CS6320: 3D Computer Vision
Project 2
Stereo and 3D Reconstruction from Disparity

Arthur Coste: *coste.arthur@gmail.com*

February 2013

Contents

1 Introduction

In this second project we are going to extend the study of camera we started in the previous project by adding one or more camera to our analysis to discuss stereo vision and epipolar geometry. This will allow us to discuss depth determination which was an issue we already discussed in the previous project, because due to the perspective projection equations we showed that it was impossible to reconstruct depth from a single picture. Here we are going to show that it becomes possible if we use two cameras which is the basis of the human vision system.

In this report, we will first discuss some theoretical problems related to the geometry with several cameras and depth reconstruction and in a second part we will work on practical problems and present their implementation and resolution.

In this report, the implementation is made using MATLAB, the functions we developed are included with this report and their implementation presented in this document.

The following functions are associated with this work :

- $select\_points.m$: $[x, y] = select\_points(InputImage)$
- $epipolar\_geometry.m$: $[] = epipolar\_geometry()$ Stand alone function
- $distance\_map.m$: $[] = distance\_map()$ Stand alone function

Important : the distance map program requires long computation time (up to a dozen minutes according to the parameters)

Note : All the images and drawings were realized for this project so there is no Copyright infringement in this work.

2 Theoretical Problems

In this section we are going to present and explain theoretical aspects regarding the geometry at stake when using multiple cameras to reconstruct depth.

2.1 Epipolar Geometry with 3 Cameras

In this section we are going to discuss the special case of three camera geometry. Here is an illustration of the set up of the system.

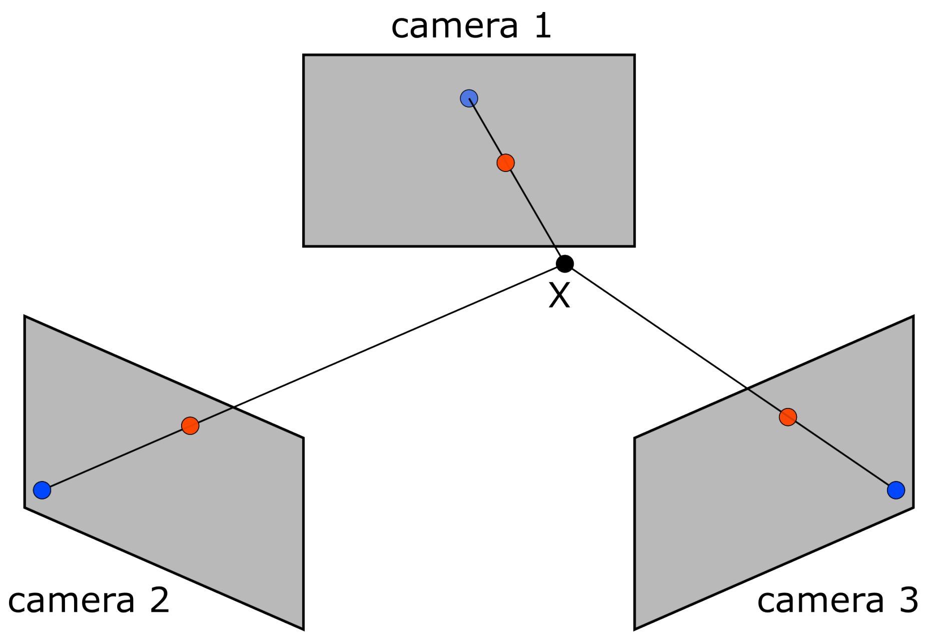

Figure 1: Geometry set up with three cameras

According to the previous set up, we have a point X and 3 cameras acquiring an image. So due to the projective equation system at stake in each camera we obtain an projection of this point on each image plane. We are now going to use the same kind of reasoning we did with the two camera model but applied to our set up with 3 cameras.

Firstly let's consider the epipoles. Indeed, as we said in the two camera geometry, an epipole is the intersection of the line linking the projection center of each camera with each image plane. In the stereo framework with two cameras we had two epipoles because we had two image plane and only one line to connect the two projection center.
In the previous framework with three camera, we have 3 lines that connect the projection center of each camera. So among those three lines, two of them are going to intersect each image plane because of the triangular geometry induced by the three cameras. So we end up with 6 epipoles which are represented in green on the following picture.

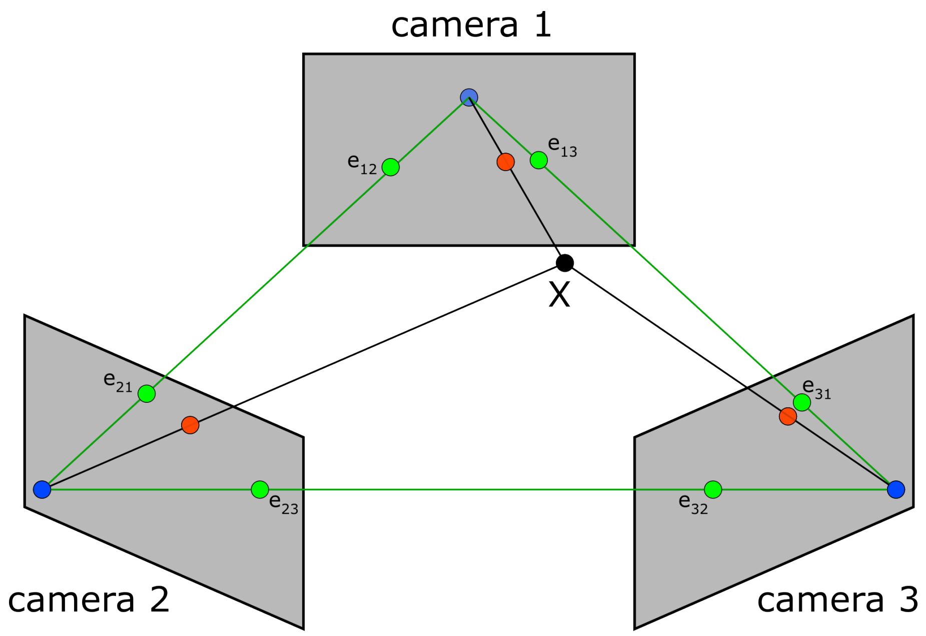

Figure 2: The 6 epipoles and three lines associated with our system

The 3 optical center of each camera or projection center belong to a single plane which is called the trifocal plane. Let's now show that thanks to this set-up we can theoretically have a point wise relationship. In fact, in the standard stereo model, we have a dimensionality reduction, because we know that the location of the point is not only contained in the image plane but also on the epipolar line created by the projection of the first camera on the second one.
In this case, we are going to reduce again the dimensionality of our match to a single point resulting from the intersection of two epipolar lines coming from the projection of the other two cameras.

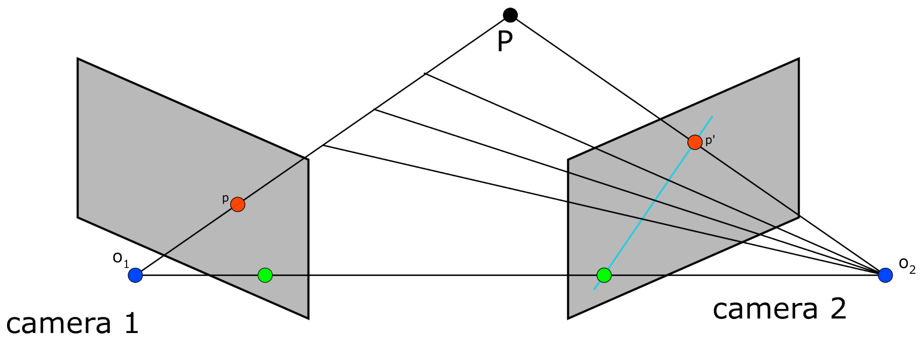

Figure 3: Epipolar geometry in the case of two camera

As we explained before, in this case we reduce our search of the matching point to the epipolar line. If we generalize our reasoning to the tripolar set up and assuming we know the projection of the point on two of the three cameras, we can then using epipolar constraint coming from the two camera obtain two epipolar line that are crossing each other on the projection of the point on the third camera.

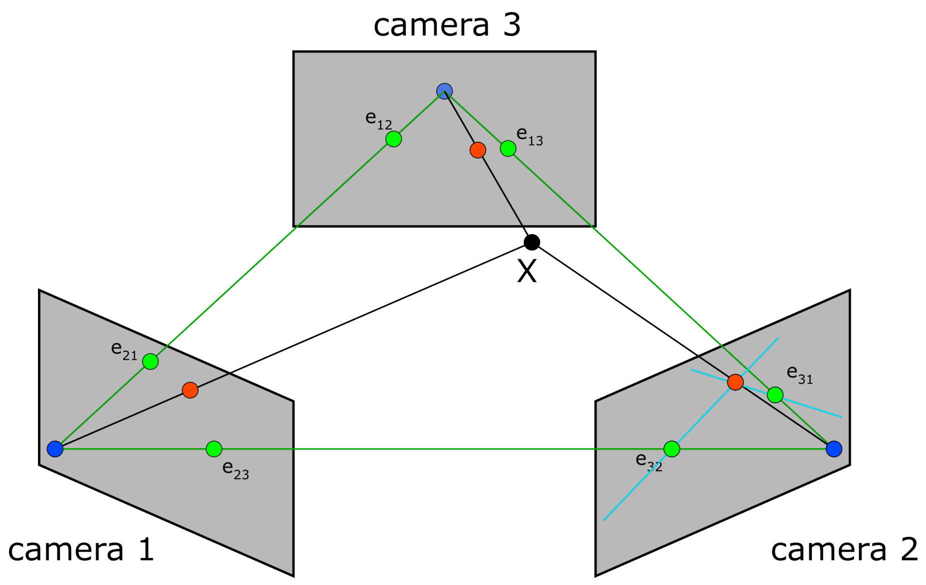

Figure 4: Projection of the point on camera 1 and 3 creates 2 epipolar lines that cross on camera 2

To conclude, if we assume that the geometry is known, thanks to the epipolar constraint associated to two cameras we can then determine the location of the point on the third camera at the intersection of the two epipolar lines coming from the two other camera.

