Università degli studi di Napoli
Federico II

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Aerospaziale

Tesi di Laurea

Tecniche di *Digital Image Processing* per l'analisi strutturale

Relatore

Chiar.mo Prof. Ing.

Francesco Marulo

Candidato

Giampaolo Pagliuca

Matricola M53/82

Anno Accademico 2011/2012

"La disumanità del computer sta nel fatto che, una volta programmato e messo in funzione, si comporta in maniera perfettamente onesta."

(Isaac Asimov)

Prefazione

Nelle ultime decadi, l'utilizzo del calcolatore per applicazioni scientifiche è cresciuto notevolmente. Sono state sviluppate nuove tecniche, nuovi metodi di analisi e, sopratutto, i tempi di calcolo si sono progressivamente ridotti.

Questa crescita, più che ad un'evoluzione ordinata, è paragonabile ad una vera e propria esplosione disordinata. Il mondo della *Computer Science* procede a velocità elevatissima, spinto da sviluppi *hardware* e *software* inimmaginabili fino a pochi anni fa. Negli anni ottanta e novanta si sono avute le prime esperienze nel campo della *Computer Vision* e della *Digital Image Processing*.

In questo contesto, non sempre le applicazioni di queste nuove tecniche sono state al passo con i tempi. I primi esperimenti di analisi strutturale con la *Digital Image Processing* risalgono a metà anni novanta: ben una decade dopo la nascita della *Computer Vision*.

Ad oggi, l'uso di tecniche come la *Image Correlation* o, ancor più, la *Object Recognition* sono relegate ai laboratori, a portata di pochi esperti di informatica e ingegneria strutturale a causa dell'intrinseca difficoltà d'uso.

L'obiettivo di questa Tesi è quello di sviluppare un *software* che applichi le ultime e più recenti tecnologie della *Digital Image Processing* e della *Computer Vision* all'analisi strutturale, con uno sguardo attento alla semplicità d'uso e all'accuratezza dei calcoli richiesti per le applicazioni ingegneristiche, in particolare aeronautiche.

1 Introduzione

Le tecniche di *Digital Image Processing* sono molte e variegate, e non tutte forniscono informazioni utili per l'analisi strutturale.

Sono interessanti le tecniche non invasive per la stima degli spostamenti, velocità ed accelerazione. La natura del campo degli spostamenti può appartenere a due categorie: piccoli spostamenti e grandi spostamenti, in riferimento ad una dimensione caratteristica del problema.

É evidente che non sarà possibile adoperare la medesima tecnica per entrambe le categorie. Fortunatamente, la *Computer Vision* ci fornisce due metodologie formidabili: l'*Object Recognition* (informalmente *Object Detection*) e l'*Image Correlation*. Ognuna di esse presenta pro e contro, ma sono utilizzabili rispettivamente per la stima di grandi spostamenti in punti ben definiti e per ricavare il campo completo dei piccoli spostamenti.

Accanto a queste tecniche, una metodologia molto recente (la cosiddetta *Edge Detection* permette di ricavare la geometria base di strutture reali utilizzando la semplice fotografia: anche questo campo della *Digital Image Processing* è stato esplorato.

1.1 Campi di applicazione

Queste tecniche presentano tre caratteristiche notevoli: non invasività, semplicità ed economicità che le rendono accessibili e interessanti per l'ingegneria.

In campo aeronautico, un futura applicazione potrebbe portare al calcolo automatico delle velocità di atterraggio e decollo. A tale scopo sarebbe sufficiente una telecamera che registri la fase di volo interessata, per poi analizzare i dati in pochi secondi con un moderno *computer*.

Nel campo dell'ingegneria strutturale, la tecnica permetterà di calcolare il campo degli spostamenti e le deformazioni di strutture reali, grazie a delle fotocamere opportunamente posizionate.

1.2 Primi passi

Con questa Tesi, è iniziato lo sviluppo del *software* in grado di proporre, in modo semplice, l'uso di queste tecniche. Se n'è validato il funzionamento, effettuando diverse prove: l'analisi completa del moto di un pendolo, il campo degli spostamenti per moti rigidi (traslazione e rotazione) e il riconoscimento di una struttura reticolare.

Il programma è stato scritto in un linguaggio di programmazione moderno, il linguaggio *Python*, consapevoli che nei tempi dell'informatica il presente è già passato.

Ovviamente, il percorso da compiere è ancora lungo: questo è solo il primo passo.

Parte I

Tecniche di base

2 Le tecniche e gli algoritmi

Come ogni sviluppo scientifico, le basi della *Computer Vision* e in particolare della *Digital Image Processing* affondano nella matematica.

É opportuno, quindi, dare un rapido sguardo alle formulazioni matematiche e agli algoritmi che, implementati al calcolatore, permettono al *software* di funzionare.

Ci soffermeremo in particolare su:

- *Image Correlation*;
- *Object Detection* con il *Cascade Classifier*;
- *Edge Detection* con il *Canny Detector* e le linee di *Hough*.

3 Image Correlation

L'*Image Correlation* è una tecnica di *Digital Image Processing* su base ottica, non intrusiva, per il calcolo del campo di deformazione e di spostamento.

La tecnica utilizza due immagini simili, nello specifico fotografie digitali, che rappresentano lo stato di un oggetto prima e dopo la deformazione. Le immagini sono confrontate utilizzando la tecnica matematica della correlazione digitale, ricavandone il campo di deformazione.

Il confronto fra immagini si basa sulla capacità di riconoscere un campione, preso dall'immagine di stato indeformato, nell'immagine di stato deformato.

L'immagine di stato indeformato viene divisa in un certo numero di campioni. Maggiore accuratezza è ottenuta dividendo l'immagine originale in sempre più campioni, di dimensione minore. Ciascun campione è ricercato, mediante tecniche statistiche e calcolo delle matrici, nell'immagine di stato deformata.

La ricerca di un campione restituisce una distribuzione bidimensionale di probabilità. Il valore più alto (picco) di probabilità sulla mappa bidimensionale, fornisce le coordinate del campione nell'immagine di stato deformato.

É possibile ottenere lo spostamento relativo del campione, in termini di `pixel`.

Nelle applicazioni più recenti (ad es. [1]) si assume una forma matematica per il campo degli spostamenti. In questo modo, l'accuratezza ed i tempi di calcolo sono notevolmente migliorati, a scapito della giusta interpretazione del problema.

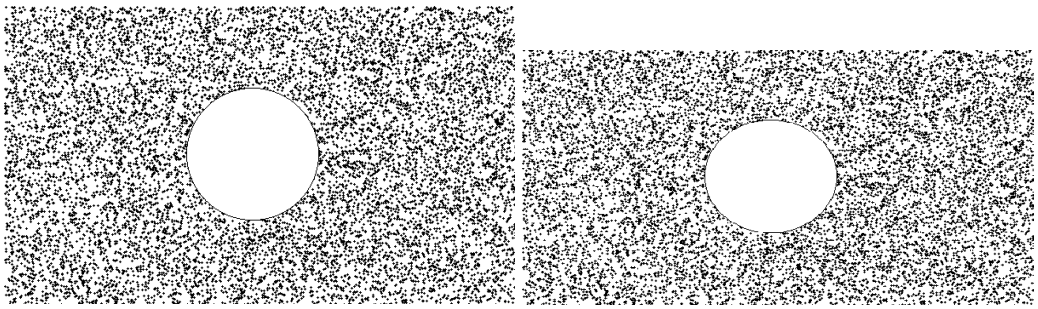


Figura 3.1: Image Correlation: confronto di due immagini.

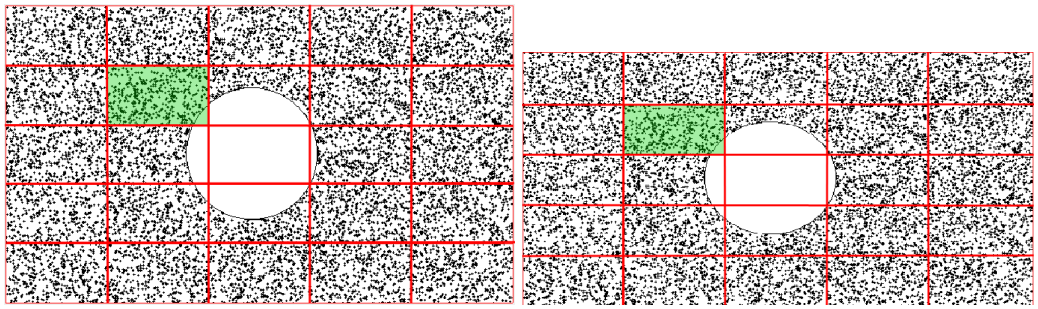


Figura 3.2: Image Correlation: ricerca del medesimo campione in due immagini.

3.1 Un esempio applicativo

Ad esempio, vediamo come funzionerebbe la tecnica dell' *Image Correlation* per queste due immagini.

Dividiamo l'immagine originale in 25 campioni. Cerchiamo poi, con gli opportuni metodi matematici, la posizione del campione verde nell'immagine dello stato deformato.

Il risultato della ricerca è la differenza in `pixel` tra la posizione del campione nella prima e nella seconda immagine.

Come nell'esempio, lo spostamento può essere prodotto sia da moti rigidi (traslazione e rotazione) sia da un'eventuale deformazione.

3.2 Cenni storici

L'idea base dell'*Image Correlation* è nota fin da tempo. Per le prime implementazioni al calcolatore dobbiamo aspettare gli anni settanta, quando i computer furono in grado di effettuare in tempi brevi i calcoli algebrici richiesti.

Inoltre, man mano che le capacità RAM dei calcolatori aumentavano, il metodo è stato implementato in modo sempre migliore: maggiore memoria portava con sé maggiori dimensioni delle immagini elaborate, e quindi migliore accuratezza.

Le prime applicazioni alla meccanica (per il calcolo delle deformazioni) sono state sviluppate negli anni ottanta. Da allora, il metodo si è diffuso ed oggi una delle applicazioni più importanti della *Image Correlation* è sotto le mani di tutti: la misura dello spostamento nei *mouse* ottici.

