Laboratorio

Tecniche di stabilizzazione digitale di video

Andrea Salvi, Alessandro Venturi

Università degli Studi di Firenze

04/06/2013

Poject Properties - Windows

Include paths (-I)

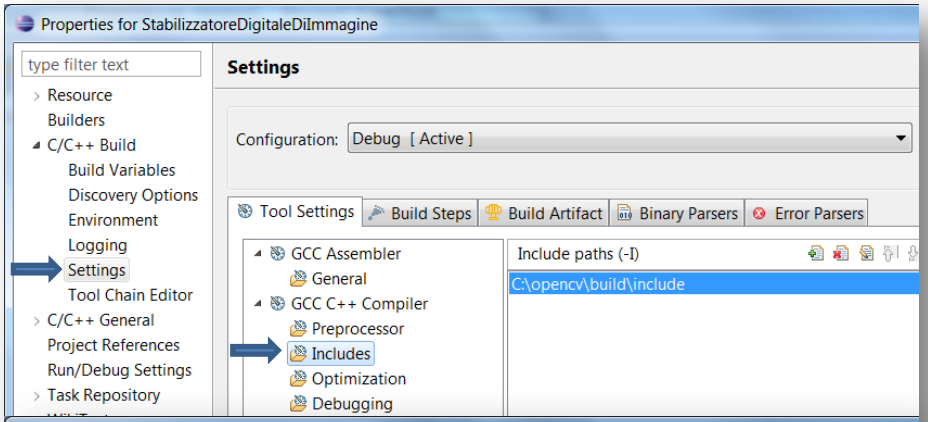
```
"C:\opencv\build\include"
```

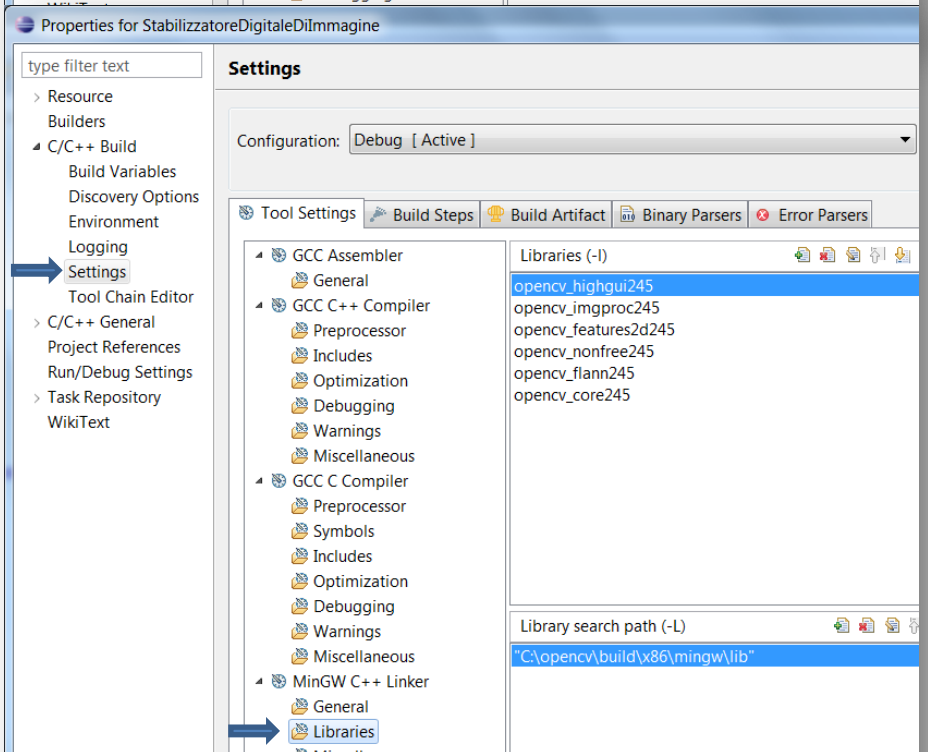
Libraries (-l)

```
opencv_highgui245
opencv_imgproc245
opencv_features2d245
opencv_nonfree245
opencv_flann245
opencv_core245
```