An other way to see this aspect would be to think about planes crossing each other which result in a line for two planes and a point for three planes.

2.2 Epipolar Geometry and disparity forward translating camera

In class we discussed the case of horizontal translation of a camera to acquire a stereo pair of images but we did not discuss a lot the case of forward translation. This is why in this section we are going to present it more. Here is a illustration of the set up of the system.

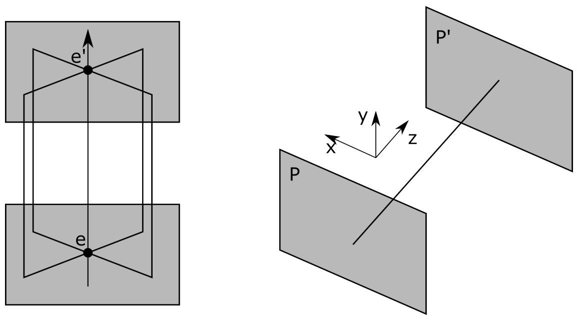

Figure 5: Forward translation of camera

Firstly, let's show that in such a framework of forward/backward translating camera results in a radially oriented set of planes. To do so we are going to do the same kind of studies we did for horizontal translation.

So we have our epipolar equation involving the essential matrix:

$$p^T \epsilon p = 0 \qquad \text{with} \qquad \epsilon = [t_z]R \qquad \text{and} \qquad R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

Due to the fact that we only consider a translation according the z axis, we can simplify the initial cross product matrix:

$$\begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \qquad \Rightarrow \qquad \begin{pmatrix} 0 & -t_z & 0 \\ t_z & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2}$$

This allow us to determine the expression of our essential matrix :

$$\epsilon = \begin{pmatrix} 0 & -t_z & 0 \\ t_z & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{3}$$

And we can finally apply it to two points $p = (u, v, 1)$ and $p' = (u', v', 1)$:

$$p^T \epsilon p' = (u, v, 1) \begin{pmatrix} 0 & -t_z & 0 \\ t_z & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \tag{4}$$

$$p^T \epsilon p' = (u, v, 1) \begin{pmatrix} -t_z v' \\ t_z u' \\ 0 \end{pmatrix} \tag{5}$$

$$p^T \epsilon p' = -u t_z v' + v t_z u' \tag{6}$$

Using the epipolar constraint : $p^T \epsilon p' = 0$ we finally obtain :

$$p^T \epsilon p' = 0 \qquad \Rightarrow \qquad -ut_z v' + vt_z u' = 0 \qquad \Rightarrow \qquad vu' = uv' \tag{7}$$

So if we write the equation of the epipolar line we obtain :

$$l = \begin{pmatrix} -v' \\ u' \\ 0 \end{pmatrix} \tag{8}$$

So for a given point on the first image, the other point lie on a line whose equation is given by $y = \frac{v'}{u'}x$. The previous equation is a linear equation so it means that the optical center belongs to this line. And in this case as we can see on the illustrations, the optical center is also an epipole of the system. So it means that all the points mapped from an image to an other are going to be mapped on affine lines going through the epipole which we showed is the optical center.

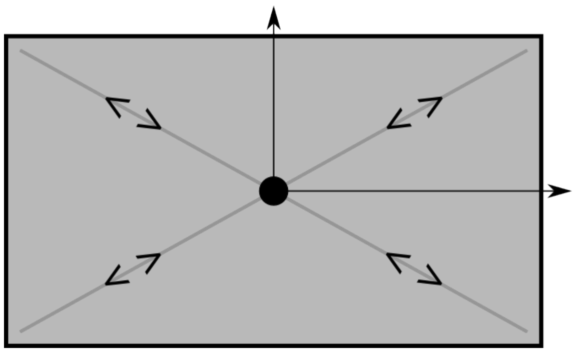

Figure 6: Epipolar line of the forward translation set up

So here we can clearly see that our points are moving along the grey epipolar lines. This traduces in fact the radial behaviour of this system. In fact it exists for each point of the scene one of this lines on which it will move so it will produce this effect of zooming in or out according to the direction of motion due to the epipolar lines.

Let's now determine disparity using the study we made for horizontal translation. In fact, in horizontal translation we showed that disparity was equal to the horizontal shift between two associated points. In this case it's going to be the same idea but on the associated epipolar lines we determined. We need in this case to measure the disparity along the epipolar lines. To compute them, we know that they go through the optical center of the image and thanks to the previous equation we know their equation. So we just need to pick up a set of landmarks and look on epipolar linear lines to find the best match. In this problem, similarly to the one with horizontal or vertical translation we can reduce the set of possible correspondences to a single line which makes computation easier.

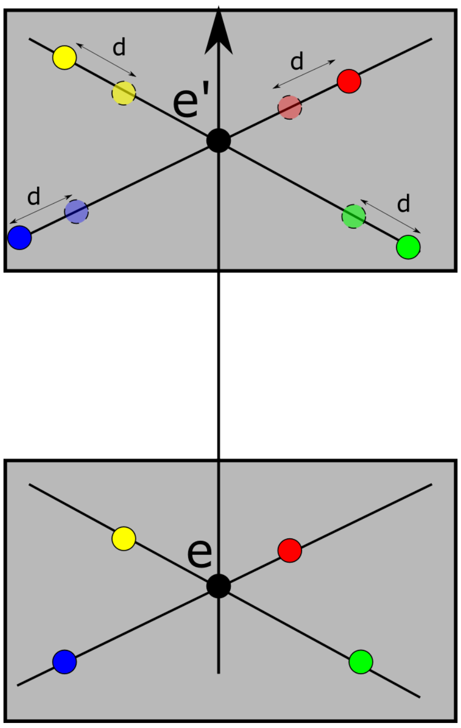

Figure 7: Illustration of disparity

Let's determine the equation to link the disparity with our set up :

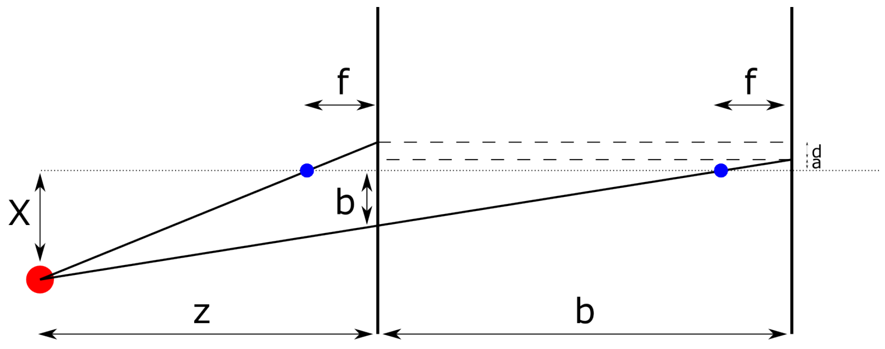

Figure 8: New set up to determine disparity

So let's write down the equations associated to various similar triangles in the previous set up :

$$\frac{b}{X} = \frac{Z}{Z+B-f} \qquad \Rightarrow \qquad b = \frac{ZX}{Z+B-f} \tag{9}$$

Then we have in an other set of triangles

$$\frac{a}{b} = \frac{f}{B-f} \qquad \Rightarrow \qquad a = \frac{fb}{B-f} \tag{10}$$

If we plug b we determined before in the previous equation we obtain:

$$a = \frac{fXZ}{(Z+B-f)(B-f)} \tag{11}$$

And finally in a last set of similar triangles we have :

$$\frac{d+a}{X} = \frac{f}{Z} \qquad \Rightarrow \qquad d = \frac{fX}{X} - a \tag{12}$$

So if we plug a we determined before in the previous equation we can obtain d :

$$d = \frac{fX}{Z} - \frac{fXZ}{(Z+B-f)(B-f)} \qquad \Rightarrow \qquad d = X\frac{f(Z+B-f)(B-f) - fZ^2}{Z(Z+B-f)(B-f)} \tag{13}$$

2.3 Point Correspondence

In this section, we are going to discuss the correlation function presented in class for neighbourhood regions. Let's consider the following correlation function :

$$C(d) = \frac{(\omega - \bar{\omega}).(\omega' - \bar{\omega'})}{||\omega - \bar{\omega}||||\omega' - \bar{\omega'}||} \tag{14}$$