3.3 Limiti del metodo

Elenchiamo nel seguito le caratteristiche positive per le quali è utile scegliere questa tecnica e gli eventuali difetti cui dovremo far fronte.

3.3.1 Caratteristiche positive. Di questa categoria fa parte la velocità e semplicità del metodo. Impostato un adeguato apparato fotografico, sarà sufficiente trasferire le immagini acquisite ad un calcolatore per essere pronti all'analisi. Il confronto tra due immagini di 5 `MPixel` impiega dai 10 ai 20 minuti su un recente calcolatore.

Il metodo è matematicamente ben fondato: il confronto tra campioni è ottenuto mediante metodi statistici che forniscono una stima dell'accuratezza dei risultati. Con l'*Image Correlation* il confronto tra due immagini è sempre possibile e saranno accettati solo i risultati con accuratezza maggiore di un valore soglia.

3.3.2 Caratteristiche negative. Tra le note dolenti, la *Image Correlation* è sensibile alle diverse condizioni di illuminazione delle immagini. Il medesimo provino, fotografato con luci provenienti da diverse angolazioni o con diverse intensità, potrà non essere riconosciuto.

Se il valore soglia non è scelto accuratamente, la probabilità di falsi positivi è molto alta. La tecnica è sensibile alla scala dell'immagine: il campione deve avere la stessa dimensione sia nell'immagine di stato indeformato che nell'immagine di stato deformato.

Infine, il problema principale della *Image Correlation* è la necessità di `texture` casuali sulle superfici da analizzare. Questo rende necessario l'applicazione di vernici con trame casuali o l'utilizzo di materiali con colorazione molto variabile.

4 Object Detection

L'*Object Detection* è una branca della *Computer Vision*. Permette di determinare la presenza di un determinato oggetto all'interno di un'immagine, calcolandone la posizione.

Il riconoscimento, mediante la vista, degli oggetti è un compito banale e scontato per gli esseri umani. Questo non è affatto vero per i calcolatori: fino a qualche decade fa la potenza di calcolo richiesta non era neanche alla portata dell'*hardware* disponibile.

Lo stato dell'arte, ci consegna oggi una piccola rivoluzione: la formulazione di nuovi algoritmi (il concetto di *Cascade Classifier*, di *Feature detection*, di *Haar Waves*) e dell'odierna potenza di calcolo permettono, per la prima volta, il riconoscimento in tempo reale degli oggetti da parte del calcolatore.

L'obiettivo della *Object Detection* è quello di fornire la posizione, in `pixel`, di un determinato oggetto in un'immagine. Questo, collegato alla possibilità di registrazione video, permette il calcolo di posizione, velocità ed accelerazione dell'oggetto e, quindi, di studiarne l'evoluzione nel tempo.

4.1 Riconoscimento basato su *Feature*

L'algoritmo è basato sul concetto di *Feature*. La posizione dell'oggetto, all'interno di un'immagine, è identificata in base alle caratteristiche (*features*) della classe di oggetti cui appartiene. Queste caratteristiche sono invarianti alla posa o all'orientamento.

In [2] è presentato, per la prima volta, l'approccio moderno alla *Object Detection*. La tecnica calcola una rappresentazione in *onde di Haar* di una classe di oggetti. L'insieme delle *onde di Haar* determinano le *caratteristiche* (o *features*) della classe di oggetti.

In tal modo è eliminato il problema connesso alla variabilità soggettiva all'interno della classe ed è possibile ricavare la rappresentazione della classe automaticamente: l'algoritmo *impara* a riconoscere l'oggetto, dati un numero sufficiente di immagini campione.

L'algoritmo prevede, quindi, alcuni passi fondamentali:

1. Raccolta delle immagini campione. Questi campioni serviranno ad identificare le caratteristiche comuni a tutti i membri della classe di oggetti;
2. Training. Questa fase costituisce il cuore della *Object Detection* durante la quale viene calcolata la rappresentazione in *onde di Haar* della classe di oggetti partendo dalle immagini campione;
3. Utilizzo. Le informazioni prodotte durante il Training sono sufficienti ad identificare, data un'immagine, qualunque oggetto appartenente alla classe.

4.2 Raccolta dei campioni

I campioni necessari per individuare le caratteristiche di una classe sono divisi in due categorie:

- Positivi. In questi campioni un oggetto da individuare è per definizione sempre presente. Viene indicata la posizione di tale oggetto al calcolatore, mediante le coordinate in `pixel` dei vertici del rettangolo circoscritto;
- Negativi. Nei campioni negativi nessun oggetto da individuare è, per definizione, presente.

I campioni positivi e negativi serviranno all'algoritmo per capire quali sono le caratteristiche comuni della classe di oggetti (ovvero, il calcolatore troverà le caratteristiche comuni a tutti i campioni positivi).

Infine, tutti i campioni sono trasformati in scala di grigi e normalizzati per l'intensità della luminosità.

4.3

Onde di Haar

Per il calcolatore, le caratteristiche descrittive della classe sono espresse in termini di distribuzione bidimensionale di intensità luminosa. L'ipotesi base è che la distribuzione di intensità luminosa possa essere scomposta in una sovrapposizione di onde di Haar di diversa frequenza.

Le *onde* sono particolari funzioni matematiche proposte per la prima volta dal matematico Alfred Haar nel 1909. La forma più semplice, riportata qui a titolo d'esempio, è la seguente:

$$\phi(t) = \begin{cases} 1 & 0 < t < \frac{1}{2} \\ -1 & \frac{1}{2} < t < 1 \\ 0 & altrimenti \end{cases} \tag{4.1}$$

Il cui grafico è dato nella Figura 4.1.

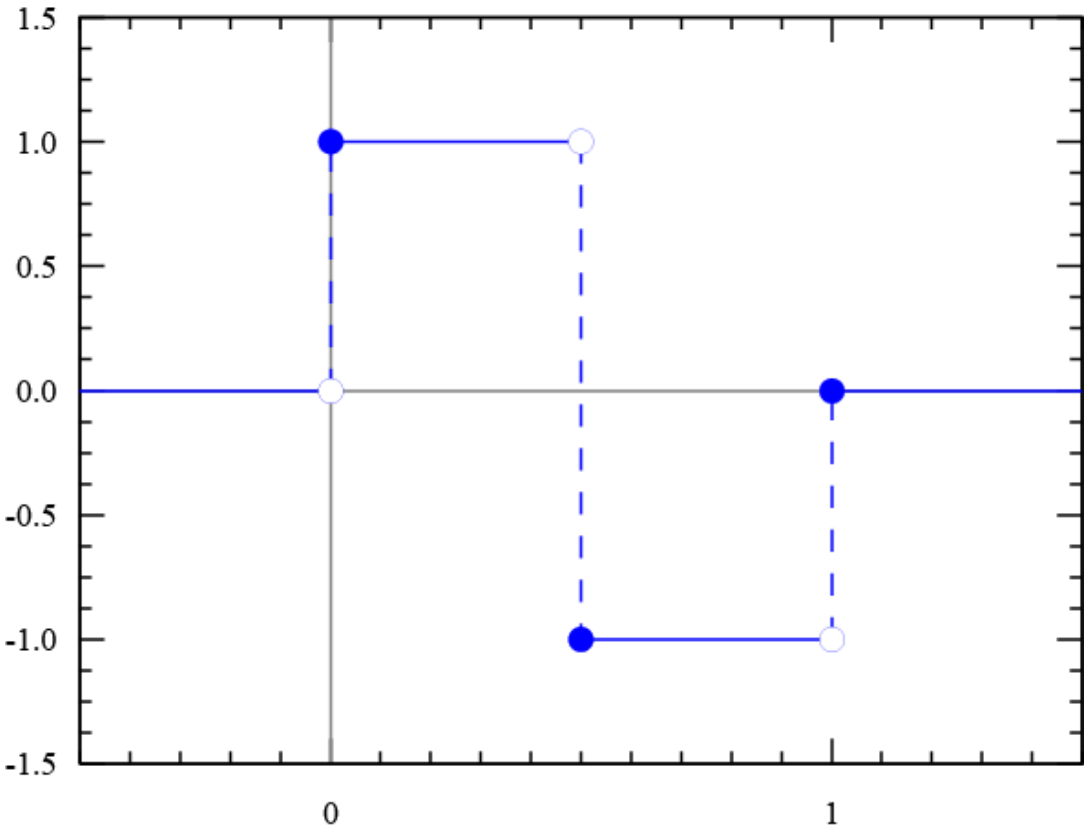


Figura 4.1: Forma più semplice dell'onda di Haar.

Un'immagine (ad esempio un campione) viene diviso in un certo numero di righe e

di colonne. Utilizzando questa suddivisione, vengono calcolati i coefficienti delle *onde di Haar* bidimensionali. Un'eventuale salto netto di intensità luminosa tra due zone vicine è segnalato mediante i valori dei coefficienti: essi saranno molto differenti tra le due zone.

4.4 Training

La fase del *training* permette di identificare le caratteristiche comuni a tutti i campioni positivi della classe, escludendo le caratteristiche presenti anche nei campioni negativi.

Questa fase è divisa in due parti.

4.4.1 Stage 1. Nella prima fase si calcolano i coefficienti delle *onde di Haar* per tutti i campioni positivi, suddividendoli nello stesso numero di righe e colonne. Tali coefficienti sono normalizzati rispetto alla media.

Una caratteristica comune all'intera classe (ad esempio un bordo) viene identificata da coefficienti molto più alti della media a causa del passaggio tra due zone con differenti luminosità.

Zone poco significative (ad esempio lo sfondo) presenteranno coefficienti nella media.

4.4.2 Stage 2. Nella fase successiva vengono calcolate le relazioni matematiche tra le funzioni base dei diversi campioni positivi. In questa fase l'algoritmo utilizza l'insieme dei campioni negativi: un rilevamento costituirà un falso positivo, e permetterà di modificare i coefficienti delle onde di Haar in modo da escludere il campione negativo.

