Une grande école pour réussir

Pattern recognition system

Breton-Belz Emmanuel - 2nd Year Internship

ENSICAEN
6, boulevard Maréchal Juin
CS 45 053 – F- 14050 Caen Cedex 4
Tél. +33 (0)2 31 45 27 50
Fax +33 (0)2 31 45 27 60

Table of contents

Chapter 1

Introduction

This application is produced by the University of Salerno. It has been made during an internship and is part of a global pattern recognition application. The final goal is extract the documentation of materials in live from videos captured by a camera headset. The subject of this report concerns the extraction of multiple pattern in a single image.

In this document you will find a description of why the project have been created. Then a presentiation of the composants used like openCV features. And finally the application itself and how it detects patterns.

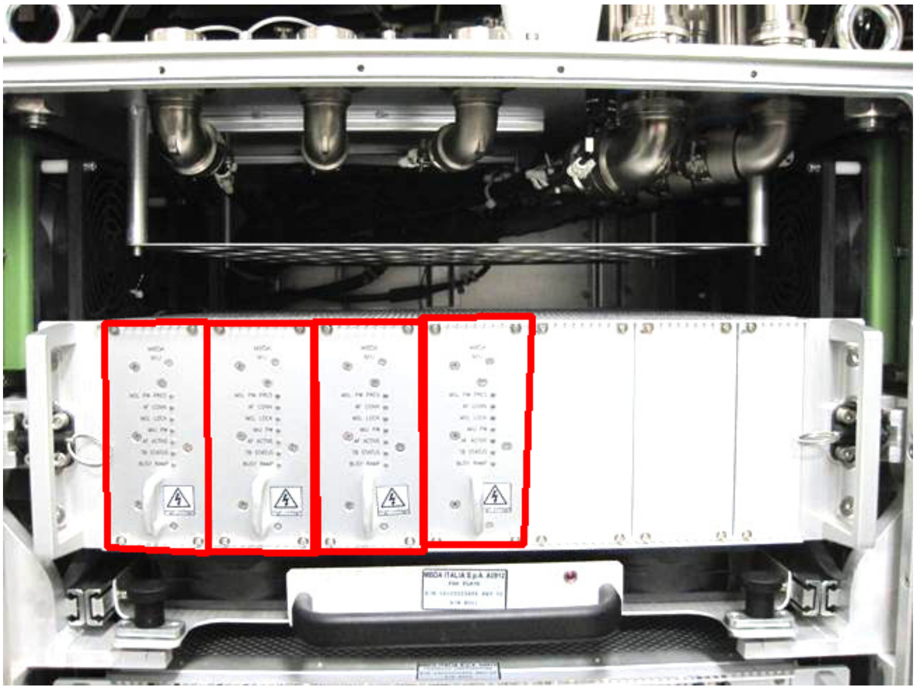

Figure 1.1: Application finding 4 patterns from a single training pattern.

Chapter 2

Need

At work, when people have to make maintenance on the material, they encounter a problem with the density of the maintenance manuals which can make around 700 pages. We try to ease the maintenance by recreate manuals that focuses on the material that the technician is looking at. To obtain this result we have to analyze the images of the camera and extract the type of material. That is what this report deals with.

This solution can be apply to a lot of other objects to find like monuments to extract show the description.

Figure 2.1: type of helmet which the application could be used with

Chapter 3

Solution

3.1 Introduction

To resolve the problem I only worked on Linux LMDE. It should work on other unix based operating system. For windows it requires Qt3 to open the windows that show the images. First I will introduce openCV because all the application is based on it. I will talk more precisely about the keypoints and the descriptors that are computed from them. I will also explain the homography system. All of this component are embedded in the OpenCV library. Then I will explain how I used them in the explication, how I tested it and what I added to obtain the informations about the patterns.

3.2 Composants used

3.2.1 OpenCV

OpenCV is a image analysis and synthesis library that brings all the necessary for video and photo computations. Introducted in 2008, it is now used a lot in the fields that require image analysis. It is built around a module system that allows users to install the library in function of their needs. Except the non free module, the library is licensed BSD which means that we can reuse and modify all the composants freely. In this document I will talk more about precise modules of opencv like that base and the features2d modules.

3.2.2 Scale-Invariant Feature Transform

SIFT points are points of interest in the image. They precise the position of an area where all the pixels have approximatly the same propreties. Using the laplacian of gaussian of the image, maximas and minimas which correspond to these areas can be extracted.

The biggest advantage of these points is the invariance in transformation like rotation, scaling and transalation. Pattern detection algorithm based on this points are unsensitive in term of scale and rotation. That is mainly why it has been choosen for this application.

As a result, by the time that the resolution of the pattern is over a accpetable threshold. We can move and rotate the object whithout alter the detection.