In the previous equation, we consider that ω and ω' are windows over the image to find correspondences. Let's now assume that they are replaced by vectors of dimension n containing the intensity value. Let's now rewrite this correlation applied to vectors:

$$C(d) = \frac{\sum_{i=1}^{n}(x_i - \bar{x}).(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{15}$$

So now let's consider that y_i is defined by :

$$y_i = \lambda x_i + \mu \tag{16}$$

And let's remind that the mean is defined by :

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{17}$$

So we can plug the two previous definitions into the correlation function:

$$C(d) = \frac{\sum_{i=1}^{n}(x_i - \frac{1}{n}\sum_{i=1}^{n} x_i).((\lambda x_i + \mu) - \frac{1}{n}\sum_{i=1}^{n}(\lambda x_i + \mu))}{\sqrt{\sum_{i=1}^{n}(x_i - \frac{1}{n}\sum_{i=1}^{n} x_i)^2}\sqrt{\sum_{i=1}^{n}((\lambda x_i + \mu) - \frac{1}{n}\sum_{i=1}^{n}(\lambda x_i + \mu))^2}} \tag{18}$$

$$C(d)=\frac{\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i).(\lambda x_i+\mu-\frac{1}{n}\lambda\sum_{i=1}^{n}x_i-\frac{1}{n}\sum_{i=1}^{n}\mu))}{\sqrt{\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i)^2}\sqrt{\sum_{i=1}^{n}((\lambda x_i+\mu)-\frac{1}{n}\sum_{i=1}^{n}x_i-\frac{1}{n}\sum_{i=1}^{n}\mu)^2}} \tag{19}$$

$$C(d)=\frac{\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i).(\lambda x_i-\frac{1}{n}\lambda\sum_{i=1}^{n}x_i+\mu-\frac{n\mu}{n})}{\sqrt{\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i)^2}\sqrt{\sum_{i=1}^{n}((\lambda x_i+\mu)-\frac{1}{n}\sum_{i=1}^{n}x_i-\frac{1}{n}\sum_{i=1}^{n}\mu)^2}} \tag{20}$$

$$C(d)=\frac{\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i).(\lambda x_i-\frac{1}{n}\lambda\sum_{i=1}^{n}x_i+\mu-\frac{n\mu}{n})}{\sqrt{\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i)^2}\sqrt{\sum_{i=1}^{n}(\lambda x_i-\frac{1}{n}\lambda\sum_{i=1}^{n}x_i+\mu-\frac{n\mu}{n})^2}} \tag{21}$$

$$C(d)=\frac{\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i).(\lambda x_i-\frac{1}{n}\lambda\sum_{i=1}^{n}x_i)}{\sqrt{\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i)^2}\sqrt{\sum_{i=1}^{n}(\lambda x_i-\frac{1}{n}\lambda\sum_{i=1}^{n}x_i)^2}} \tag{22}$$

$$C(d)=\frac{\lambda\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i).(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i)}{\sqrt{\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i)^2}\sqrt{\lambda^2\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i)^2}} \tag{23}$$

$$C(d)=\frac{\lambda\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i)^2}{\lambda\sqrt{\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i)^2}\sqrt{\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i)^2}} \tag{24}$$

$$C(d)=\frac{\lambda\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i)^2}{\lambda\sum_{i=1}^{n}(x_i-\frac{1}{n}\sum_{i=1}^{n}x_i)^2}=1 \tag{25}$$

Theoretically, based on the definition of the correlation, we know that it belongs to the interval $[-1,1]\subset\mathbb{R}$. In the previous derivation we showed that the maximum of correlation is reached when the intensity of the two patches (windows) is linearly related. It means that if the intensity is linked with a scaling factor λ and a constant offset μ.

3 Practical Problems

3.1 Epipolar Geometry from F matrix

In this first practical problem, we are going to give a deeper presentation of the stereoscopic set up to acquire images. We briefly introduced it when we presented the tripolar geometry at stake in the first exercise.

3.1.1 Theoretical presentation

As mention before, the stereoscopic system is trying to mimic the human perception system using two cameras with a converging angle. So we need two cameras to acquire two slightly different images on which we can find correspondences and estimate some physical properties.
The following picture illustrates the set up of the system.

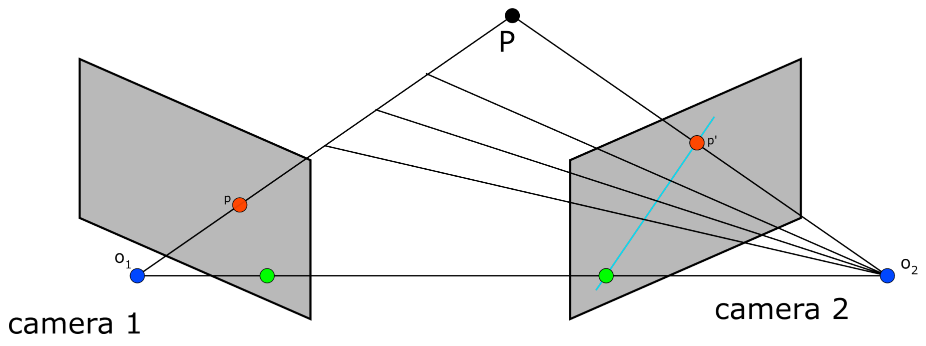

Figure 9: Epipolar geometry in the case of two camera

According to the epipolar constraint we should have all the vectors : $\overrightarrow{O_1p}, \overrightarrow{O_2p'}$ and $\overrightarrow{O_1O_2}$ coplanar.
This leads to formulate the epipolar constraint with this equation:

$$\overrightarrow{O_1p}.[\overrightarrow{O_1O_2} \times \overrightarrow{O_2p}] = 0 \tag{26}$$

We can simplify it, if we introduce the coordinate independent framework associated to the first camera.

$$p.[t \times (\mathcal{R}p')] = 0 \tag{27}$$

We can finally introduce the essential matrix ϵ defined by :

$$\epsilon = [t_x]\mathcal{R} \qquad \Rightarrow \qquad p^T \epsilon p' = 0 \tag{28}$$

With t_x being the skew symetric matrix resulting in the cross product $t \times x$:

$$[t_x] = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = [t_x]X = T \times X \tag{29}$$

As we discussed in the previous project, an important problem still needs to be considered which is the internal and external camera calibration. This can be integrated in our framework using the two following equations : $p = \mathcal{K}\hat{p}$ and $p' = \mathcal{K}'\hat{p}'$ with $\mathcal{K}$ and $\mathcal{K}'$ are the calibration matrices associated to each camera.
According to Longuet-Higgins relation we can now use them in the previous equation providing us with:

$$p^T \mathcal{F} p' = 0 \tag{30}$$

We will rely on this equation in the implementation to compute epipolar lines and determine epipoles locations.

3.1.2 Image acquisition

In this practical application, we did not have two same camera so we used the same camera that we moved to acquire our two images.

