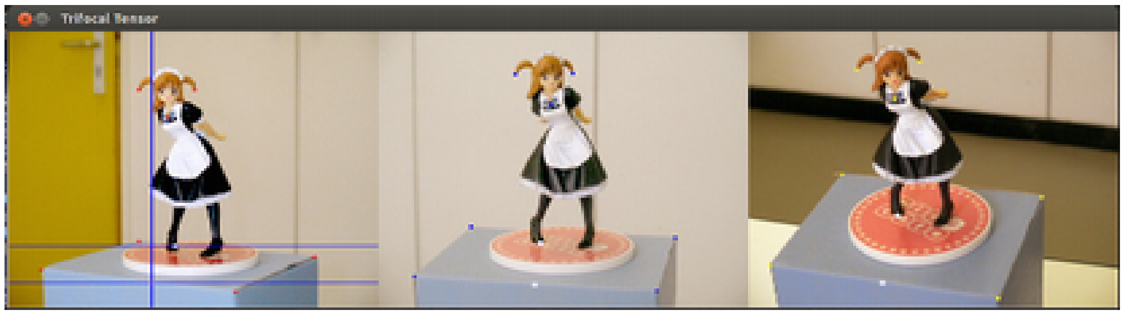
Mathematics project

Trifocal Tensor

Damaris Ankou
Jérémy Ta

Teacher :
Vincent Nozick

December 2012



SUMMARY

Check-list

Elements wanted coded

- The program is written in C/C++ language, using a makefile and it compiles with no warnings. It runs well on Linux.
- The program allows the user to run the program using a specific image or list of points as an option. It uses SDL for rendering.
- The Tensor is well computed.
- The transfer of the points goes well for image1.
- The transfer of the points goes well for image2.
- The transfer of the points goes well for image3.
- Help page as the man, by execution with option -h.

Elements wanted coded without success

-

Elements wanted but not coded

-

Elements not wanted but coded

- Function drawing a line, with Bresenham algorithm.

Elements not wanted coded without success

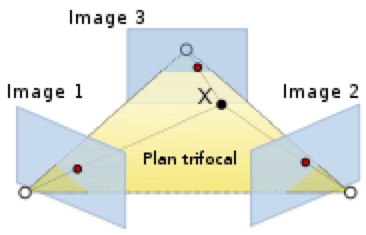
-

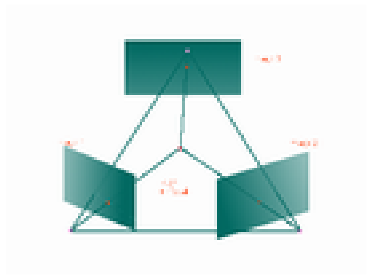
Mathematics

Principle and methodology

The main topic of this project is to find a point on an image, knowing the position of the related point on the two other images. The three images represent the same model, from three different points of view.

It is possible to relate three images through projective geometry, thanks to the acknowledge of each cameras projection center. After locating the intersection of the straight lines passing through the chosen pixel and the camera projection center for two of the three pictures, we can find the pixel location on the third image, just thanks to the straight line linking this intersection and the last camera projection center.





Sadly, it is not our case. We don't know the cameras positions through the scene. However, by using Computer vision mathematics, we can find a solution - yet it is a 'projective' solution, not a metric one - by triangularization. Even if the 3D model is distorted, the transfer seems to match : the location of the point will be an 'at least squares' solution. It's the best solution whatever the considered point.

After some simplifications of the equations describing the model, with Computer vision mathematics, we used this formula :

x_p^k are the coordinates of one point p on image k (we used x for image 1, x' for image 2, $x"$ for image 3) .

T_k^{il} means one value of the Tensor : the value from matrix k, at row i and column l.

This formula is used for computing the tensor and the transfer.