Library search path (-L)

```
"C:\opencv\build\x86\mingw\lib"
```





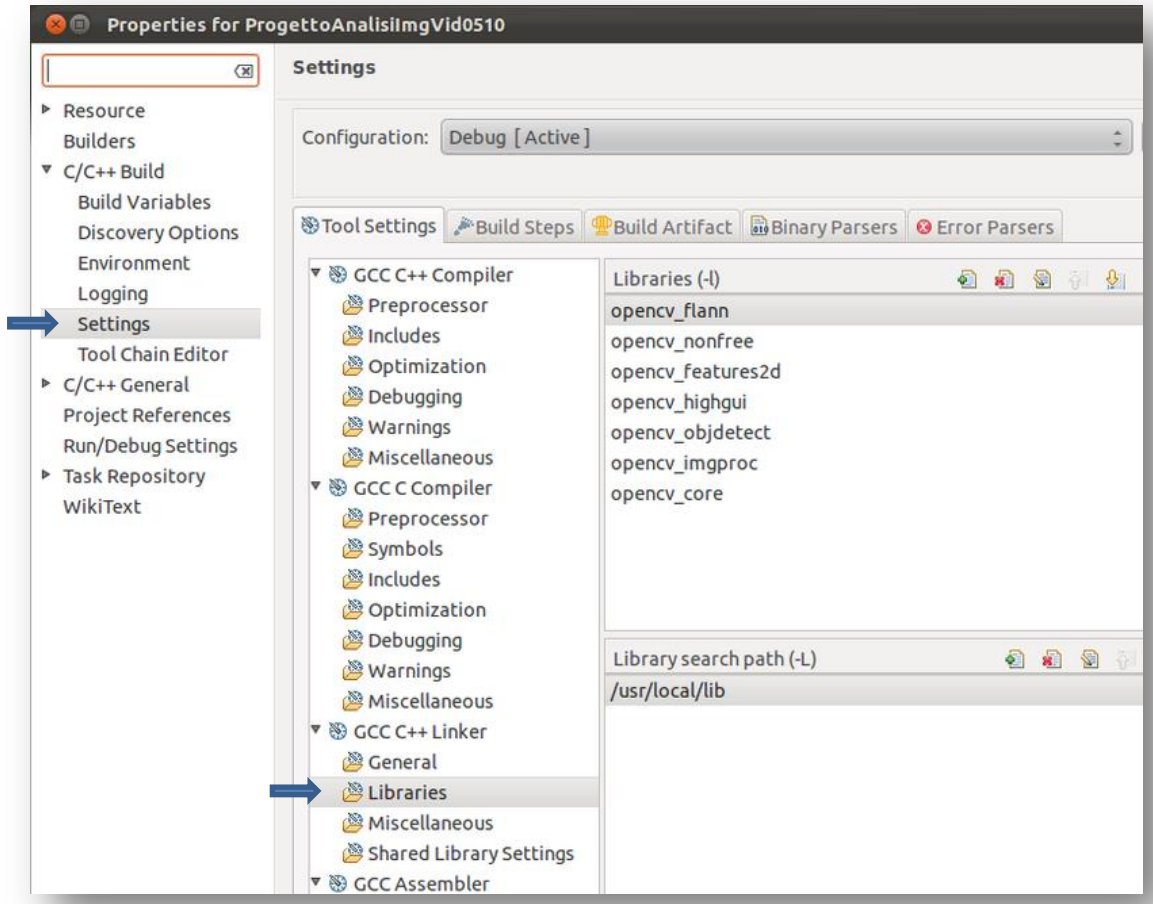
Poject Properties - Ubuntu

Libraries (-l)