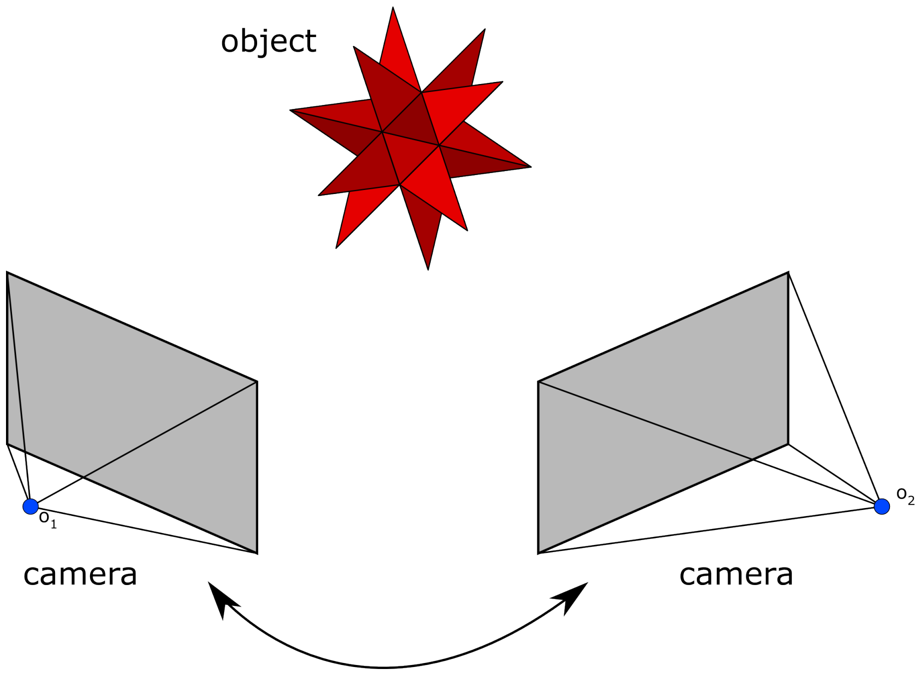

Figure 10: Epipolar geometry in the case of two camera

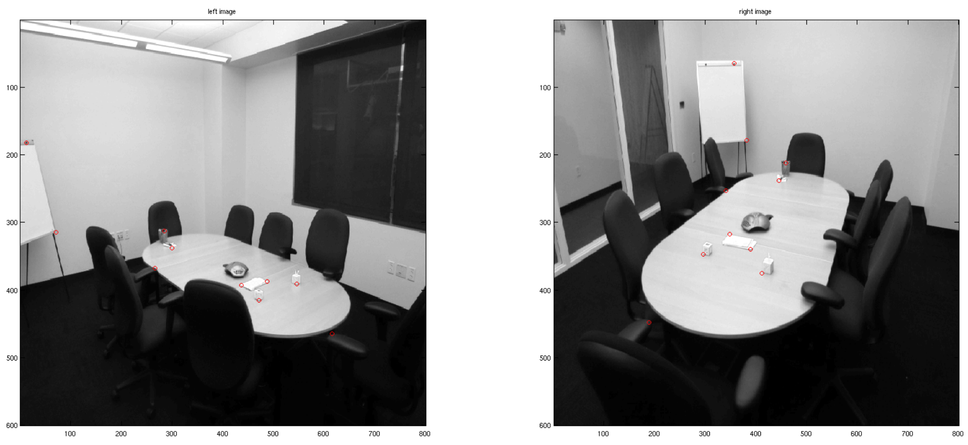

Figure 11: Example of our set up and some points selection

3.1.3 Fundamental matrix computation

As we introduced in the previous theoretical part, the fundamental matrix $\mathcal{F}$ is providing us the epipolar relation between the two camera. Here is a presentation of the method to estimate it.
As we previously did in the camera calibration project we need to estimate the parameters of an unknown matrix which is here the fundamental matrix. We have the following equation:

$$(u, v, 1)\begin{pmatrix} \mathcal{F}_{11} & \mathcal{F}_{12} & \mathcal{F}_{13} \\ \mathcal{F}_{21} & \mathcal{F}_{22} & \mathcal{F}_{23} \\ \mathcal{F}_{31} & \mathcal{F}_{32} & \mathcal{F}_{33} \end{pmatrix}\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0 \tag{31}$$

Due to the fact that we are up to scale on this analysis we can set up one the unknown values of $\mathcal{F}$, $\mathcal{F}_{33}$ to 1. So we can reduce our problem to an estimation of eight unknown parameters. So we need at least 8 pairs of corresponding points to be able to obtain a square system that we can invert and solve using Gauss Jordan elimination. Or we can use more points an use a Linear Least Square estimation (LLS).

In our implementation (presented in the dedicated section) we are going to rely on the Singular Value Decomposition of the Fundamental matrix $\mathcal{F}$ as we did in the camera calibration project. Here is an example of a computed fundamental matrix:

$$\mathcal{F} = \begin{pmatrix} -0.000004326479519 & -0.000014769523227 & 0.005886367091656 \\ 0.000013087874774 & 0.000002053062968 & -0.034802900316000 \\ 0.005393876255144 & 0.034313298289956 & -0.998773052944921 \end{pmatrix} \tag{32}$$

As we can see, as we mentioned earlier, our $\mathcal{F}_{33}$ is really close to 1 and all the values contained in the fundamental matrix are really small.

3.1.4 Computation of epipolar lines

As we mentioned before, a point in the first camera is located on a special line called the epipolar line on the second camera. It works both ways from one camera to an other. Using this principle, we are going to determine the associated set of epipolar lines to a set of points on each camera. The equation of an epipolar line on the right camera is given by is given by :

$$l_r = \mathcal{F}p' \tag{33}$$

The equation of an epipolar line on the right camera is given by is given by :

$$l_l = p^T \mathcal{F} \tag{34}$$

We can then using the information contained in the result we obtain in homogeneous coordinates, we can extract a Cartesian line equation :

$$au + bv + \rho = 0 \qquad \Rightarrow \qquad v = -\frac{a}{b}u - \frac{\rho}{b} \tag{35}$$

Once we have the coefficients of l_l or l_r using the previous equations we can then plot the associated epipolar lines. Here are some results in case of a general stereoscopic set up and then with an horizontal translation :

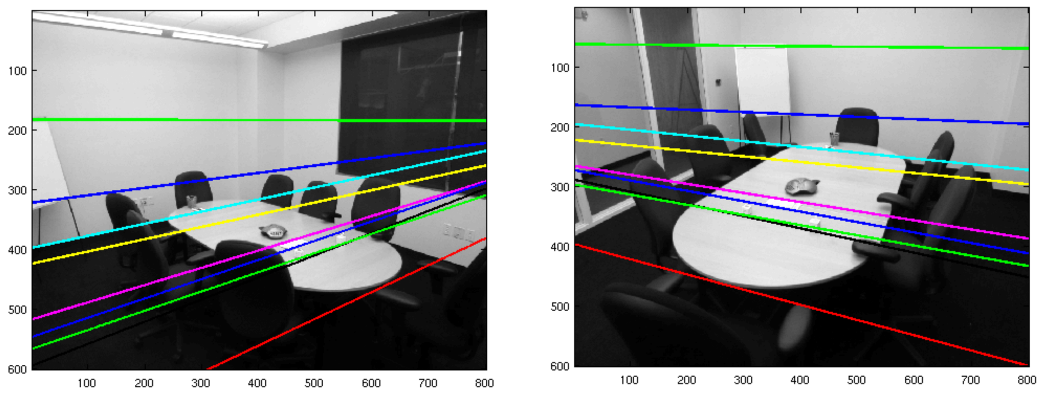

Figure 12: Epipolar lines reconstruction on left and right image

In this last result a small issue is to be notified even if it does not affect the correctness of it. Indeed, one of the green line and the black line are crossing each other on both images, this is a consequence of the accuracy of our point selection for two landmarks which are almost lying on the same line. Everywhere else all the lines look just fine and the use of color allow us to see pairwise matchings.

Another aspect that can be noticed on those images but which will be discuss in the next part is that we can clearly see a pencil, which is a convergence of all the epipolar lines to the epipole which is in this case located outside of the picture.

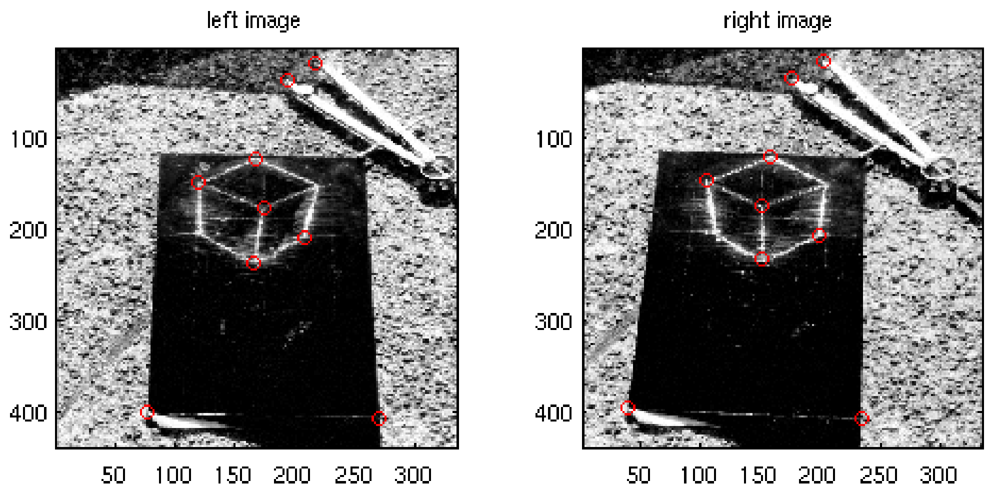

Figure 13: Points selection for horizontal translation stereoscopy

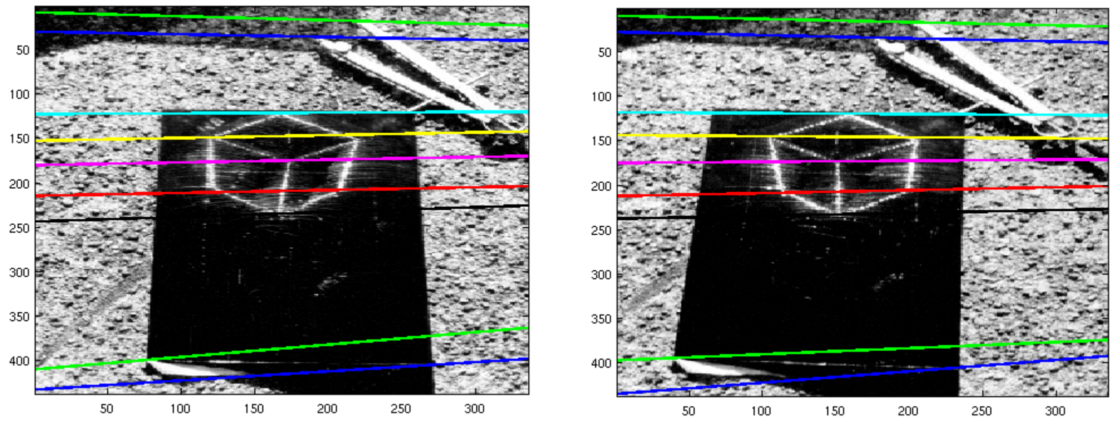

Figure 14: Epipolar lines reconstruction of left and right image in case of translation stereoscopy

In this case the epipolar lines are globally parallel to each other because the motion is supposed to be completely horizontally. In fact, here again the fact that our lines are not strictly parallel can be linked to some uncertainty in point selection and we can explain this issue with the same argument we used in the camera calibration project. Indeed, in this previous project, we illustrated the fact that if all our landmarks used for calibration lied on the same line the estimation algorithm failed. This is the same point to make here. In fact if our points lie on the same scene plan we end up with an homography which a non invertible transformation occurring in perspective geometry that will prevent our singular value decomposition to be effective.

