A row vector
r=[1, 2, 3, 4]
% A column vector
c=[1; 2; 3; 4]
% Vectors by regular increment operator
% [start:step:end]
a = [1:2:10];
% A matrix
```

```
v=[ 1 2;
    3 4 ]
```

It is possible to concatenate arrays:

```
B=[v', v']
C=[v; v]
```

The arrays can be divided in subarrays:

```
% First row and second column
v(1,2)
% The second column of v
v(:,2)
% The first row of v
v(1,:)
% Some of the columns from 2 to 4
B(:,2:4)
```

Some useful mathematical operations are:

```
% . means elementwise operation
[1 2 3].*[4 5 6]
[1 2 3]+5
[1 2 3]*2
[1 2 3].*2
[1 2 3]/2
[1 2 3]./2
[1 2 3].^2
% Inner product
[1 2 3]*[4 5 6]'
% This is the matrix product, returns a matrix
[1 2 3]'*[4 5 6]
% Functions for rounding functions
ceil(10.56)
floor(10.56)
round(10.56)
% Arithmetic functions
sum([1 2 3 4])
sum([1:4;5:8])
sum([1:4;5:8],2)
```

1.2 Commands for images

The images in MATLAB are treated as matrices:

```
im=imread('photo.png');
% Show the image
imshow(im);
% Show two concatenated images orizontally
imshow([im im]);
% Show two concatenated images vertically
imshow([im; im]);
```

To plot the histogram of the various pixels it is possible to write:

```
h=hist(im(:),[0: im_length]);
figure(2), stairs([0: im_length], h)
axis tight
```

It is possible to modify the brightness and contrast with a simple operation:

```
figure(1),imshow(im+50),title('50 graylevels')
figure(1),imshow(im+100),title('100 graylevels')
% contrast modify
eq=double(im-min(im(:)))/double(max(im(:))-min(im(:)
   ))*255;
```

Image ranges (in the visualization) can be also controlled:

```
imshow(im,[-100 156]);title('more brighness');
imshow(im,[0 156]);title('more brighness and
   contrast');
imshow(im,[ 100 256]);title('less brighness, more
   contrast');
imshow(im,[ 100 356]);title('less brighness');
```

The gamma correction is done in this way:

```
for gamma= [.04 .1 .2 .4 .7 1 1.5 2.5 5 10 25]
    y=x.^gamma;
    plot(x,y,'DisplayName',sprintf('\\gamma=%.2f',
       gamma));
    % display the text
    text(x(round(end/2)),y(round(end/2)),sprintf('\\
       gamma=%.2f',gamma));
end
```

Color is represented by 3 channels (RGB). We can read each channel:

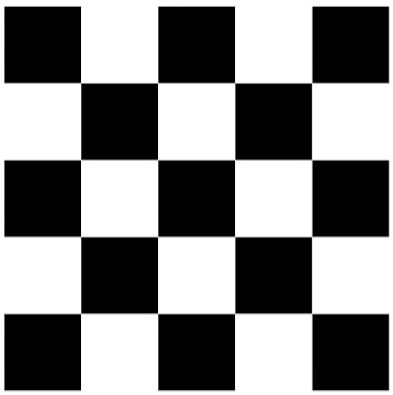
```
% Red channel
imr=im(:,:,1);
% Green channel
imr=im(:,:,2);
% Blue channel
imr=im(:,:,3);
```

Chapter 2

Laboratory session I

Exercise 1

Elaborate the following image called "checkerboard.png":



Answer of exercise 1

Initially, we have to load the image in a matrix:

```
I=imread('checkerboard.png');
FNT_SZ=28;
```

We now want to manually select four points and save them in homogeneous coordinates:

```
figure(1), imshow(I);
hold on;
[x,y]=getpts();
```

```
a=[x(1);y(1);1];
b=[x(2);y(2);1];
c=[x(3);y(3);1];
d=[x(4);y(4);1];
```

And now we can show the points label on the picture with:

```
text(a(1),a(2),'a','FontSize',FNT_SZ,'Color','b')
text(b(1),b(2),'b','FontSize',FNT_SZ,'Color','b')
text(c(1),c(2),'c','FontSize',FNT_SZ,'Color','b')
text(d(1),d(2),'d','FontSize',FNT_SZ,'Color','b')
```

We now define the lines passing through these points:

```
lab=cross(a,b);
lad=cross(a,d);
lac=cross(a,c);
lcd=cross(c,d);
```

Intersect these lines with the image borders

```
r1=[0;1;-1];        % Parameters of top-most row
r500=[0;1;-500];    % Parameters of bottom row
c1=[1;0;-1];        % Parameters of left-most column
c500=[1;0;-500];    % Parameters of right-most column
% Intersection between lab and the first column
x1=cross(c1,lab);
x1=x1/x1(3);
text(x1(1),x1(2),'x1','FontSize',FNT_SZ,'Color','b')
% Same with the right most column
x500=cross(c500,lab);
x500=x500/x500(3);
text(x500(1),x500(2),'x500','FontSize',FNT_SZ,'Color
   ','b')
plot([x1(1),x500(1)],[x1(2),x500(2)],'LineWidth',3)
```

Compute the angles on these images:

```
computeEuclideanAngles(lab,lac)
computeEuclideanAngles(lab,lcd)
computeEuclideanAngles(lab,lad)
```

Verify the proporty any linear combination of a, b belongs to lab:

```
lambda=rand(1);
```

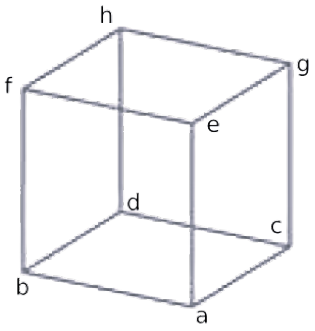
```
mu=1-lambda;
p= lambda*a+mu*b;
p'*lab
p=p/p(3);
text(p(1),p(2),'p','FontSize',FNT_SZ,'Color','b')
```

Intersect parallel lines

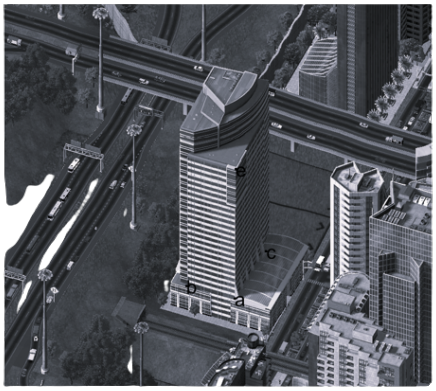
```
vac=cross(lab,lcd)
vac=vac/vac(3)
vab=cross(r1,r500)
vad=cross(c1,c500)
```

Exercise 2

Elaborate the following image called "simplecube-letters.png":



And the following image called "buildingSmall.png":



Answer of exercise 2

Initially, we have to load the image in a matrix:

```
figure(1),imshow(imread('simplecube-letters.png'));
```

Data entering:

```
figure(2),imshow(imread('buildingSmall.png'))
hold on;
[x y]=getpts
plot(x,y,'.w','MarkerSize',12,'LineWidth',3);
a=[x(1) y(1) 1]';
```

```
text(a(1),a(2),'a','FontSize',FNT_SZ,'Color','w')
b=[x(2) y(2) 1]';
text(b(1),b(2),'b','FontSize',FNT_SZ,'Color','w')
c=[x(3) y(3) 1]';
text(c(1),c(2),'c','FontSize',FNT_SZ,'Color','w')
e=[x(4) y(4) 1]';
text(e(1),e(2),'e','FontSize',FNT_SZ,'Color','w')
```

Finding some lines:

```
lab=cross(a,b)
lac=cross(a,c)
lae=cross(a,e)
```

The incidence equation is staisfied if one of the following operations are null:

```
lab'*a
lab'*b
lac'*a
lac'*c
lae'*a
lae'*e
```

We now need to compute the line parallel to ab and passing through c. In order to do this, we create the line at infinity. Then, we find the directions of segments by ab, ac and ae intersecting their lines with the line at infinity.

```
linf=[0 0 1]';
dab=cross(lab,linf)
dac=cross(lac,linf)
dae=cross(lae,linf)
```

dab, dac and dae are points at the infinity, which represent directions. All lines with a given direction pass through the corresponding point at the infinity. We can then find the lines containing segments bd and cd.

```
lbd=cross(b,dac);
lcd=cross(c,dab);
```

point d is now found by just intersecting lbd and lcd

```
d=cross(lbd,lcd);
```

We normalize d's coordinates so that we can read its cartesian coordinates in d(1) and d(2). We plot the point with a blue circle.

```
d=d/d(3);
plot(d(1),d(2),'.b','MarkerSize',12);
text(d(1),d(2),'d','FontSize',FNT_SZ,'Color','w')
```

The rest of the procedure is straightforward, and follows exactly the same technique.

```
lbf=cross(b,dae);
lcg=cross(c,dae);
ldh=cross(d,dae);
lef=cross(e,dab);
leg=cross(e,dac);
f=cross(lbf,lef);
g=cross(lcg,leg);
lfh=cross(f,dac);
h=cross(lfh,ldh);
f=f/f(3);
g=g/g(3);
h=h/h(3);
plot(f(1),f(2),'.w','MarkerSize',12,'LineWidth',3);
text(f(1),f(2),'f','FontSize',FNT_SZ,'Color','w')
plot(g(1),g(2),'.w','MarkerSize',12,'LineWidth',3);
text(g(1),g(2),'g','FontSize',FNT_SZ,'Color','w')
plot(h(1),h(2),'.w','MarkerSize',12, 'LineWidth',3);
text(h(1),h(2),'h','FontSize',FNT_SZ,'Color','w')
```

We can now finally draw the cube:

```
myline=[a';b';d';c';a'];
line(myline(:,1),myline(:,2),'LineWidth',5);
myline=[e';f';h';g';e'];
line(myline(:,1),myline(:,2),'LineWidth',5);
myline=[a';e'];
line(myline(:,1),myline(:,2),'LineWidth',5);
myline=[b';f'];
line(myline(:,1),myline(:,2),'LineWidth',5);
myline=[c';g'];
line(myline(:,1),myline(:,2),'LineWidth',5);
myline=[d';h'];
line(myline(:,1),myline(:,2),'LineWidth',5);
hold off
```