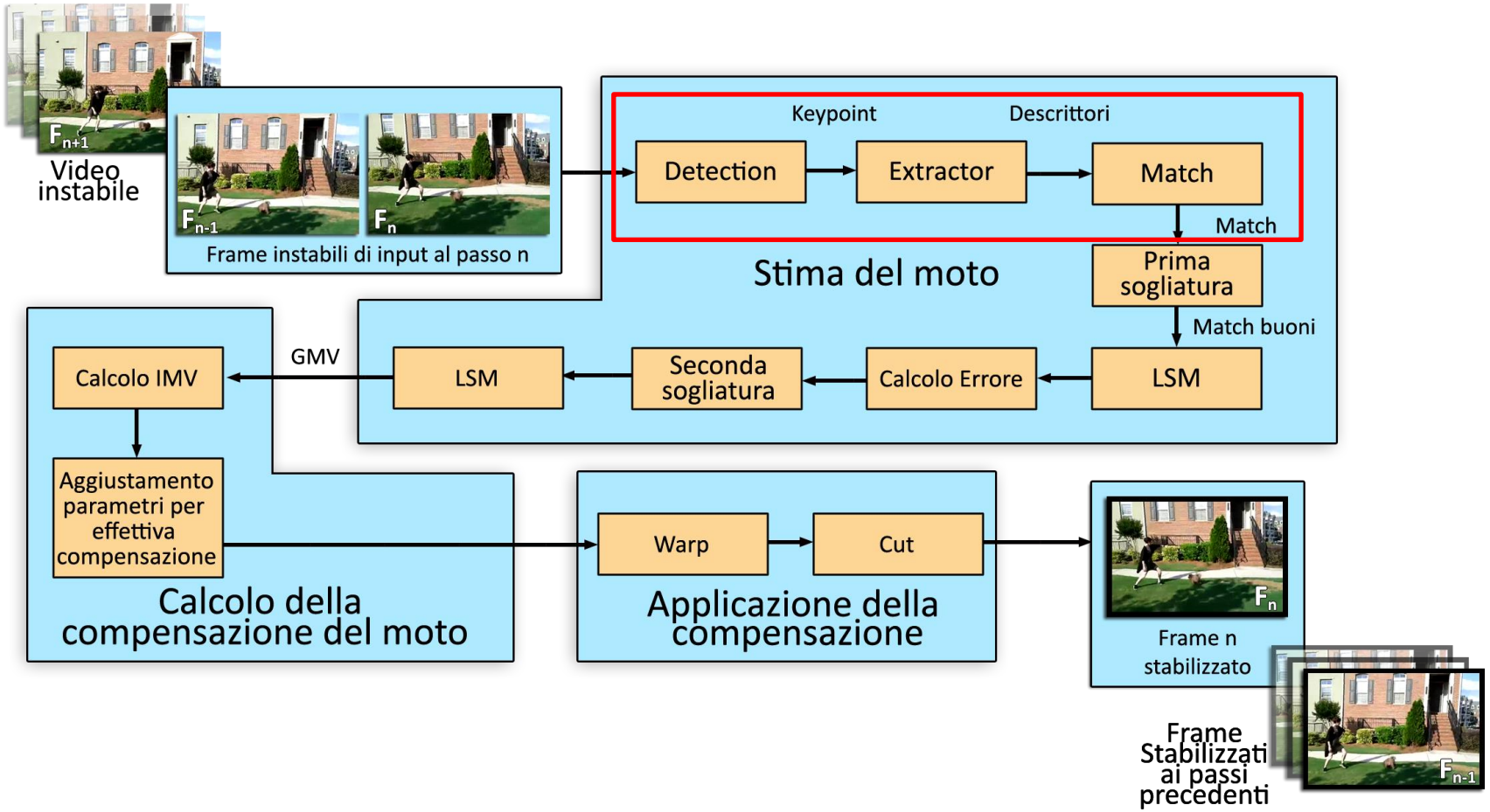
```
opencv_highgui
opencv_imgproc
opencv_features2d
opencv_nonfree
opencv_flann
opencv_core
```

Library search path (-L)

```
/usr/local/lib
```



Creiamo un modulo per calcolare i Local Motion Vector



Cosa ci serve…

- **Grab del frame da un video**

```
VideoCapture capture("../videos/aula.mp4");
Mat frame;
capture >> frame;
```

- **Detection dei keypoint**

```
vector <KeyPoint> keypoints;
Ptr<FeatureDetector> detector = FeatureDetector::create("GFTT");
detector->detect( frame, keypoints );
```

- **Extraction dei descrittori dai keypoint**

```
Mat descriptors;
Ptr<DescriptorExtractor> extractor(Ptr<DescriptorExtractor>
                                                    (new SiftDescriptorExtractor()));
extractor->compute( frame, keypoints, descriptors);
```

```
////////////////////////////////////////////////////////////////////////////////////////////
```

- **Match dei descrittori**

```
vector<DMatch> matches;
Ptr<DescriptorMatcher> matcher = DescriptorMatcher::create("FlannBased");
matcher->match(descriptorsPrec, descriptors ,matches );
```

Per disegnare e mostrare gli output

- **Disegnare cerchi**

```
circle(frame,keypoints[i].pt,3,Scalar(0,0,255),1,8,0);
```