Tensor

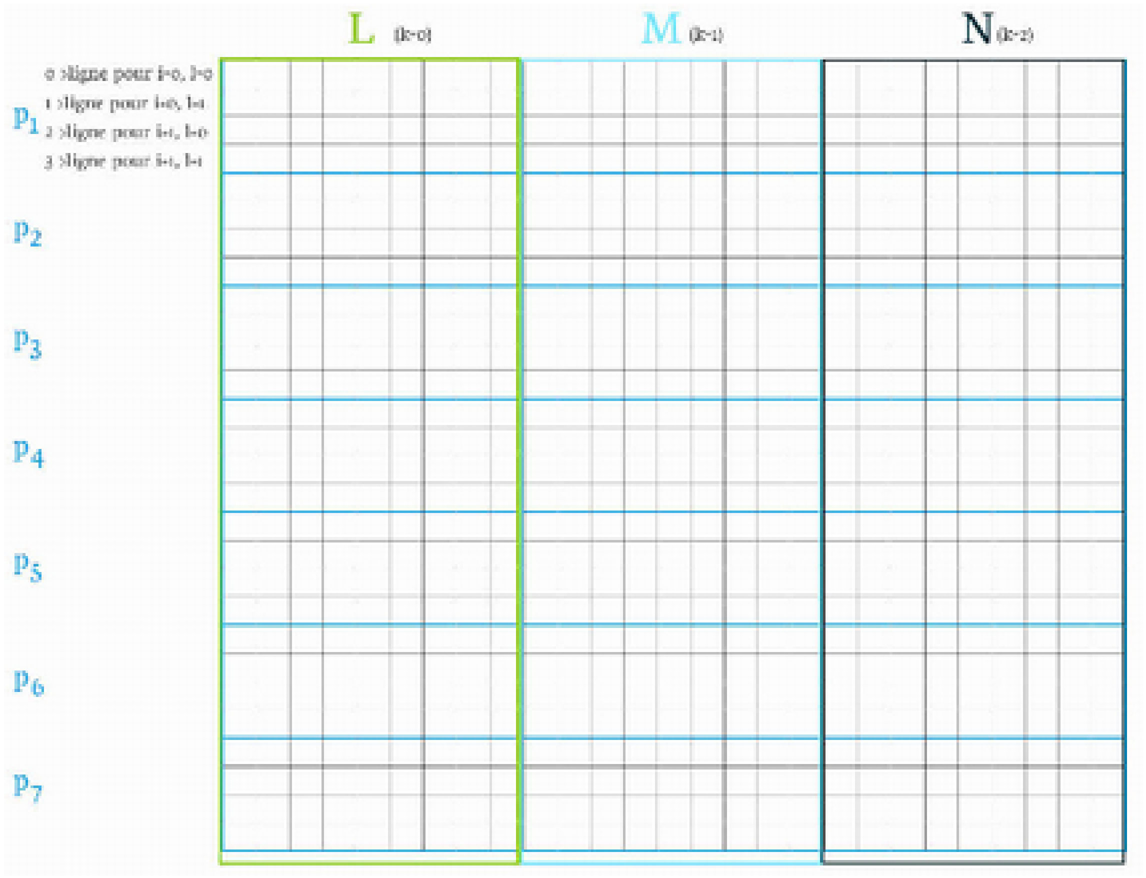
We need to compute the tensor by solving an overdetermined system :

$$A * t = 0$$

A is a matrix which has at least 28 rows (or more, as long as it has 28 + 4n rows, n $\in$ N) and 27 columns.

t is a vector that has 27 rows. 0 is a null vector.

We solved this system using SVD. It provides an “at least squares” solution. See Informatics for its implementation.



Transfer

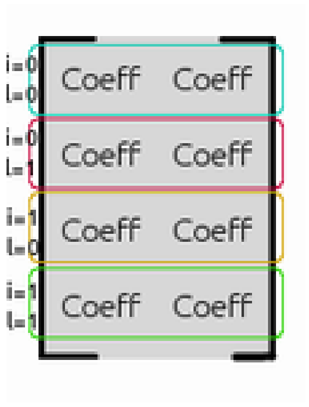
We now have our tensor. Now, when we add a new point on two images., a new systema must be solved, aiming to find the third point.

Again, we will solve an overdetermined system :

$$P * x = b$$

P is a new matrix 4 rows by 2 columns.
x is the vector of coordinates we need to find.
b is an unknown vector to solve, based on the formula.

The formula used again.



Matrix P for transfer to image 1.