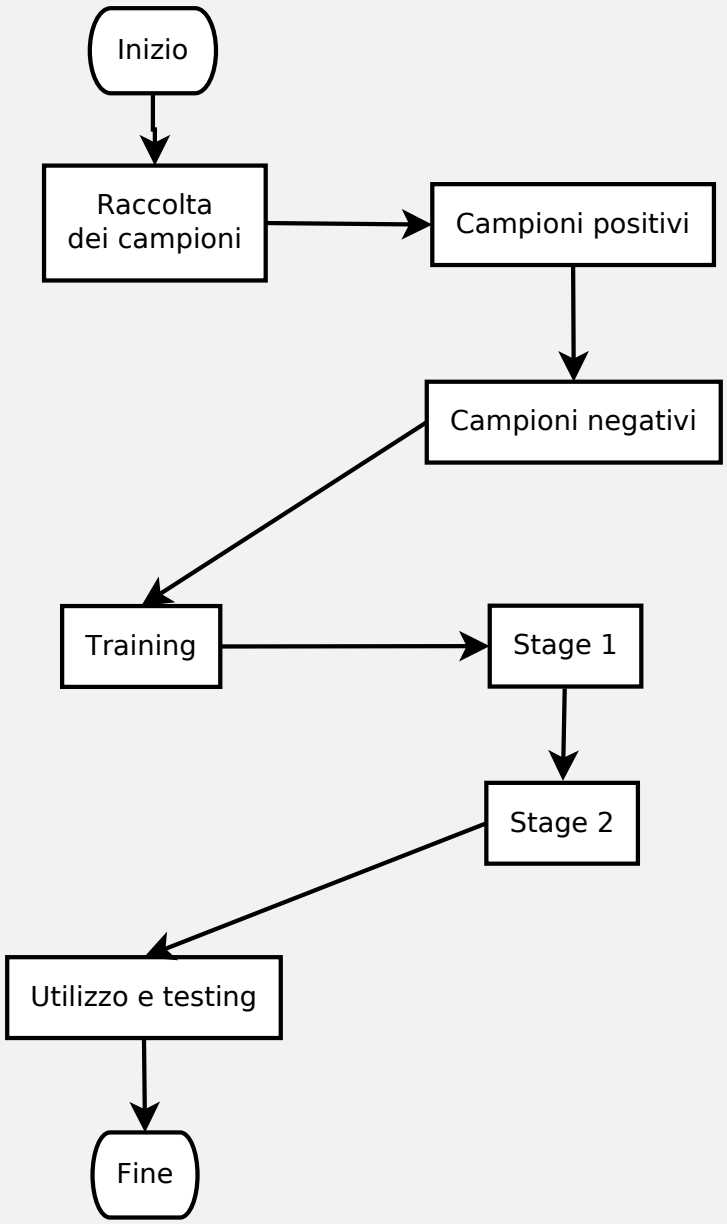
4.5 Un esempio applicativo

La tecnica *Object Detection* è da sempre collegata con il riconoscimento dei volti. Fino a pochi anni fa, il riconoscimento dei volti in tempo reale era impensabile.

La tecnica descritta in questo capitolo ha permesso, per la prima volta nella storia dell'informatica, di realizzare un identificatore facciale semplice e veloce.

In [3] il metodo è stato testato per il riconoscimento facciale utilizzando in *real-time* una videocamera a 15 FPS su un calcolatore funzionante a 700 MHz. Il campione di positivi è costituito da ben 133 immagini contenenti 577 volti.

Diagramma 4.2: Object Detector



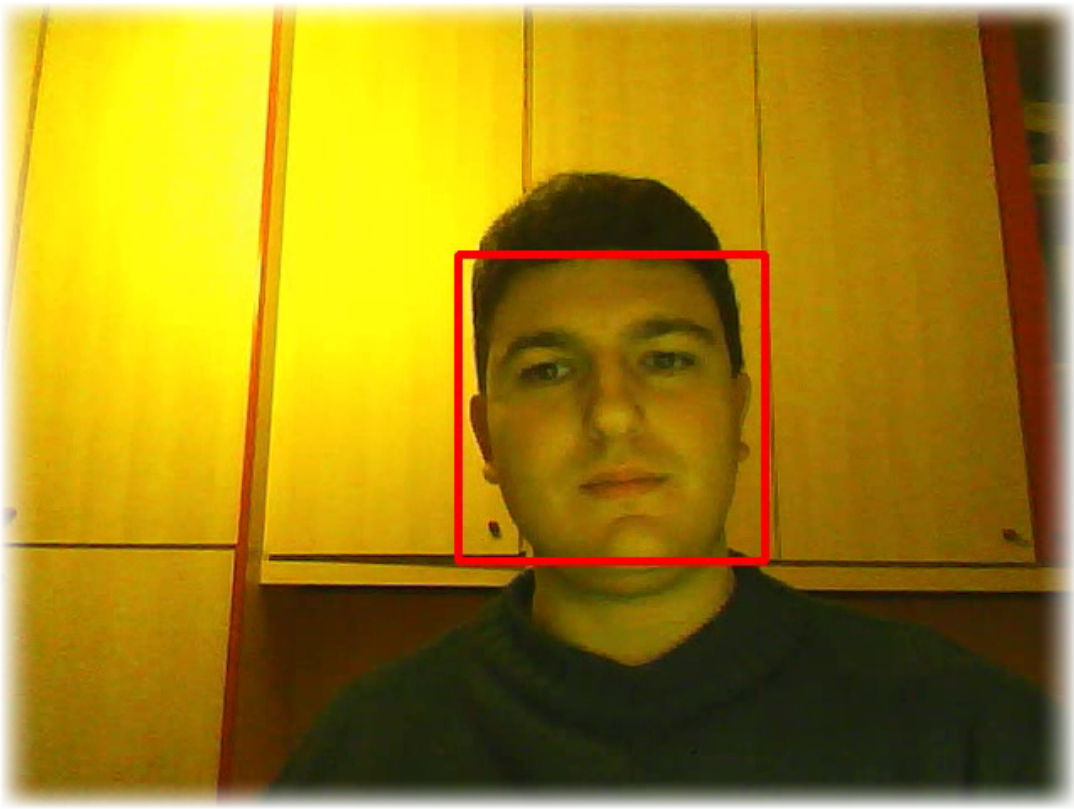


Figura 4.3: Object Detection. Riconoscimento dei volti.

Il risultato è visibile in Figura 4.3.

4.6 Limiti del metodo

L'*Object Detection* può calcolare gli spostamenti e le velocità in determinati punti nodali.

Dopo un opportuno training il calcolatore sarà in grado di riconoscere determinati punti di interesse. Mediante la videoregistrazione il calcolatore potrà calcolarne gli spostamenti, la velocità e l'accelerazione di tali punti.

Inoltre, la tecnica può controllare anche più punti, simultaneamente, in una sola inquadratura.

4.6.1 Caratteristiche positive. Tra le note positive, bisogna elencare l'insensibilità del metodo alle condizioni di illuminazione, allo sfondo e alle condizioni meteorologiche.

Con una buona raccolta dei campioni, i falsi positivi sono rari. La distanza oggetto-fotocamera non è importante: il metodo riconosce un oggetto in base alla sua distribuzione cromatica.

Il training va effettuato una sola volta.

4.6.2 Caratteristiche negative. É necessaria una lunga fase di raccolta dei campioni e di training. Questa fase deve essere ripetuta solo una volta, per il calcolo delle caratteristiche.

La matematica alla base del metodo è molto complessa, l'algoritmo è molto recente ed innovativo. Questo porta a buoni risultati ma anche a situazioni di difficile interpretazione.

La fase di training deve essere ripetuta per ogni classe e, se necessario, affinata per ogni oggetto in particolare.

5 Edge Detector

La tecnica dell'*Edge Detector* serve a determinare i segmenti rettilinei all'interno di un'immagine, calcolare le coordinate dei nodi (punti di inizio e fine linea).

Può essere utilizzata per determinare soltanto le linee più importanti in un'immagine. Ad esempio, per una fotografia di una struttura reale, la tecnica fornirebbe i segmenti corrispondenti ai pilatri e all'architrave.

Questa tecnica è frutto della sinergia di due algoritmi: il *Canny Detector* e le *linee di Hough*. Il primo, ha il compito di individuare i bordi presenti nell'immagine. Viene generata un'immagine in bianco e nero, sulla quale sono presenti solo i bordi. Il secondo ha il compito di ricercare i tratti rettilinei e di calcolarne i nodi.

5.1 Canny Detector

Il *Canny Detector* prende il nome da J. Canny il quale pubblicò, nel 1986, l'algoritmo in [4]. L'obiettivo era di ideare un rilevatore di bordi molto efficiente. I risultati ottenuti presentano una percentuale dei falsi allarmi molto bassa, così come una bassissima percentuale di bordi non rilevati.

L'immagine, trattata con il *Canny Detector*, è in bianco e nero. Basandosi sulla distribuzione di intensità luminosa, un bordo è identificato con un salto di intensità e rappresentato in bianco su sfondo nero. Lo spessore del bordo e la precisione con cui è identificato dipendono dalla scomposizione in zone effettuata dal *Canny Detector*.

In Figura 5.1 e 5.2 possiamo notare un'immagine rispettivamente prima e dopo l'applicazione dell'algoritmo.

Figura 5.1: Edge Detector. Immagine prima dell'applicazione dell'algoritmo *Canny Detector*

Il risultato può essere modulato mediante opportuni parametri. É possibile scegliere la soglia di probabilità: un brusco cambiamento di intensità luminosa è considerato un bordo solo se la variazione è maggiore di una determinata soglia.

5.2

Linee di Hough

L'immagine ottenuta dal *Canny Detector* permette di ricercare solo le linee rette. Questa operazione avviene con una trasformazione dal dominio di R^2 allo *spazio di Hough*.

Nello *spazio di Hough* è possibile ricercare la presenza di qualunque curva, data la sua espressione matematica. Una linea retta è identificata mediante la sua perpendicolare. Si identifica il raggio vettore perpendicolare alla retta data. La distanza ρ dall'origine e l'angolo θ di tale raggio vettore e forniscono i parametri identificativi nello spazio di Hough.