3.1.5 Epipole location

As we briefly mentioned before, we can, in the case of a non horizontal translation stereoscopy set up reconstruct the location of the epipole of our system. In fact, we illustrated the fact that in the horizontal translation case our epipoles are projected to infinity, which is another interesting property we illustrated in the previous project, that parallel lines cross at infinity in the perspective geometry framework. So if we consider again our previous result :

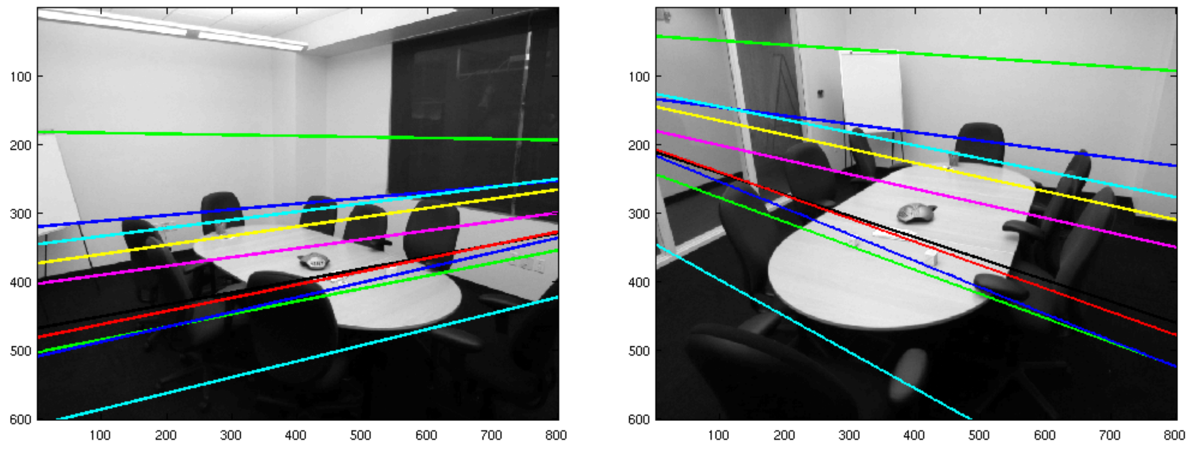

Figure 15: Epipolar lines reconstruction on left and right image

The epipoles are the null spaces of the matrix $\mathcal{F}$, and can be determined using :

$$e_r\mathcal{F} = 0 \qquad \text{and} \qquad \mathcal{F}e_l = 0 \tag{36}$$

To do so we can rely on the computation the normalized (third component equal to 1) eigenvector associated to the smallest eigenvalue of the matrix $\mathcal{F}'\mathcal{F}$.

As we can see on the previous images, epipoles are far away from each pictures, here is an estimation of their location made with our implementation:

$$e_l = \begin{pmatrix} 1390.4 \\ 358.2 \\ 1 \end{pmatrix} \qquad \text{and} \qquad e_r = \begin{pmatrix} -2705.9 \\ -93.4 \\ 1 \end{pmatrix} \tag{37}$$

These two estimated values make sense because if we extend the lines on each image we can find an approximate location close to what our program is estimating. But let's try to convince ourselves that it is working by using a different set up.

Here is an other situation on which we computed our epipole location on right camera. The computation of the eigenvector with last component equal to 1 provides us with these estimated values for epipoles are :

$$e_l = \begin{pmatrix} 210.9968 \\ 120.3111 \\ 1 \end{pmatrix} \qquad \text{and} \qquad e_r = \begin{pmatrix} -35.6204 \\ 149.7950 \\ 1 \end{pmatrix} \tag{38}$$

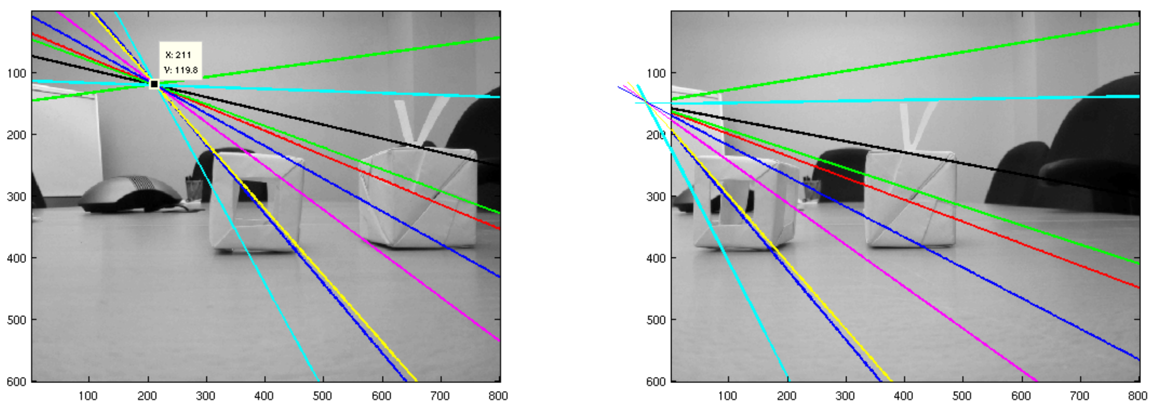

Figure 16: Epipole location on a new experiment

As we can see on the previous picture, our computed value for the right image seems to be pretty accurate, we extended the lines to see where they meet and it seems to be close of the estimated location. An interesting point we can notice is that our left epipole is located on the image and that our estimation is pretty good.
Once again the matching uncertainty between two pair points is really important and could explain some uncertainty in the estimation.

3.1.6 Conclusion

In this first practical problem, we illustrated how we acquired different types of stereo pairs: a set with rotation and translation and a set of horizontal translation. We presented and illustrated the computation of epipolar lines that match associated point pairs on both images and we also computed epipoles location. The results we presented are quite accurate, they required to have a good matching from one image landmark to the corresponding landmark in the other image so our results are not perfect and suffer from uncertainty due to point selection.

3.2 Dense distance map from dense disparity

In this section we are going to consider a pair of stereoscopic images acquired with an horizontal translation to reconstruct the depth map. Indeed we discussed in the first project that one picture was performing a dimension reduction by loosing the depth information. But this information can be recovered using two or more cameras.
We assumed an horizontal stereo image pair but we can notice that this step could also work with rectified stereo image pairs.
Here is an illustration of the set up :

3.2.1 Theoretical presentation

Here is an illustration of our horizontal camera set up :

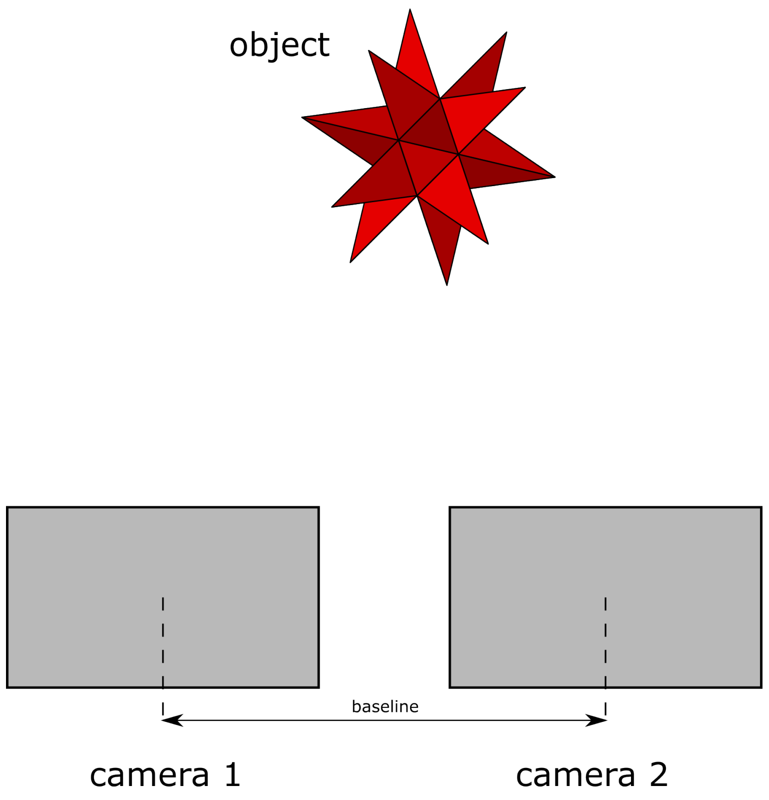

Figure 17: Horizontal stereographic acquisition

Then we can simplify this model to a geometric model with our known and unknown parameters:

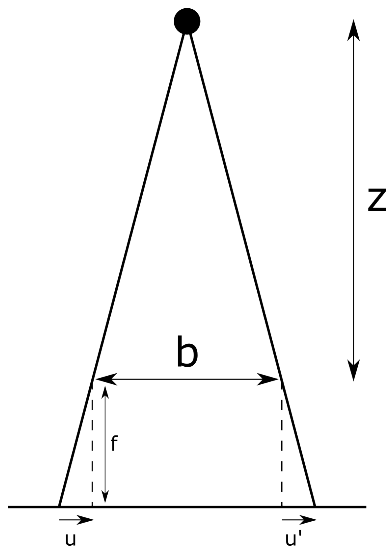

Figure 18: Geometric model

Based on two images and the fact that we know that we know the baseline and the focal length, we are going to measure disparity on the two images and reconstruct the depth map.
With this set up, the disparity is given by:

$$d = |u' - u| \tag{39}$$

Then if we apply the relationships associated to similar triangles we can write :

$$\frac{B}{z} = \frac{d}{f} \tag{40}$$

And finally we obtain:

$$z = f\frac{B}{d} \tag{41}$$

3.2.2 Zero mean normalized cross correlation

Cross correlation is a measurement used to quantify how close two shapes or vector are from each other. This technique is widely used in image processing to perform pattern matching and determine how close is the content of an image to a given pattern.
In this section we are going to implement and use the definition of zero mean normalized cross correlation given in the theoretical part:

$$C(d) = \frac{(\omega - \bar{\omega}).(\omega' - \bar{\omega'})}{||\omega - \bar{\omega}||||\omega' - \bar{\omega'}||} \tag{42}$$

In each case, our ω vectors are windows of mask on one image and the other and we are going to find the best corresponding match. Based on the geometry we introduced before and the mathematical

property on independence to intensity changes this framework seems interesting.
Regarding implementation, we are going to consider each window as a vector and compute a vector wise operation to determine the cross correlation for each given landmark in an image.

Two interesting things can be done here, firstly look at the evolution of normalized cross correlation along a line of the right image and then look at the evolution of correlation as images.

3.2.3 Results

Here are few results we obtained. We did some benchmarking of our implementation with test data sets and then we applied it to our images. Here are the results and some discussion about the choice of parameters. Indeed, the size of the kernel used to compute the correlation matters and influence the kind of results we are going to obtain.

The first test data set we used was the Tsukuba data set presented here :

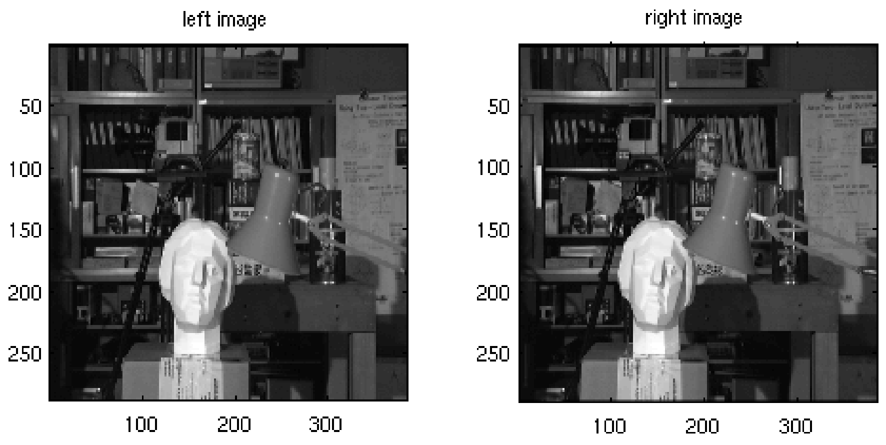

Figure 19: Left and right image of the Tsukuba test data set

Here are the result with a 3 and a 5 kernel :

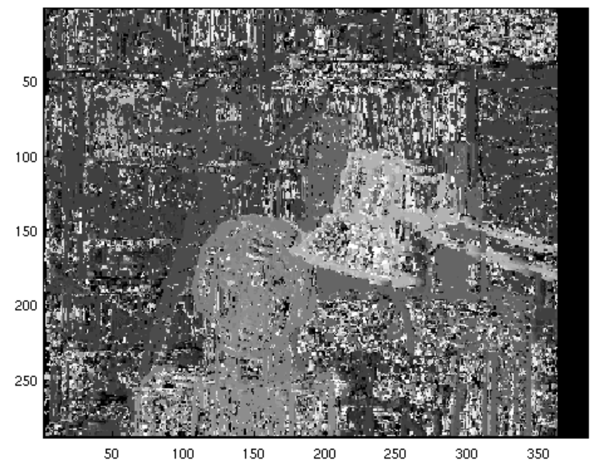

Figure 20: Disparity image with 3 by 3 kernel

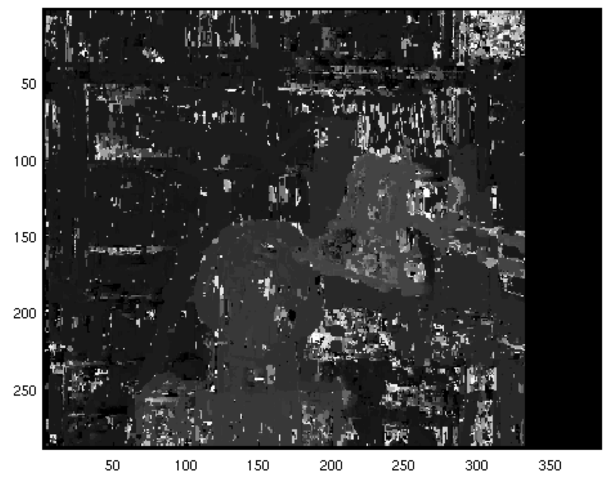

Figure 21: Disparity image with 5 by 5 kernel

As we can see on the two previous images, firstly the size of the kernel influence the "sharpness" of the results. Indeed on the first image the results seems a bit more messy while the second one seems more homogeneous. The reason is that with a bigger kernel it's easier to match structures with more accuracy because the intensity variation pattern is bigger and allow a better match. The small kernel allow us to have more details on structure with a similar disparity but in our analysis that might not be what we want. Indeed, in this type of analysis we are interested in knowing where the objects contained in the scene are located. In fact the disparity measurement will provide us with a mapping of depth computed from the difference between the two images. That's why the second picture is more convenient for this type of analysis. So if we briefly analyse the result obtain with this data set, the "brighter" the closer to the camera. We can clearly locate the lamp firstly, then the sculpted head, its then a bit more complicated but we can also see the camera behind the head, and then at the back we have a more messy picture of what is there, with a prior knowledge of the scene, we might be able to identify majors structures of the shelf but without it's harder.

To then get the distance we apply the equation presented before, which is just a conversion factor multiplying the pixel disparity into a mm distance. In the test data set we did know the baseline, which was 160mm in this particular case but we ignored the focal length, we obtained only a disparity image which might not be very meaningful in terms of values to get the distance, but is able to give an illustration of depth:

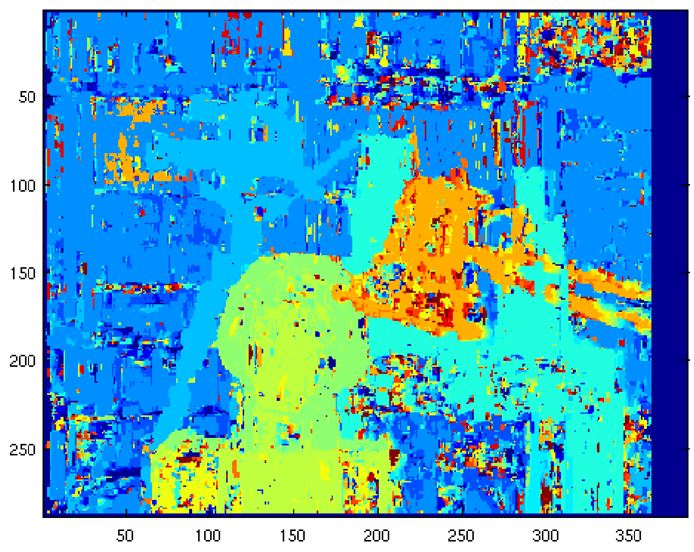

Figure 22: Disparity image with 5x5 kernel and limitation on disparity to 20 pixels

Based on that we took an estimated value of focal length to 24mm and here is the distance map we obtained :

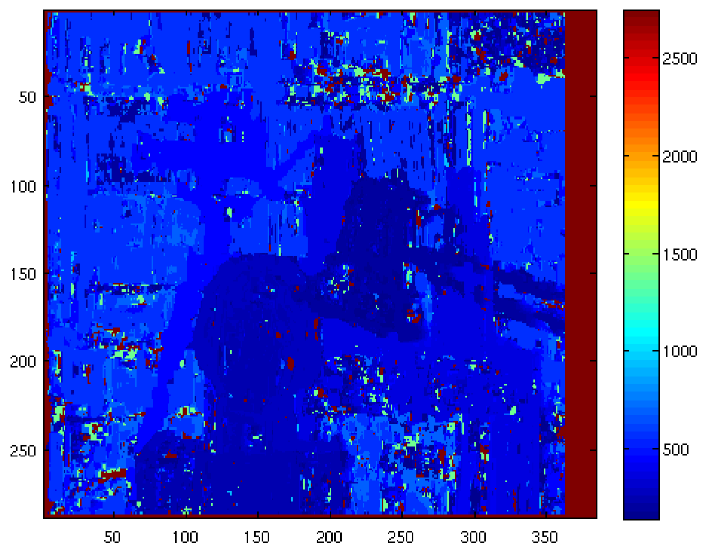

Figure 23: Test distance map with test value of focal length

The use of the color scale on the right allow us to have an estimated depth in millimetres and it seems to be pretty fair, even if we extrapolate a possible value of f the distance between objects seems to make some kind of sense.