For instance, the first coefficient is : $x''_2 (x'_i - x'_2) T_0^{2l}$

Informatics

The main function of the program is **buildMatrixA**.
This is its algorithm :

```
Matrix buildMatrixA( Matrix l1, Matrix l1, Matrix l3) {
        Matrix result;
        for each point p {
                for coordinates k =0 to k<3 {
                        for i = 0 to i <2 by step of one {
                                for l = 0 to l <2 by step of one {
                                        // testing values of i and l with 4 if,
                                        // aiming to fill 12 values for each line.
                                        // The access for each line is done with
                                        // result(4*p + 2*i + l,  ... )
                                        // ... columns change often, it depends on k
```

```
                    //
                }
            }
        }
    }
    return result;
}
```

The second main functions are **transfertTo1**, **transfertTo2**, **transfertTo3**.

```
Matrix transfertTo1(MatrixXd LI, MatrixXd LJ) {
    Matrix P = initWithZero_size(4,2);
    Vector X2 = initWithZero_size(2,1);
    Vector b = initWithZero_size(4,1);

    for i = 0 to i <2 by step of one {
        for l = 0 to l <2 by step of one {

            Set in P good values of the formula.
            Set in b good values of the formula.
            // For instance,
            // P(2*i + l, 0) = ( LJ(2) * ( LI(i)*T(2,l,0) - LI(2)*T(i,l,0 ) ) );
        }
    }

    // Then it does the SVD decomposition of P
    // svd(P);
    // X2 = svd.solve(b);

    return X2;
}
```

Implementation

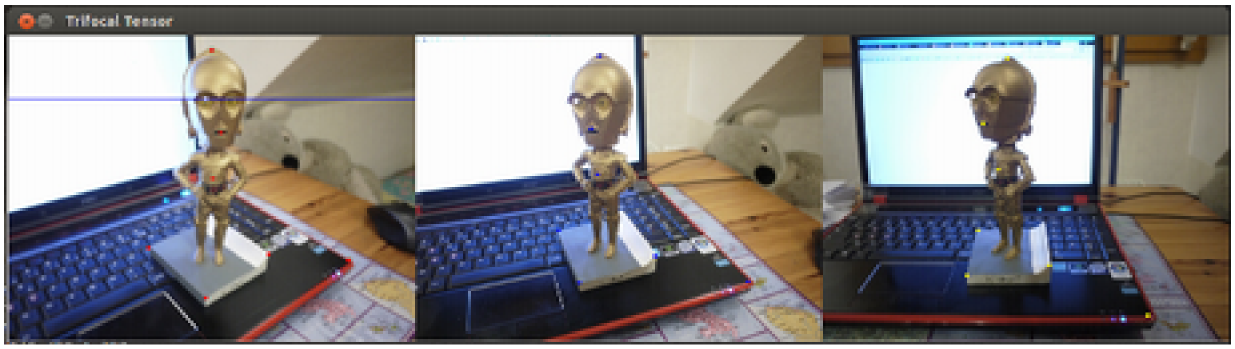
Tensor3d is our class. **ttt** (for trifocal tensor tools) is a file with some functions for computing, or use matrix.

draw is a file with drawing function.

Optimisation

We could have used an implementation more object-oriented.

The program could allowed the user to choose over solving solution : by QR decomposition for instance.



The trifocal tensor used on an other image (c-3po from Star Wars). First, its right-eye is clicked on image 2 then image 3 : the program finds the eye on the first image as expected !

User guide

You can run the program called “trifocal” with some options :

- “h” for showing help in the bash.
- “image1.jpg image2.png image3.jpg” option for loading other images.
  They should be in the folder “input”, and size 400px * 300px.

- “list1.list list2.list list3.list” option can be added to the previous one for loading list of points which match the previous images.

Afterwards,

If you click on an image with right-button, the coordinates of the matching point will appear in the bash.

Press ‘l’ to start drawing points by clicking with left-button on an image. Otherwise, you will just get its coordinates in the bash.
WARNING : you should click on one image then on the other.

Press 'p' to be allowed to click all the points you want in the lists. It needs at minima 7 for each image.

Press 'r' to save this new points list - You should then press 'p' for finishing.

Press 'f' to update the tensor with the new points found : each time you click, the tensor is updated.

Press 'q' or 'suppr' or 'Echap' to close the program. It will save each new list of points clicked in a folder called “save”, in “input” folder.

Conclusion

In conclusion, this project has been rewarding in many ways.
Concerning the organization of the project, at first we took the time to understand the mathematics part of the subject. The main problem was to consider the equation through a matrix model.
We tried step by step to develop the program, validating our equations for the calculation of the tensor and the transfer.