Matematicamente una retta, nello spazio di Hough è data dall'Equazione 5.1.

Figura 5.2: Edge Detector. Immagine dopo l'applicazione dell'algoritmo *Canny Detector*

$$\rho = x\cos(\theta) + y\sin(\theta) \quad \begin{cases} 0 < \rho < D \\ 0 < \theta < \pi \end{cases} \tag{5.1}$$

In Figura 5.3 è possibile vedere l'insieme (univoco) dei due parametri ρ e θ che identificano una retta.

Nel piano cartesiano per ogni punto, ovvero per ogni `pixel`, passa una stella di rette. Con la trasformazione nel piano di Hough, questa condizione viene tradotta mediante l'equazione 5.1 in una curva sinusoidale unica per quel dato punto.

Per ogni `pixel` di un bordo (`pixel` identificati con il colore bianco dal *Canny Detector*) è possibile ottenere l'espressione sinusoidale delle rette passanti per quel punto. Si calcolano, per ogni `pixel`, tutti i valori di ρ con $\theta \in [0, \pi]$.

In una matrice vengono sommati i corrispettivi valori ρ per il medesimo θ. Ogni elemento della matrice indica, quantitativamente, quanti punti giacciono sulla retta corrispondente a quei parametri (θ, ρ): statisticamente, più è grande il valore dell'elemento della matrice più è probabile che tale retta esista, e sia importante, nella struttura reale.

Come nel caso del *Canny Detector* anche per le *Linee di Hough* è possibile modificare la sensibilità del metodo agendo su opportuni parametri riassunti nella Tabella 5.1.

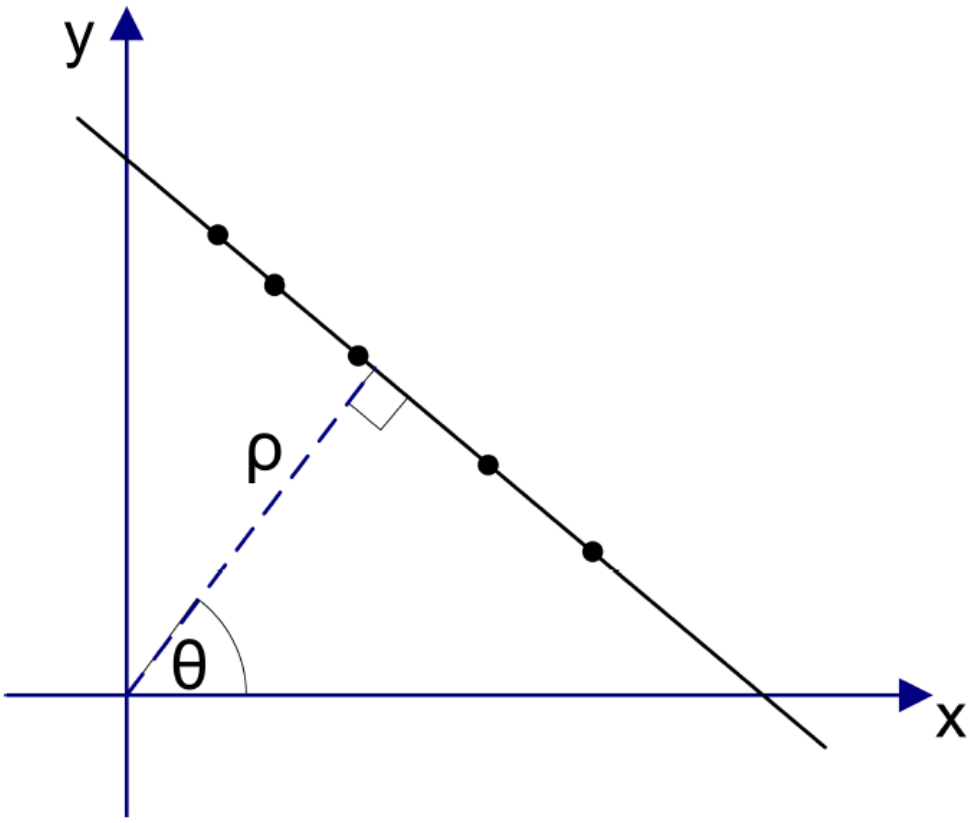


Figura 5.3: Spazio di Hough. Identificazione di una retta mediante la coppia (ρ, θ).

Il risultato dell'applicazione dell'algoritmo alla Figura 5.2 è visibile nella Figura 5.4.

5.3 Limiti del metodo

La tecnica della *Edge Detector* può essere utilizzata per l'identificazione, su base probabilistica, delle linee fondamentali delle geometrie reali. Per strutture reticolari, costituite quindi solo da segmenti, questo implica il ricavare la geometria basandosi su immagini fotografiche.

Come per ogni tecnica, riportiamo le caratteristiche positive e negative.

5.3.1 Caratteristiche positive. Il metodo è molto veloce, addirittura è possibile utilizzarlo in *real-time* durante le acquisizioni video. Calcolate le equazioni delle rette, è possibile ottenere i nodi ed identificarne i più importanti.

5.3.2 Caratteristiche negative. Non esistono dei metodi automatici per la modulazione dei parametri in Tabella 5.1. É necessario effettuare una serie di prove e confrontarne

Parametro	Effetto
Risoluzione spaziale	Aumenta o diminuisce l'errore su ρ
Risoluzione angolare	Aumenta o diminuisce l'errore su θ
Threshold	Soglia probabilistica. Una linea è tale solo se la sua probabilità è superiore alla soglia
Lunghezza minima	Segmenti più corti della distanza minima sono ignorati
Massima distanza	Linee che distano meno della massima distanza sono fuse insieme

Tabella 5.1: Line di Hough. Parametri che influenzano il funzionamento dell'algoritmo.

i risultati.

La tecnica presentata è efficace solo per strutture bidimensionali, e non ottiene risultati significativi per strutture riprese in prospettiva.

Può sovrastimare il numero di nodi e segmenti necessari a descrivere la struttura.

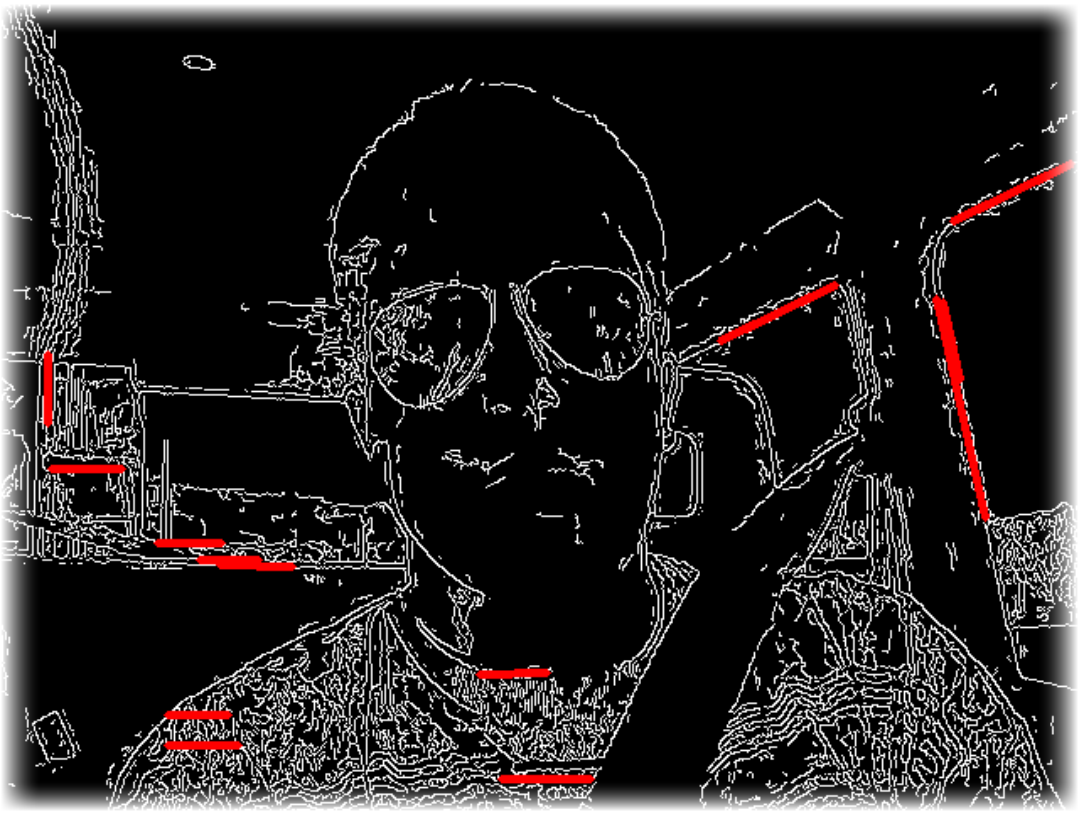


Figura 5.4: Linee di Houg. Risultato dell'applicazione dell'algoritmo alla Figura 5.2.

Parte II

Implementazione

6 La struttura del programma

Nel capitolo precedente sono state descritte le tecniche basilari della *Digital Image Processing*. La formulazione è stata pressoché teorica ed in particolare sono stati diagrammati gli algoritmi per la corretta applicazione delle metodologie.

L'obiettivo di questa Tesi consiste nell'implementare queste tecniche, possibilmente migliorarne alcuni aspetti, ed applicarle al campo strutturale.

Fin ora, l'utilizzo di questi algoritmi era difficile. In primo luogo era necessario scriverne il codice, oppure utilizzarne uno scritto da altri. Solitamente, il *software* scritto da terzi era presentato sotto forma di applicazione senza interfaccia grafica, poiché indirizzato ad un pubblico di esperti informatici.

L'obiettivo di applicare queste tecniche all'analisi strutturale al di là della *Computer Vision*, richiede di semplificare alcuni passaggi.

Innanzitutto, non è possibile gestire la mole di dati necessari all'analisi visuale e strutturale utilizzando i metodi fin ora usati dagli informatici. In ausilio al *software* è necessario un sistema automatico, o almeno semi-automatico, per la gestione di video ed immagini.