- **Disegnare i Match**

```
Mat out;
drawMatches(
            framePrec,
            keypointsPrec,
            frame,
            keypoints,
            matches,
            out,
            Scalar::all(-1),
            Scalar::all(-1),
            Mat(),
            0);
imshow("Matches",out);
waitKey(WAIT_MS);
```

Detection Extraction e Matching – parte 1

Costruiamo, in modo semplificato, i primi tre moduli dello stabilizzatore

```
#include "opencv2/highgui/highgui.hpp" /*definisce una serie di funzioni per l'I/O e per visualizzare su finestre le immagini o un video; si occupa in pratica della parte user-interface.*/
#include "opencv2/imgproc/imgproc.hpp" /*come dice il nome stesso, libreria con le funzioni di Image Processing; ciò significa che se dovete applicare un filtro ad una immagine o una trasformazione geometrica, questa è la libreria che dovete linkare.*/
#include "opencv2/features2d/features2d.hpp" /*contiene tutte le funzioni per estrarre descrittori dalle immagini per effettuare confronti tra queste a seconda delle caratteristiche selezionate.*/
#include "opencv2/nonfree/nonfree.hpp"
#include "opencv2/flann/flann.hpp" /*contiene le funzioni per la ricerca approssimata*/
#include "opencv2/core/core.hpp" /*libreria principale di OpenCV; contiene tutte le strutture dati e le funzioni di base per lavorare sulle immagini.*/


using namespace std;
using namespace cv;

#define NUM_KEYPOINTS 200 /*serve per settare il numero di keypoint su cui vogliamo lavorare.*/

#define WAIT_MS 0 /*millisecondi sui waitKey*/

//...
```

Detection Extraction e Matching – parte 2

```
int main() {
	//struttura per caricare il video
	VideoCapture capture("../videos/aula.mp4"); //tra virgolette la directory del video di input

	Ptr<FeatureDetector> detector = FeatureDetector::create("GFTT"); //GFTT = GoodFeatureToTrack
	Ptr<DescriptorExtractor> extractor(Ptr<DescriptorExtractor>(new SiftDescriptorExtractor()));
	Ptr<DescriptorMatcher> matcher = DescriptorMatcher::create("FlannBased");

	Mat frame;                    //frame n
	Mat framePrec;     //frame n-1

	vector <KeyPoint> keypoints;   //keypoint al frame n
	vector <KeyPoint> keypointsPrec;          //keypoint al frame n-1

	Mat descriptors;   //descrittori dei keypoint al frame n
	Mat descriptorsPrec;          //descrittori dei keypoint al frame n-1

	vector<DMatch> matches;         //struttura dove verranno salvati i match

	capture >> frame; //metto dentro la Mat frame il prossimo frame del video

	//passo base sul primo frame
	//trova i keypoint tramite GFTT
	detector->detect( frame, keypoints );
	if(keypoints.size()>NUM_KEYPOINTS) keypoints.resize(NUM_KEYPOINTS);
	//estrae i descrittori SIFT dei keypoint trovati al passo precedente
	extractor->compute( frame, keypoints, descriptors);
	 //...
```

Detection Extraction e Matching – parte 3

```
//...
for (;;) {
    //le strutture dell'iterazione precedente diventano istanze Prec
    framePrec=frame.clone();
    descriptorsPrec=descriptors.clone();
    keypointsPrec=keypoints;

    //grab del frame
    capture >> frame;
    if (frame.empty()) break;

    //troviamo i key point col GFTT
    detector->detect( frame, keypoints );
    if(keypoints.size()>NUM_KEYPOINTS) keypoints.resize(NUM_KEYPOINTS);

    //disegnamo i keypoint del frame corrente
    for (int i = 0; i < keypoints.size(); i++){
     circle(frame,keypoints[i].pt,3,Scalar(0,0,255),1,8,0); //BGR
    }
    imshow("KeyPoint",frame);
    waitKey(WAIT_MS);
    //...
```

Detection Extraction e Matching – parte 4

```
        //...

        //estraiamo i descrittori SIFT dai keypoint
        extractor->compute( frame, keypoints, descriptors);

        //fare i match con i descrittori del frame precedente con quello attuale
        matcher->match(descriptorsPrec, descriptors ,matches );

        //disenga i match
        Mat out;
        drawMatches(framePrec,keypointsPrec,frame,keypoints,matches,out,Scalar::all(-
        1), Scalar::all(-1),Mat(),0);
        imshow("Matches",out);

        waitKey(WAIT_MS);
    }
   return 0;
}
```