Let's now apply it to our own images and see what we obtain and how we can analyse the result.

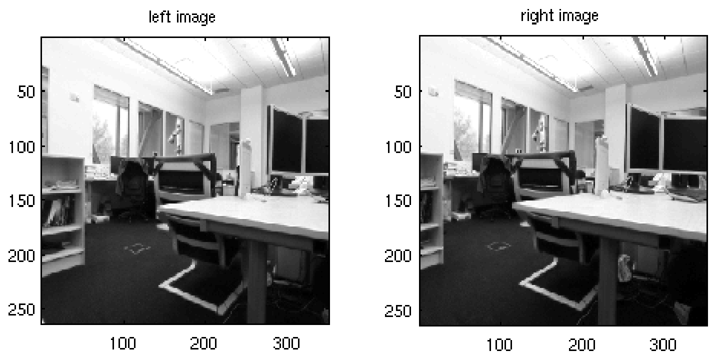

Figure 24: Left and right image acquired with 60mm baseline

As we can see on this picture, there are numerous objects on the image with a quite important variation of depth, here are the results of our processing on those two images.

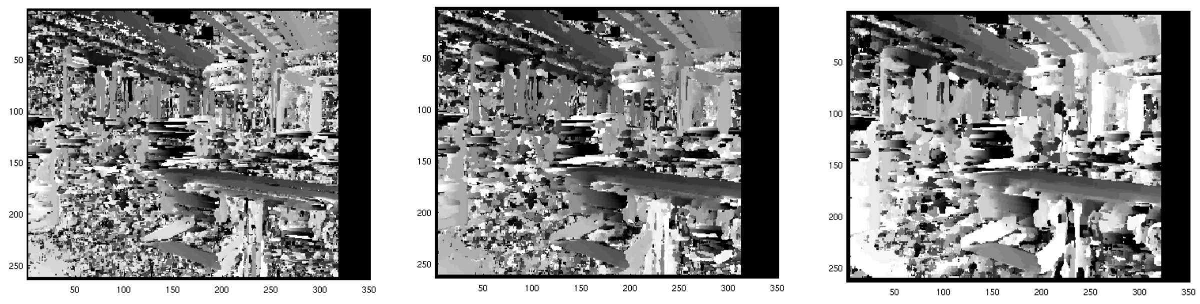

Figure 25: disparity maps with kernel size 5, 7 and 9

As we can see on the previous images, with a small kernel we get a kind of blurry image so we decided to increase the size of the kernel to be able to have less noise and maybe more structures visible. And as we can see we can distinguish some structures if we have a prior knowledge about what are the picture about. Indeed, on the last picture, we can identify correctly the desk table, the screens on the right, some windows at the back and an overall room structure. But only with a knowledge that it's pictures from our lab. In this real situation the result is less clear than what it could be for certain benchmarking data sets. It might also come from the fact that I did not chose the right set of parameters or that maybe there are too much texture less elements in my images.

So here is a last example on an other data set.

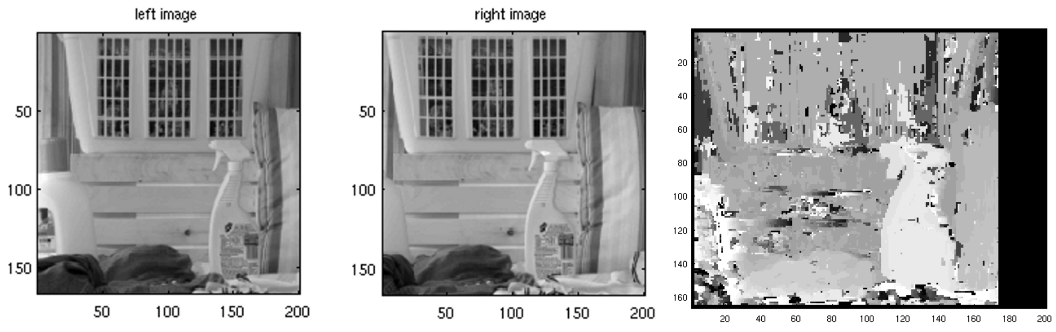

Figure 26: other result with 5 by 5 kernel

3.2.4 Computational expense

The way how my implementation is built required a very long processing time. In fact, the structure is made with five nested **for** loops it requires a long time to compute. Further more the kernel size used to compute cross correlation is also involve and the bigger the kernel the more operations to perform and the longer it requires. So we had to make some optimizations to reduce a bit the computation time.

The first thing we did was to significantly reduce the size of the images we used to reduce the number of pixel to go through on both images.
Then we worked only with reduced size kernel of 3 or 5.
Finally we reduced the set of scanned locations to compute the normalized cross correlation, in fact instead of scanning the whole line on the other image we can introduce a prior knowledge of the maximum disparity which could be used to reduce the scan computation.

3.2.5 Depth perception

Just a few words about a technique to render depth with images acquired with small baseline horizontal translation. In fact, this technique intend to mimic the human visual system using the overlay with two colors of a stereoscopic pair of images. It allows us with special color filtering glasses to reconstruct depth. Here is a small illustration of this technique, it can be performed by modifying the original color channel of two images and blending them together. Here is a example:

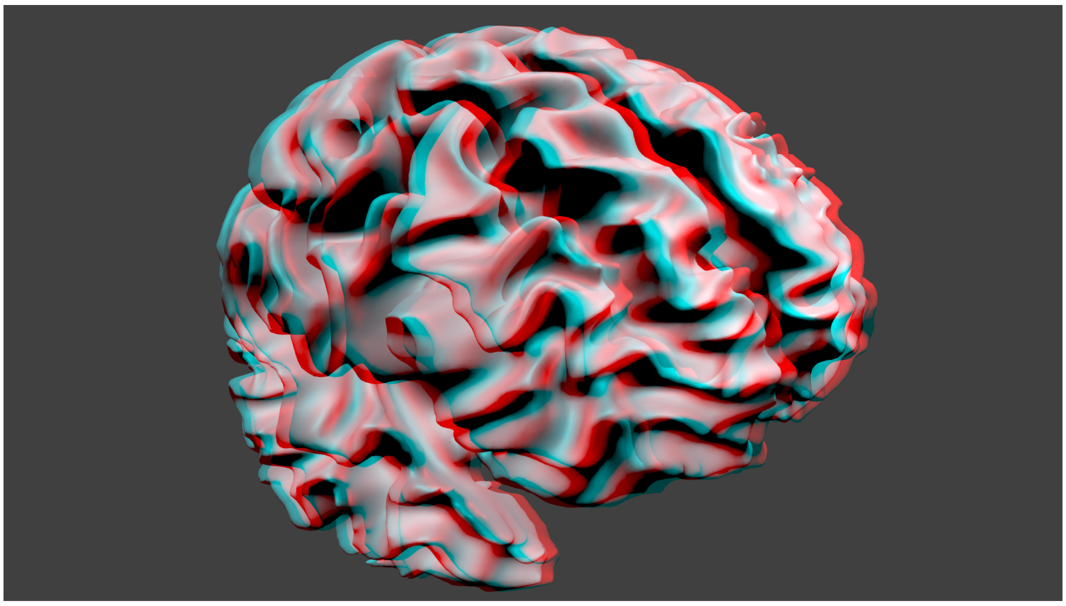

Figure 27: Anaglyph red/cyan stereoscopic image

3.3 3D object geometry via triangulation

In this last section, we are going to discuss the geometry reconstruction using triangulation. Unfortunately, due to a lack of time we are just going in this section to present briefly the method to use to be able to estimate the geometry of an object using triangulation. Here is a set up of this method :