6.1 Un punto di vista generale

I campi della *Computer Vision* e dell'ingegneria strutturale sembrano diversi e lontani. Invece, è possibile impiegare la *Digital Image Processing* per stimare gli spostamenti o le deformazioni strutturali.

Queste operazioni sono effettuate, e tutt'ora si effettuano, nei laboratori di tutto il mondo. Tuttavia, le metodologie sono un'evoluzione di quelle utilizzate fin dal principio: si utilizzano estensimetri, accelerometri, e sensori in generale.

Questi sistemi sono efficaci e robusti, ma presentano dei difetti. La stima degli spostamenti è effettuata solo in determinati punti critici (dove sono applicati i sensori) e le misure possono avvenire solo sotto sorveglianza (ad esempio in un laboratorio o in fase di collaudo). Infine, il metodo è invasivo.

L'elaborazione al calcolatore delle immagini digitali, è invece un metodo non invasivo, economico e veloce. L'apparato necessario consiste solo in una videocamera o fotocamera. Si apre la possibilità di calcolare velocità di funzionamento per organi in movimento, anche durante il loro utilizzo.

Collegato al campo strettamente aeronautico, è possibile calcolare le velocità di decollo ed atterraggio degli aeromobili riprendendoli da lontano.

In generale, si può dividere il campo di applicazioni in due rami. Da un lato, la misura degli spostamenti in un numero discreto di punti. In questi casi, gli spostamenti si presentano di grande entità. Dall'altro lato la stima dell'intero campo di spostamenti: in questo caso non è possibile individuare un numero discreto di punti da monitorare, ma l'obiettivo è il campo di deformazione di un'intera zona. Generalmente, nel secondo caso, gli spostamenti (o le deformazioni) sono di lieve entità.

Discorso a parte, è il riconoscimento delle geometrie strutturali: questo campo è più innovativo ed è già un buon risultato riconoscere correttamente la geometria di una struttura travi-forme bidimensionale.

6.2 Specifiche del *software*

Dall'analisi condotta al punto precedente, il *software* dovrà rispettare le seguenti specifiche:

- Riconoscere un punto o una serie di punti (in numero discreto) e calcolarne spostamenti, velocità, accelerazione di grande entità rispetto alla lunghezza caratteristica del problema;
- Calcolare il campo di spostamenti, ad esempio in una piastra bidimensionale, anche se con lieve entità rispetto alla lunghezza caratteristica del problema;
- Ricavare le linee basilari di una struttura reticolare bidimensionale.

6.3 Scelta del linguaggio di programmazione

Nell'ambito di Tesi, il linguaggio di programmazione con cui implementare queste tecniche, deve:

- Supportare l'implementazione delle funzioni di *Digital Image Processing*;
- Effettuare le operazioni per l'algebra lineare in modo veloce ed impeccabile;
- Gestire i grafici dei dati;
- Possibilità di realizzare interfacce grafiche GUI;

Oltre questi requisiti, alcune caratteristiche desiderabili sono:

- Gestione semplificata del codice (in caso di molteplici modifiche);
- Facilità di importazione ed esportazione dei dati in formato CSV, AutoCAD, Nastran;
- Programmazione Multithreading e Multiprocessing;
- Caratteristica Cross-Platform.

I linguaggi tradizionali della programmazione scientifica, quali il FORTRAN e il MATLAB, presentano delle limitazioni. Questi linguaggi permettono di effettuare le operazioni dell'algebra lineare in modo impeccabile e veloce. Il linguaggio MatLab, in particolare, risponde ai requisiti necessari. Tuttavia, la realizzazione di interfacce grafiche in linguaggio MatLab è particolarmente complessa, così come la gestione del codice risulta caotica. La necessità di dover acquistare l'ambiente MatLab ad un costo non indifferente, costituisce un vincolo importante.

Per ovviare a queste problematiche, la comunità scientifica sta adottando da qualche anno il linguaggio Python[1].

Tra le caratteristiche salienti del linguaggio, c'è sicuramente la semplicità, la programmazione ad oggetti e la modularità del codice. L'approccio modulare permette di estendere le capacità del linguaggio all'infinito: ogni modulo si occuperà di una sola funzionalità del programma.

Seguendo la tendenza attuale all'utilizzo di questi nuovi strumenti, sarà adottato anche in questo lavoro il linguaggio di programmazione Python.

[1] Ulteriori informazioni su Python sono disponibili sul sito `www.python.org` ed in italiano `www.python.it`.

6.3.1 La programmazione ad oggetti. Una marcia vincente dei moderni linguaggi di programmazione è sicuramente la *programmazione ad oggetti*.

Questa tipologia di programmazione, prevede la gestione di *oggetti software* che interagiscono tra loro scambiandosi dei messaggi. Utilizzando questa tecnica si creano degli oggetti, descrivendone le proprietà della classe di appartenenza. Inoltre è possibile *riutilizzare* il codice in maniera semplice: gli oggetti utilizzano altri oggetti per compiti specifici.

Tra i vantaggi della programmazione orientata agli oggetti, abbiamo quindi:

- Un supporto naturale alla modellazione *software* degli oggetti del mondo reale o del modello astratto da riprodurre;
- Una più facile gestione e manutenzione di progetti di grandi dimensioni;
- L'organizzazione del codice sotto forma di classi. Questo favorisce la modularità e il riuso del codice.

6.3.2 La divisione in moduli. Le gestione dei moduli in Python è semplice ed immediata. La divisione in moduli porta alla realizzazione di una *libreria* di funzioni comuni all'intero programma.

Ad esempio, l'acquisizione video è comune a tutte le tecniche della *Digital Image Processing*, per cui è opportuno raggruppare tutte le funzioni e tutte le *classi di oggetti* che si occupano dell'acquisizione video nel medesimo modulo. Ogni parte del programma a cui servirà acquisire un video, utilizzerà tale modulo.

La modularità del linguaggio Python è una caratteristica potente. Grazie al sempre più ampio utilizzo nell'ambito scientifico, molti moduli sono già stati scritti da qualcun altro: basterà solo conoscerne il funzionamento e imparare ad utilizzarli.

L'insieme di più moduli con lo stesso obiettivo è detto, in gergo, *libreria software*. Ogni libreria ha il compito di gestire un aspetto del programma. Come vedremo nel seguito, il *software* realizzato per questa Tesi, adopera librerie molto diversificate tra loro.

Caratteristica	Descrizione
Facilità d'uso	Con il *Qt Designer* è possibile progettare l'interfaccia grafica indipendentemente dal codice
Cross-Platform	Il codice, se correttamente scritto, presenterà il medesimo aspetto sui sistemi Windows, Linux e MacOsX
Modularità	La libreria è suddivisa in moduli: possiamo scegliere solo i moduli di nostro interesse
Moderna	Aggiornata alle ultime tecnologie del settore
Integrazione	Ottima integrazione con il linguaggio Python

Tabella 6.1: Libreria PyQt4 per il GUI-programming. Caratteristiche salienti.

6.4 Le librerie

Il *software* utilizza a pieno la modularità offerta dal linguaggio. Le funzioni sviluppate sono state raggruppate in moduli, per il riuso del codice.

Oltre ai moduli scritti appositamente, è utilizzata una serie di librerie per la gestione dei compiti più disparati: ciascuna apporta il suo contributo al progetto.

6.4.1 L'interfaccia grafica con la libreria Qt. Il *software* è dotato di interfaccia grafica, in gergo GUI (Graphical User Interface). La realizzazione della GUI ha spinto verso la scelta di una libreria appositamente progettata per le interfacce grafiche.

Sono disponibili molteplici librerie per tale scopo, ma la scelta è ricaduta sulle librerie Qt prodotte da Nokia, scegliendone l'implementazione in linguaggio Python, chiamata PyQt4.

I punti di forza di questa libreria sono riassunti nella Tabella 6.1.

In Figura 6.2 è possibile vedere il *Qt Designer* per la progettazione grafica delle interfacce GUI.

6.4.2 L'algebra lineare con NumPy. Con la libreria *NumPy* (Numeric Python) sono stati svolti i calcoli con le matrici, le operazioni di algebra lineare, l'analisi in frequenza.

Le capacità di calcolo di questa libreria sono simili al linguaggio MATLAB. Inoltre per *NumPy* sono stati sviluppati dei codici per l'importazione e per l'esportazione dei dati in formato CSV (dati condivisibili con Excell).

6.4.3 I grafici con Matplotlib. Un parte importante del lavoro è sicuramente la presentazione dei risultati in forma grafica. Per questo è stata scelta la libreria *Matplotlib*.

Come affermato dagli stessi autori, questa libreria ha l'obiettivo di essere compatibile quanto più possibile con i comandi MATLAB, utilizzando i vettori e le matrici della libreria NumPy.

Un esempio di finestra per il grafico dei risultati è presente in Figura 6.3.

6.4.4 La gestione del *Multithreating*. Le librerie *Multithreating* e *Multiprocessing* sono utilizzate per ridurre i tempi di calcolo. Su calcolatori a più processori, il *software* è in grado di dividere i dati da elaborare ridistribuendone il carico su tutti i processori disponibili.

Le classi `Worker` e `DigitalCorrelation` hanno il compito di elaborare i dati in ingresso utilizzando tutti i processori disponibili.

I dati da elaborare sono sottoposti ad una prima semplificazione. Successivamente, sono divisi in parti eguali ciascuna destinata aad un singolo processore. Il modulo `Multiprocessing` crea un *pool* di processi, ovvero un insieme di compiti da effettuare in parallelo utilizzando i dati già ripartiti.

Ogni compito fa capo ad una funzione la quale svolge i veri e propri calcoli in base ad un insieme di parametri. Il processo originario resta, dunque, in attesa finché non viene terminato il *pool* di processi.

A titolo d'esempio, il codice per la gestione in parallelo di un'analisi di ImageCorrelation è riportato in queste pagine. Oltre al codice Python è riportato, per maggiore chiarezza, il corrispettivo diagramma di flusso per il calcolo parallelo.