3.2.3 SIFT Descriptor

Descriptors are computed from the keypoints. The descriptors associated to the SIFT key points are localy oriented histograms around a SIFT key point [3] :

- We divide space arround each key point (x,y) N^2 squares of 4 by 4
- We compute the gradiant $G_x(a, b, \sigma)$, $G_y(a, b, \sigma)$ for the 4 by 4 by N^2 points (a,b)
- For each 4 by 4 square, we compute an histogram of the orientations in 8 directions, multiplying by : (1) the module of the gradiant (2) the inverse of the distance to the keypoint (x,y).
- To be invariant in rotation : the local orientation of the key point $\theta(x, y)$ is used as origin (nul orientation) of the histograms.

All the key points and the descriptors extracted and computed in the Constructors part just after. They are stored in a vector.

3.2.4 Homography

Just to give a simplified idea, the familiar Cartesian plane is composed by a set of points which have a one-to-one correlation to pairs of real numbers, i.e. X-Y on the two axis. [1]

In our case, the points are the key points extracted. By extracting the homography matrice, we can transform the corners of our image to 4 other points representing the position of the object on the scene [2].

On the image behind we can see that the homography doesn't depend on the corners. All the key points can be a base to analyse the transformation. And using the perspective transform of openCV, the corners are multiplyed by the homography matrix. That gives us the position of the corners on the scene image.

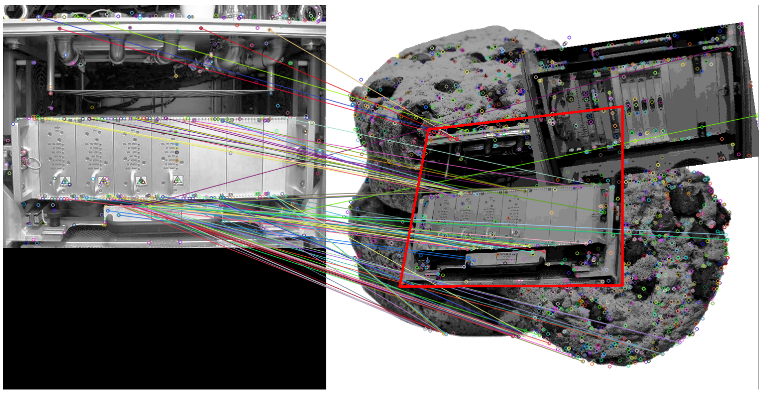

Figure 3.1: Homography with a corner hidden with openCV

3.3 Main functions

3.3.1 Main

The main idea of the system is to extract all the key points of the scene and all the key points of the pattern in the constructors. Like that you don't have to worry about it in the main program. Then you compute the associated descriptors. Both of this containers are implemented in opencv and really easy to use. The main loop consists to match the descriptors of the patterns one by one. Comparing them to the scene image we obtain a vector of dmatches that contains the informations of each match. That's made by the radius matcher included in opencv. Then you can perform an homography that uses the descriptors of the matches found to extract an image that looks like the pattern we are looking for. The advantage of this method is the fact that it doesn't depends on the position and the scaling of the image. We also obtain the corners of the object found which allows us to remove the points of interest and descriptors associated from their respective vectors. That method avoids recomputing all the key points and descriptors which is very expensive in resources. To end the loop we have 2 options. First there is not enough match between the two images. Secondly the number of key values removes after finding a pattern is to low. In both cases the match is not drawn on the image and the next pattern to analyze is set.

3.3.2 Constructors

Scene

The scene represents the vision of the camera and the current position of its analysis. It requires only the location of the image to be built. It sets the key points and computes the descriptors.

It uses a SiftFeatureDector from openCV with a minHessian (exigence of the detector) value of 10000. It is a very high value but the tests show me that more this value is high, better are the results[1] and the quality of the image itself.

Pattern

A pattern represent an object that you want to be recognized on the image. It has simple attributs like a name and a matrix corresponding to the image it represents. But it also has more complexe attributs like the vector of keypoints that we extract in the constructor[2], the matrix of descriptor that represents the propreties of each point and the thresholds. The thresholds are separeted in two different attributs, the minHessian that we told about just before in the scene and the threshold proprely spoken. The second one refers to the radius matching system, it is the maximum acceptable distance between to descriptor during the matching process. There is no universal value so we use values values between 150 to 300 that we have to test to see which one gives the best results.

[1] It also depends on the pattern minhessian value
[2] It can also be stored in a file

The program contains a little part of code that brings the possibility to try differents values quickly.

List of Figures

Bibliography

[1] CorrMap. The homography transformation. http://www.corrmap.com/features/homography_transformation.php, 2013.

[2] D.J. Fleet & A.D. Jepson M. Brubaker, S. Mathe. Tutorial: 2d homographies. http://www.cs.toronto.edu/~jepson/csc2503/tutorials/homography.pdf, 2004-2007.

[3] Antoine Manzarena. Cours etasm. http://www.google.it/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CCIQFjAA&url=http%3A%2F%2Fwww.ensta-paristech.fr%2F~manzaner%2FDownload%2FEPF_Bucarest2006%2FCours_EPF%2FChapitre3.ppt&ei=K4ujVdrjMcOssgHHg4bYDw&usg=AFQjCNEgM2Usx3DKDfBQbo3dXfu6dQ2nzw&bvm=bv.97653015,d.bGg, 2006.