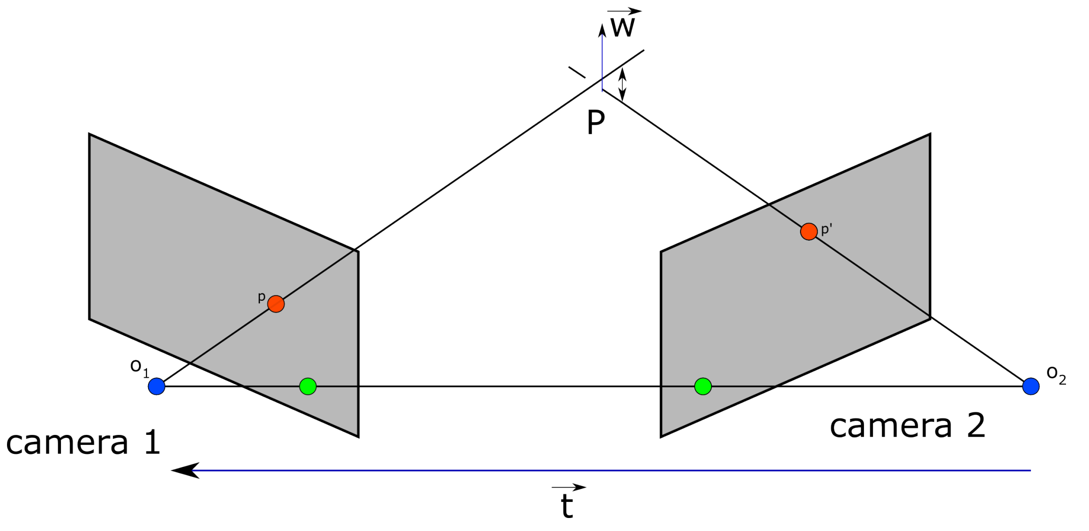

Figure 28: Triangulation set up

In this problem, we have to solve the triangulation equation that comes from the triangular geometry of our problem.

$$\alpha\overrightarrow{O_1p} + \gamma\overrightarrow{w} - \beta Rp' - \overrightarrow{t} = \overrightarrow{0} \tag{43}$$

To do so, we are going to build an equation system allowing us based on a set of landmarks to estimate the values of α, β and γ Thanks to a set of landmarks and the calibration matrices we already studied, we have the following equations:

$$p = \mathcal{M}P \qquad \text{and} \qquad p' = \mathcal{M}'P \tag{44}$$

So we are going to create a system based on a cross product between the previous equation to extract the coordinates of point P which is the point in the world we try to get.

$$\begin{pmatrix} um_3^T - m_1^T \\ vm_3^T - m_2^T \end{pmatrix} P = 0 \tag{45}$$

$$\begin{pmatrix} u'm_3'^T - m_1'^T \\ v'm_3'^T - m_2'^T \end{pmatrix} P = 0 \tag{46}$$

By solving this system we could reconstruct a three dimensional world structure using two rectified stereo images. Unfortunately due to a lack of time we could not get through the whole implementation and produce results.

4 Implementation

4.1 Select Points on Image

To be able to estimate the parameters we need to be able to get the location of a set of landmarks so we implemented a function to get the coordinates of each selected points.

```
but = 1;
while but == 1
   clf
   imagesc(I);
   colormap(gray)
   hold on
   plot(x, y, 'b+','linewidth',3);
   axis square

   [s, t, but] = ginput(1);

x = [x;s];
y = [y;t];

end
```

4.2 Epipolar Geometry computation

This program gets points on the first and second image of the stereo pair and then compute the Fundamental matrix, estimates and plot the epipolar lines associated to each point in each image and also compute the location of each epipole.

```
color = ['r','g','b','c','y','m','k'].

% Select points on both images
I=imread('P1020694_2.JPG');
I2=double(I(:,:,1));
[x,y] = select_points(I2)
I3=imread('P1020697_2.JPG');
I4=double(I3(:,:,1));
[x2,y2] = select_points(I4)

%plot images with selected points
subplot(1,2,1)
imagesc(I2)
hold on
plot(x(1:length(x)-1),y(1:length(y)-1),'or')
axis square
```

```
colormap(gray)
title ('left image')

subplot(1,2,2)
imagesc(I4)
hold on
plot(x2(1:length(x2)-1),y2(1:length(y2)-1),'or')
colormap(gray)
axis square
title('right image')

if(length(x) ~= length(x2))
     disp('not the same number of points')
     break
end


%compute fundamental matrix
F=zeros(1,9);
for i = 1:length(x)-1
     L=[x(i)*x2(i), x(i)*y2(i), x(i), y(i)*x2(i), y(i)*y2(i), y(i), x2(i), y2(i),1]
     F=[F;L]
end
[U,S,V]=svd(F)


F=reshape(V(:,end),3,3)

%compute epipoles
[Dl,El] =eig(F*F')
el = Dl(:,1)./Dl(3,1)
[Dr,Er] =eig(F'*F)
er = Dr(:,1)./Dr(3,1)


%plot epipolar lines
figure(3)
imagesc(I4)
colormap(gray)
figure(2)
imagesc(I2)
colormap(gray)
xx = 0:size(I2,2);
```

```
for i =1:size(x)-1

        temp = F*[x(i) y(i) 1]'; temp = temp./temp(3)
        yy = -temp(1)/temp(2)* xx - temp(3)/temp(2);
        figure(3)
        hold on
        plot(xx,yy,color(mod(i,7)+1),'LineWidth',2)

        temp2 = [x2(i) y2(i) 1]*F; temp2 = temp2./temp2(3)
        yy = -temp2(1)/temp2(2)* xx - temp2(3)/temp2(2);

        figure(2)
        hold on
        plot(xx,yy,color(mod(i,7)+1),'LineWidth',2)

end
```

4.3 Disparity and depth map

Here is our implementation of the disparity computation using two rectified or horizontally acquired set of stereo images. This code uses the Zero Mean Normalized Cross Correlation method as presented before.

```
% Read Images
I = imread('tsukubaleft.jpg');
I2=double(I(:,:,1));
I3=imread('tsukubaright.jpg');
I4=double(I3(:,:,1));

% plot the two images
figure(1)
subplot(1,2,1)
imagesc(I2)
axis square
colormap(gray)
title ('left image')

subplot(1,2,2)
imagesc(I4)
colormap(gray)
axis square
title('right image')
```

```
% Various initializations

%for the whole image
for i = 1+ceil(size(weight,1)/2):1: size(I4,1)-ceil(size(weight,1)/2)
        for j = 1+ceil(size(weight,1)/2):1: size(I4,2)-ceil(size(weight,1)/2)-max_line_scan
            previous_NCC_val = 0;
             estimated_disparity = 0;

             %for selected part of the line (use of prior estimation of disparity)
             for(position=0:1:max_line_scan)
             t=1;

             %for each element of the kernel get vectors to compute NCC
              for(a=-ceil(size(weight,1)/2):1:ceil(size(weight,1)/2))
                    for(b=-ceil(size(weight,1)/2):1:ceil(size(weight,1)/2))
                      w(t) = I4(i+a,j+b);
                          w2(t) = I2(i+a,j+b+position);
                          t=t+1;
                    end
                 end

                 % compute NCC
                 num = dot(w-mean(w),w2-mean(w2));
                 den = norm(w-mean(w))*norm(w2-mean(w2));
                 norm_cross_correl(i,j,position+1) = num/den;

                 % find disparity giving max NCC
                 if (previous_NCC_val < norm_cross_correl(i,j,position+1))
                        previous_NCC_val = norm_cross_correl(i,j,position+1);
                        estimated_disparity = position;
                 end
               end
             OutputIm(i,j)=estimated_disparity;
    end
end
%compute distance

for i = 1:1:size(OutputIm,1)
       for j = 1:1:size(OutputIm,2)
           distance(i,j) = (f*baseline)./OutputIm(i,j);
       end
end
```

5 Conclusion

In this second project, we had the opportunity to work and study some important aspects regarding a very important technique which is stereoscopy and which allow us to reconstruct depth using two or more images.

In fact we firstly studied the theoretical model and techniques about multiple camera geometry, a fundamental stereoscopy algorithm used in robotics where stereo vision is recreated with the forward translation of a single camera. And finally we showed that the correlation function was independent to linear intensity variations.

In the first practical problem, we illustrated how we acquired different types of stereo pairs: a set with rotation and translation and a set of horizontal translation. We presented and illustrated the computation of epipolar lines that match associated point pairs on both images and we also computed epipoles location. This first experiment allowed us to understand and implement the techniques to create a matching between points in a stereo pairs of images.

In all our experiment a very important thing appeared which is precision in point measurement which could influence a lot the result obtained when computing the underlying geometry. We had to try several times to get a good match between landmarks on each images.

In the second experiment we made, we had to deal with depth reconstruction from a pair of rectified image. The implementation we made was using the Zero Mean Normalized Cross Correlation method. Our implementation is very slow and was run on a multi-core machine so it wasn't a too big issue, but on a standard single core or dual core machine it can takes up to several minutes according to the size of the image used. In this experiment, in order to make sure that our implementation was correct we used several benchmarking data sets. The experiment on those data set appeared to be fine but it was more complicated with the real data set.
In this part we had to find a trade off between several parameters to have a not to slow running of the program and a quite good result. We discussed those aspects in the report but maybe other methods could maybe work better, such as the Sum of Square Differences (SSD), with or without zero mean or locally scaled or not, or a more complicated one : the Sum of Hamming Distances.

In the last problem, we could have implemented a way to get the three dimensional geometry of an object using a rectified stereo pair which is a major application of stereoscopy to know the geometry and shape of objects using pictures.

References

[1] Wikipedia contributors. Wikipedia, The Free Encyclopaedia, 2013. Available at: http://en.wikipedia.org, Accessed January, 2013.

[2] D. A. Forsyth, J. Ponce, Computer Vision: A modern approach, Second Edition, Pearson, 2011.

List of Figures