Codice 6.1: Codice Multiprocessing

```
1  def analysingProcess(arg):
     deformedImg, undeformed, threshold, contour, method, rects = arg
3
     if len(rects) <= 0:
5      return []

7    uImg = cv.LoadImage(undeformed)
     dImg = cv.LoadImage(deformedImg)
9
     wD, hD = cv.GetSize(dImg)
11
     values = []
13
     for (x, y, wT, hT) in rects:
```

```
    template = cv.GetSubRect(uImg, (x, y, wT, hT))

    result = cv.CreateImage((wD - wT + 1, hD - hT + 1), 32, 1)
    cv.MatchTemplate(dImg, template, result, method)
    minVal,maxVal,minLoc,maxLoc = cv.MinMaxLoc(result)

    if math.fabs(maxVal- minVal) >= threshold:
      newx = maxLoc[0]
      newy = maxLoc[1]

      if contour == 0:
        val = math.fabs(maxVal- minVal) * 100.0
      elif (contour == 1) or (contour >= 4):
        val = math.sqrt( (x - newx)**2 + (y - newy)**2 )
      elif contour == 2:
        val = newx - x
      elif contour == 3:
        val = -(newy - y)

      values.append((x, y, wT, hT, newx, newy, val))

  return values


class DigitalImageCorrelation:

  methodList = [cv.CV_TM_SQDIFF, cv.CV_TM_SQDIFF_NORMED,
        cv.CV_TM_CCORR, cv.CV_TM_CCORR_NORMED, cv.CV_TM_CCOEFF]


    def Correlate(self, undeformed, deformed, W, H, method=3,
    contour=1,
    minWidthPercentValue=0.00):

    uImg = cv.LoadImage(undeformed)
    wU, hU = cv.GetSize(uImg)

    rectsGroup = []
    index = 0
    valList = []

    beg_time = time.time()

    nHorizWnd = int(int(wU) / int(W))
    nVertWnd = int(int(hU) / int(H))

    for i in range(0, self.thGroup):
      rectsGroup.append([])

    if wU % W > 0:
      nHorizWnd += 1

    if hU % H > 0:
      nVertWnd += 1

    for i in range(0, nHorizWnd):
      wT = W
      hT = H

      x = i * wT
      if (x + wT) > wU:
        wT = wU - x

      rects = []

      for j in range(0, nVertWnd):
        y = j * hT
        if (y + hT) > hU:
          hT = hU - y

        rectsGroup[index].append((x, y, wT, hT))
        index += 1

        if index >= self.thGroup:
          index = 0

    del uImg

    args = []
    for rects in rectsGroup:
```

```
95        args.append((deformed, undeformed, self.threshold,
            contour, self.methodList[method], rects))
97
      pool = multiprocessing.Pool(self.thGroup)
99    res = pool.map_async(analysingProcess, args)
101
      for item in res.get():
103     valList = valList + item
105   del pool
107
      max_val = 0
109   for (x, y, wT, hT, newx, newy, val) in valList:
        if math.fabs(val) > max_val:
111       max_val = math.fabs(val)
113   colored_box = []
      for (x, y, wT, hT, newx, newy, val) in valList:
115     if (math.fabs(float(val) / wU) < minWidthPercentValue)
          and (contour > 0):
117
          continue
119
        box = (x, y, newx, newy, wT, hT, val, max_val)
121     colored_box.append(box)
123
      self.sendMsg('Done. Taken ' + self.timeToString(time.time()
125     - beg_time))
127   return colored_box
```

6.5 La libreria *OpenCV* per la *Computer Vision*

Per quanto riguarda le tecniche base della *Computer Vision*, al giorno d'oggi è impensabile, ed inutile, realizzarne un'implementazione a partire da zero. Esistono infatti, svariate librerie *open source* che mettono a disposizione il proprio codice per eventuali utilizzi e modifiche.

La libreria necessaria al lavoro di Tesi fornisce un'interfaccia di programmazione semplice ed orientata agli oggetti. Inoltre implementa le più recenti tecnologie a disposizione.

6.5.1 Stato dell'arte. Ad oggi, la *Computer Vision* si sta sviluppando maggiormente per i dispositivi ad alta tecnologia, quali smartphone e tablet. Per questo motivo, la libreria di *Computer Vision* più all'avanguardia è `OpenCV`[2].

La libreria `OpenCV` è stata inizialmente scritta dalla Intel. Il codice sorgente è disponibile gratuitamente in rete, in quanto gode di una natura *open source*. Il linguaggio di programmazione della libreria è il C++, in quanto tale linguaggio è capace di grande velocità di elaborazione. É comunque possibile l'utilizzo di `OpenCV` in Python.

[2] http://www.opencv.willowgarage.com

Questa libreria è stata la prima ad implementare l'algoritmo della *Object Detector* in *real-time*.

6.5.2 Libreria OpenCV. Le tecniche di base descritte nella Parte I sono parzialmente disponibili nella libreria `OpenCV`. La libreria gestisce automaticamente le immagini, l'acquisizione video e le operazioni base per l'applicazione delle tecniche della *Computer Vision*.

Per l'*Object Detector*, è gestita in modo semi-automatico la fase del testing. Le fasi preliminari (raccolta dei campioni e training) sono ancora escluse dall'automatizzazione.

Per l'*Image Correlation*, è possibile cercare un campione all'interno di un'immagine e ottenere come risultato una distribuzione di probabilità. Manca completamente la gestione dell'immagine originale, la sua divisione e l'analisi dei risultati.

Per l'*Edge Detector*, gli algoritmi *Canny Detector* e *Linee di Houg* sono presenti separatamente e non forniscono i nodi di intersezione delle rette.

La libreria è disponibile per tutti i sistemi operativi maggiormente utilizzati (Windows, Macintosh, Linux) oltre che per i dispositivi mobili.

6.6 Il diagramma del *Software*

L'approccio modulare ha influenzato la realizzazione del *software* fin dall'inizio. Le operazioni comuni a tutte le possibili analisi (acquisizione video, interfaccia grafica) sono state inserire in moduli a sé stanti.

Il codice presenta tre moduli fondamentali elencati qui e descritti nel seguito.

6.6.1 Modulo *Experiment*. Con questo nome si indica il modulo del programma che prevede un'acquisizione o un'analisi di video. Un Experiment può essere di tre tipologie: `CAPTURE`, `DETECT`, `EDGE`.

Un Experiment di tipo `CAPTURE` ha il compito di acquisire un video da una webcam o da un *file* e salvarne i fotogrammi come immagini. Al termine verrà chiesto se modificare o catalogare le immagini.

Un Experiment di tipo `EDGE` ha il compito di acquisire un video da una webcam o da un *file* ed analizzarlo con la tecnica *Edge Detector*. Il risultato può essere visualizzato direttamente sullo schermo o registrato in un *file*.

Numero di *file*	34 in tutto (17 di codice, 4 di immagine, 9 file GUI, 4 file di aiuto)
Dimensioni del codice	3376 linee, pari a 88.1 KByte
Finestre di dialogo	9
Funzionalità	Experiment, DeformationFromImage, GeometryFromImage
Tipi di Experiment	3 (espandibili)

Tabella 6.2: Informazioni sul *software*.

Un Experiment di tipo `DETECT` ha il compito di acquisire un video da una webcam o da un *file* e di analizzarlo con l'*Object Detector*. Questo tipo di Experiment è sicuramente più complesso e verrà approfondito nei capitoli successivi.

Ogni Experiment, presenta un *file* di configurazione che spiega al *software* cosa fare e come farlo.

6.6.2 Modulo *Deformation From Image*. Questo modulo calcola il campo di deformazione e/o di spostamento basandosi su due immagini: una dello stato indeformato ed una dello stato deformato.

Questa funzionalità utilizza la tecnica dell'*Image Correlation* e del *Multiprocessing* per l'analisi automatica di due o più immagini.

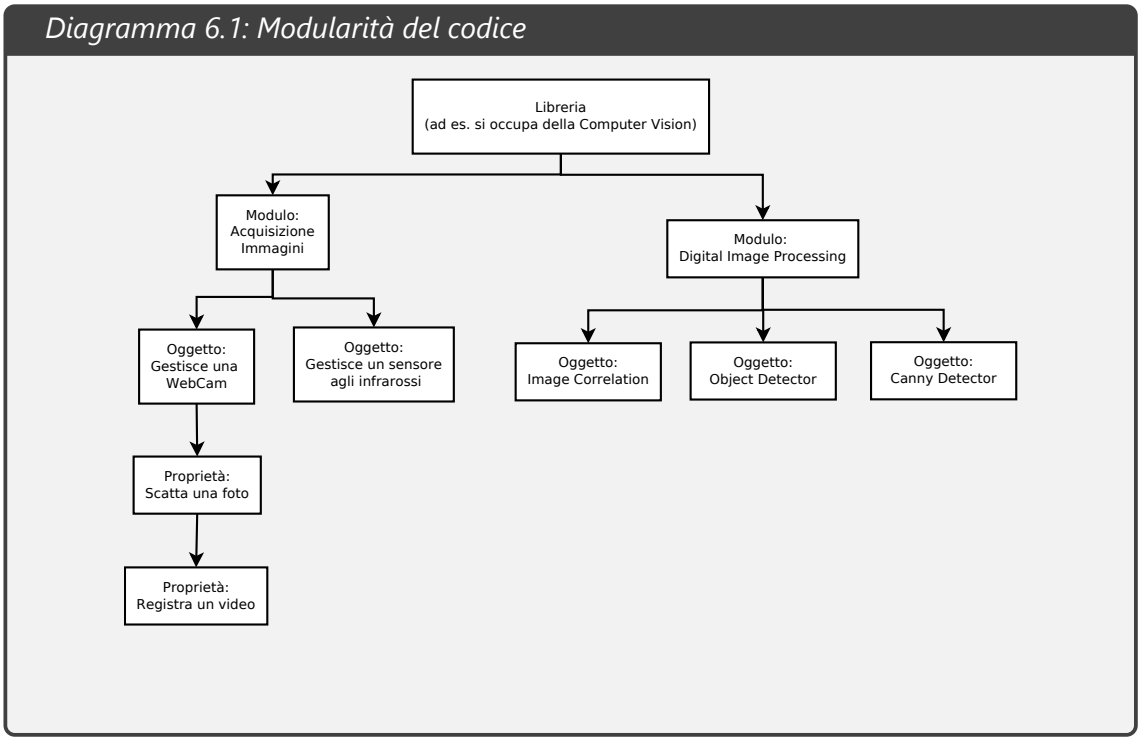
6.6.3 Modulo *Geometry From Image*. Il modulo *Geometry From Image* consente di ricavare la geometria base delle strutture, adoperando un'immagine.

