Load one of the images into matlab and display it using the matlab command `imagesc`.

```
bild = readpgm('plc001.pgm');
figure(1);
colormap(gray(64));
imagesc(bild);
zoom on;
```

Try zooming in on different parts of the image. Do you see anything special? Try zooming in on the sharp edges.

The errors are introduces in the frame grabber. This is quite typical. Each frame grabber or camera has its on quirks. We have implemented a special routine that removes these problems (for this specific frame grabber). Try

```
bild = pgmlas('plc001.pgm',1);
figure(1);
colormap(gray(64));
imagesc(bild);
zoom on;
```

Image points

A point in the image can be represented by its Cartesian coordinates (x, y). Most often, however, we will use so called extended or homogeneous coordinates

$$\mathbf{x} \sim \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad ,$$

where $\sim$ denotes equality up to scale. What Cartesian coordinates do the following points have:

$$\begin{pmatrix} 100 \\ 250 \\ 1 \end{pmatrix}, \qquad \begin{pmatrix} 100 \\ 250 \\ 2 \end{pmatrix}, \qquad \begin{pmatrix} 100 \\ 231 \\ -1 \end{pmatrix}$$

Load one of the images into matlab and display it using the matlab command `imagesc`. Then plot a point using a home made command `rita`.

```
bild = pgmlas('plc001.pgm',1);
figure(1);
colormap(gray(64));
imagesc(bild);
zoom on;
hold on;
u1=[229.639 251.5886 1]';
rita(u1,'*');
```

Where are the coordinates (230,251) located? In the upper left corner of a pixel or in the middle of a pixel? What interpolation method does matlab use when displaying images?

Image lines

An image line can be written in affine form as

$$\{(x,y)\,|\,ax+by+c=0\}$$

i.e. all points that fulfill the constraint $ax+by+c=0$. We will usually use the representation

```
l=[a;b;c];
```

for this line. Notice that an image point u lies on the line l if and only if $l^T u = 0$. Notice also that two lines are identical if they differ by scale only. The notation $\sim$ is used to denote equality up to scale.

$$\begin{pmatrix}1\\2\\4\end{pmatrix} \sim \begin{pmatrix}5\\10\\20\end{pmatrix}$$

Try

```
l=[ 0.00307478677193; -0.00678159650595; 0.99997227743330 ];
rital(l);
```

image conics

Similarly image conics can be written as

$$\{(x,y)\,|\,C_{11}x^2+2C_{12}xy+2C_{13}x+C_{22}y^2+2C_{23}y+C_{33}=0\}$$

i.e. all points that fulfill the constraint. We will usually use the representation

$$C=\begin{pmatrix}C_{11} & C_{12} & C_{13}\\ C_{12} & C_{22} & C_{23}\\ C_{13} & C_{23} & C_{33}\end{pmatrix} \tag{1}$$

for this conic. Sometimes it is convenient to represent the conic using all tangent lines to the conic, i.e. the conic is defined as the conic which contains all the following tangent lines

$$\{(a,b,c)\,|\,D_{11}a^2+2D_{12}ab+2D_{13}ac+D_{22}b^2+2D_{23}bc+D_{33}c^2=0\}$$

The two matrices D and C are inverses of each other $D=C^{-1}$.

Try

```
C=[ ...
  -0.00000459720004  -0.00000142168902   0.00265719581592; ...
  -0.00000142168902  -0.00000208695435   0.00112867557884; ...
   0.00265719581592   0.00112867557884  -1.59016798885548; ...
];
ritac(C);
```

The camera equation

Points in a world coordinate system is projected onto points in the image using the camera equation.

$$x \sim PX$$

Since we only get the image points up to scale it is necessary to normalize the third coordinate to one, before plotting them. This can be done with the command `pflat`. Try

```
X = [randn(3,20);ones(1,20)];
P = [eye(3) [0;0;10]];
x = pflat(P*X);
figure(2);
rita(x,'*');
axis([-0.3 0.3 -0.3 0.3]);
```

This gives the projection of 20 points from one viewpoint. The next script generates a sequence of camera matrices Pt, using a local parametrisation of rotations:

$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \mapsto e^{\begin{pmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{pmatrix}})$$

Try

```
R=eye(3);
r=0.05*randn(3,1);
for i=1:100;
 R=expm([0 -r(3) r(2);r(3) 0 -r(1);-r(2) r(1) 0])*R;
 T=[R zeros(3,1);zeros(1,3) 1];
 Pt = P*T;
 xt = pflat(Pt*X);
 rita(xt,'*');
 axis([-0.3 0.3 -0.3 0.3]);
 drawnow;
 pause(0.1);
end;
```

Consider two lines in the scene. The first line goes through points $(-0.5, -1, 0)$ and $(-0.5, -1, 100)$. The second line goes through points $(0.5, -1, 0)$ and $(0.5, -1, 100)$. Do the lines lie in a common plane? Do they intersect?

In matlab plot the projection of these four points with the camera matrix P above.

```
Xl11 = [-0.5 -1 0 1]';
Xl12 = [-0.5 -1 100 1]';
Xl21 = [0.5 -1 0 1]';
Xl22 = [0.5 -1 100 1]';
xl11 = pflat(P*Xl11);
xl12 = pflat(P*Xl12);
xl21 = pflat(P*Xl21);
xl22 = pflat(P*Xl22);
hold off;
rita(x,'*');
hold on
rita([xl11 xl12]);
rita([xl21 xl22]);
```

Calculate the intersection of the two object lines. What are the homogeneous coordinates? What are the Cartesian coordinates? Calculate the projection of the intersection and plot it in the image.

```
Xint = [0 0 1 0]';
xint = pflat(P*Xint);
rita(xint,'*');
```

Notice that the intersection in world space is a strange point (a point at infinity), but the projection is an ordinary point.

Additional exercises

When you have time, try to solve the exercises in chapter 1 and 2.

Log out

Erase *all* big files that you have created or copied. Exit all applications and log out.