Step by Step

Impostando lo STEP_BY_STEP a true analizzaremo passo passo l'esecuzione dell'algoritmo di stabilizzazione. Settare i parametri iniziali nel seguente modo:

```
#define THRESHOLD 200
#define WEIGHT_ERR 0.35 //alpha
#define TYPE_OF_DETECTOR "GFTT"

#define theta_threshold 3
#define CROP 0 // Crop the borders of the output video. We have crop with positive values (25 is a good value)

#define OUTPUT_COMPARE false //If true, compares the input video with the stabilized video.
#define IMSHOW false //If true, shows the output video during processing. If OUTPUT_COMPARE is true, it shows the input video next to the stabilized one
#define SAVE_OUTPUT false //If true, saves the stabilized videos. If OUTPUT_COMPARE is true, it saves the input video next to the stabilized one
#define SHOW_LOCAL_MOTION_VECTOR false //If true then show local motion vector on the current frame
#define STEP_BY_STEP true //If true, it shows the execution step by step
#define WAITms 0

#define auto_DELTA true
```

File di input shaky.mp4

```
string nameVideoInput="shaky";
```

Soffermarsi al frame 3 e vedere passo passo l'esecuzione

Preparazione al test sul δ

Analizziamo ora come cambiano i risultati al variare del parametro δ. Ricordiamo che $IMV(n) = \delta IMV(n-1) + GMV(n)$. Settare i parametri iniziali nel seguente modo:

```
#define THRESHOLD 1000 //max number of keypoint
#define WEIGHT_ERR 0.35 //alpha
#define TYPE_OF_DETECTOR "GFTT"

#define theta_threshold 3
#define CROP 0 // Crop the borders of the output video. We have crop with positive values (25 is a good value)

#define OUTPUT_COMPARE true //If true, compares the input video with the stabilized video.
#define IMSHOW false //If true, shows the output video during processing. If OUTPUT_COMPARE is true, it shows the input video next to the stabilized one
#define SAVE_OUTPUT true //If true, saves the stabilized videos. If OUTPUT_COMPARE is true, it saves the input video next to the stabilized one
#define SHOW_LOCAL_MOTION_VECTOR false //If true then show local motion vector on the current frame
#define STEP_BY_STEP false //If true, it shows the execution step by step
#define WAITms 0
```

File di input fiesole.mp4

```
string nameVideoInput= "fiesole";
```

Al ciclo **for** con indice x impostare **x<#numDiFrameDaElaborare**

Un buon valore per vedere dei risultati è 40sec*30fps = **1200**

Test sul δ

Eseguire il programma in queste tre varianti e osservare i risultati salvati all'interno della cartella del progetto:

- $\delta_{T_x} = 1,\ \delta_{T_y} = 1$

```
#define auto_DELTA false //if true, dynamic deltas are used
float DELTAX=1;
float DELTAY=1;
```

- $\delta_{T_x} = 0,\ \delta_{T_y} = 0$

```
#define auto_DELTA false //if true, dynamic deltas are used
float DELTAX=0;
float DELTAY=0;
```

- $\delta_{T_x},\ \delta_{T_y}$ dinamici

```
#define auto_DELTA true //if true, dynamic deltas are used
```

Esecuzione completa

Eseguiamo ora l'algoritmo di stabilizzazione e vediamo il risultato. Il video di output si trova nella cartella del progetto. Settare i parametri iniziali nel seguente modo:

```
#define THRESHOLD 1000
#define WEIGHT_ERR 0.35 //alpha
#define TYPE_OF_DETECTOR "GFTT"

#define theta_threshold 3
#define CROP 25 // Crop the borders of the output video. We have crop with positive values (25 is a good value)

#define OUTPUT_COMPARE true //If true, compares the input video with the stabilized video.
#define IMSHOW false //If true, shows the output video during processing. If OUTPUT_COMPARE is true, it shows the input video next to the stabilized one
#define SAVE_OUTPUT true //If true, saves the stabilized videos. If OUTPUT_COMPARE is true, it saves the input video next to the stabilized one
#define SHOW_LOCAL_MOTION_VECTOR false//If true then show local motion vector on the current frame
#define STEP_BY_STEP false //If true, it shows the execution step by step
#define WAITms 0

#define auto_DELTA true
```

File di input shaky.mp4

```
string nameVideoInput="shaky";
```