La caratteristica principale, è la possibilità di esportare la geometria in un *file* formato `DXF` per AutoCAD.

6.6.4 Lo schema generale. Lo schema modulare del programma può essere rappresentato mediante un diagramma di flusso. Le librerie esterne (ad es. PyQt4, NumPy, OpenCV) non sono rappresentate per brevità: il loro uso è cospicuo e molto presente.

Nella Tabella 6.2 sono presentate, a titolo di curiosità, alcune informazioni sul *software*.

Diagramma 6.1: Modularità del codice



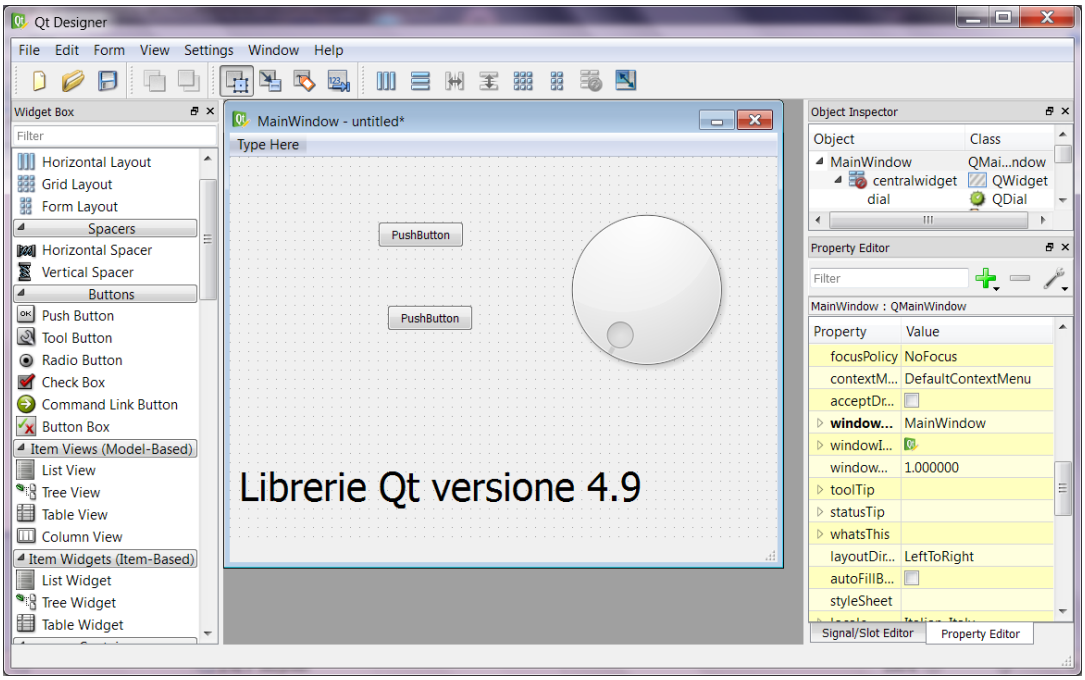


Figura 6.2: Libreria PyQt4. QtDesigner per la realizzazione delle GUI.

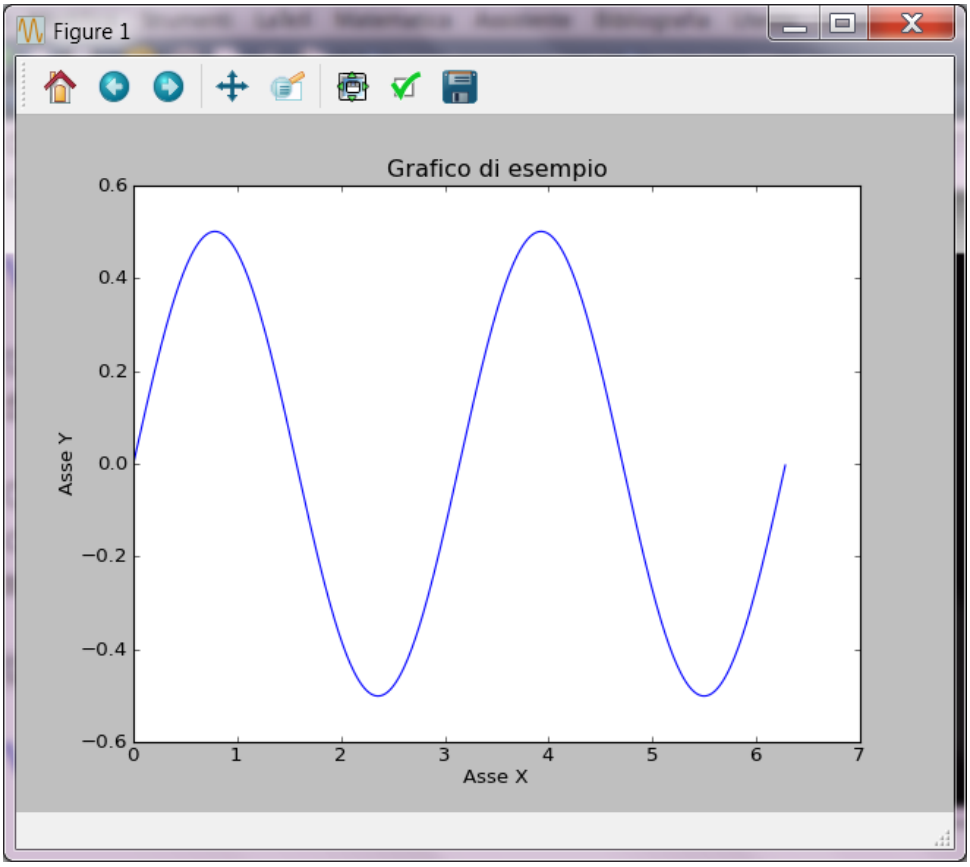


Figura 6.3: Libreria Matplotlib. Finestra dei grafici in stile MATLAB.

Diagramma 6.4: Calcolo parallelo

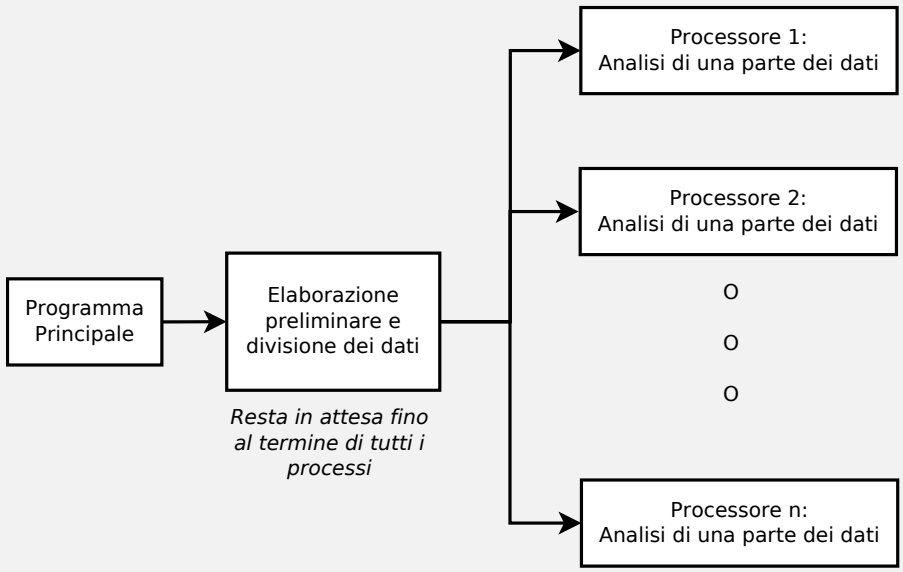
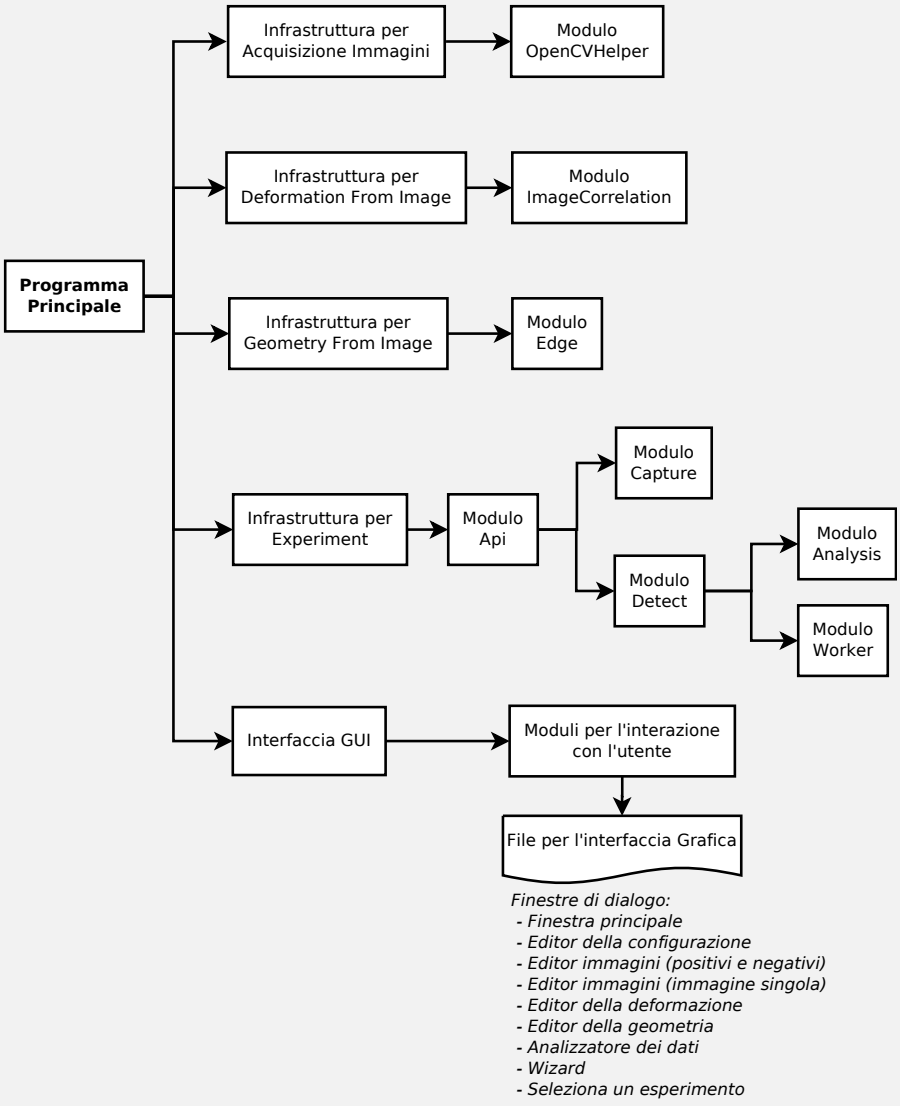


Diagramma 6.5: Struttura del programma



7 Modulo *Experiment*

Il modulo *Experiment* si occupa della gestione, dell'analisi ed elaborazioni dei video. I video possono essere acquisiti direttamente dalla webcam o, in alternativa, da *file* video: in generale da una *sorgente video*.

Il risultato dell'*Experiment* è sempre visibile sullo schermo del calcolatore, indipendentemente dalla tipologia della sorgente.

Poiché, solitamente, le analisi sono costituite da molti *file* immagine, il programma utilizza una cartella predefinita per memorizzare tutte le informazioni relative agli *Experiment*. Questa cartella può essere scelta dall'utente: se non specificata viene utilizzata la cartella di installazione del programma.

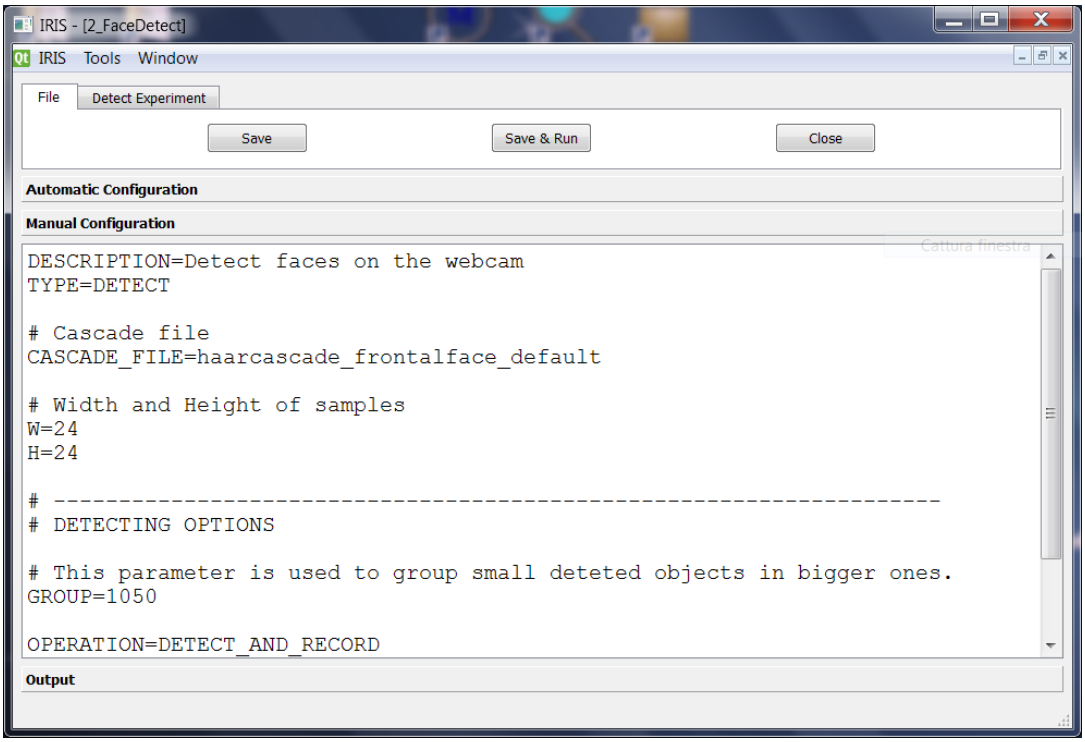


Figura 7.1: Finestra per la gestione degli *Experiment*.

In Figura 7.1 si può vedere la finestra che il programma utilizza per la gestione degli *Experiment*.

In alto le schede raggruppano i comandi più utilizzati, tra cui `Save & Run` che permette di avviare l'analisi.

7.1 Struttura di un esperimento

Un *Experiment* deve avere i seguenti requisiti:

- Tutto il materiale dell'*Experiment* deve essere contenuto in una sola cartella (ovviamente, il materiale può essere suddiviso in diverse sottocartelle);
- Il nome dell'*Experiment* coincide con il nome della cartella che contiene i dati;
- All'interno della cartella, deve essere presente il *file* di configurazione, dal nome predefinito di `configuration.txt`.

La cartella contenente l'*Experiment* può trovarsi in qualsiasi percorso. Come detto in precedenza, la gestione è molto facilitata se la cartella dell'*Experiment* è contenuta nella cartella predefinita.

All'avvio del programma, avviene un primo controllo: tutte le cartelle presenti nella cartella predefinita vengono sottoposte ad una verifica. Se sono rispettate le tre condizioni sopra elencate, l' *Experiment* è aggiunto automaticamente all'elenco delle analisi disponibili.

7.2 Tipo di *Experiment*

Le operazioni da svolgere sulla sorgente video, possono essere svariate. Le funzionalità relative ai video sono state raggruppate in tre *tipologie* di *Experiment*.

La Tabella 7.1 aiuta a decidere quale tipologia si avvicini di più alle esigenze.

7.2.1 La configurazione. Il programma distinguerà la tipologia di *Experiment* dal *file* di configurazione. Questo *file* è costituito da un lungo elenco di parole chiave, seguite da un valore: si potrebbe definire una sorta di dizionario.

Tipo	Cosa possiamo fare?
DETECT	Utilizzare l'*Object Detector*: raccolta dei campioni positivi e negativi, training, testing. Applicare l'*Object Detector* ad una sorgente. Registrazione dei video analizzati con l'*Object Detector*. Analisi del moto (grafici di spostamento, velocità, accelerazione), esportazione dei dati di spostamento in formato CSV.
CAPTURE	Registrazione dei video (ad es. per un'analisi successiva). Salvare ogni fotogramma come immagine.
EDGE	Applicare l'*Edge Detector* ad una sorgente video. *Tuning* dei parametri per l'*Edge Detector*, registrazione dei risultati in un video.

Tabella 7.1: Modulo *Experiment*. Le operazioni per ogni tipologia di *Experiment*.

Esempio di file *di configurazione*

```
DESCRIPTION=Capture some images.
TYPE=CAPTURE

SOURCE=DSCN3692.MOV
FPS=25

# This is a comment
POS_DIR=pos
NEG_DIR=neg
```

A seconda del valore delle parole chiave, il programma comprende le operazioni che deve effettuare.

Alcune parole chiave hanno senso per tutte le tipologie, altre invece sono specifiche.

La parola chiave che sicuramente deve essere presente è TYPE. A seconda del suo valore, determina il tipo di *Experiment* che avrà luogo. Se tale parola non è presente viene generato un messaggio di errore.

L'elenco delle parole chiave condivise tra gli *Experiment*, insieme ad alcuni caratteri speciali, sono riportate nella Tabella 7.2.

In Figura 7.1 possiamo vedere, al centro, l'*Editor della configurazione* ovvero lo spazio (bianco) in cui scrivere e/o modificare il *file* di configurazione.

Parola chiave	Descrizione
`TYPE`	Definisce la tipologia dell*Experiment*. Deve essere sempre presente
`DESCRIPTION`	Una breve descrizione, ad uso dell'utente
`SOURCE`	Indica la sorgente di acquisizione. Può essere il nome di un *file* o il numero ordinale che indica una webcam
`#`	Posto all'inizio della riga, fa ignorare tale riga al programma (commento)

Tabella 7.2: Modulo *Experiment*. Parole chiave condivise.

7.3 Tipologia *DETECT*

Questa tipologia di *Experiment* è sicuramente quella più importante poiché implementa la tecnica *Object Detector*.

L'obiettivo finale è far riconoscere un determinato oggetto (ad esempio un bersaglio) al calcolatore, in modo da poterne calcolare gli spostamenti, la velocità e l'accelerazione.

7.3.1 Raccolta dei campioni. Individuato l'oggetto da utilizzare, bisogna raccogliere i campioni (positivi e negativi). Per questo scopo, usufruiamo di una qualsiasi sorgente video. A titolo di esempio, descriviamo la procedura per l'utilizzo di una webcam.

Per la raccolta dei campioni negativi, dobbiamo aver cura di non far comparire l'oggetto interessato all'interno delle immagini. In maniera analoga, per i campioni positivi dovremo aver cura di presentare l'oggetto nel campo di ripresa. É utile sottoporre al calcolatore l'oggetto interessato sotto varie condizioni di luminosità o punti di vista.

In Figura 7.2 possiamo vedere il programma durante la registrazione: nella finestra in basso a destra viene mostrato il video per controllare la zona inquadrata.

Al termine, il programma chiederà, per ogni fotogramma registrato, di catalogare i campioni negativi, i campioni positivi e i fotogrammi da ignorare. Nel caso di campione positivo, verrà chiesta la parte dell'immagine in cui è presente l'oggetto.

In Figura 7.3 è presente la finestra per la scelta dei fotogrammi. In alto sono presenti i comandi `Mark as Negative` e `Cut And Mark as Positive` che permettono rispetti-

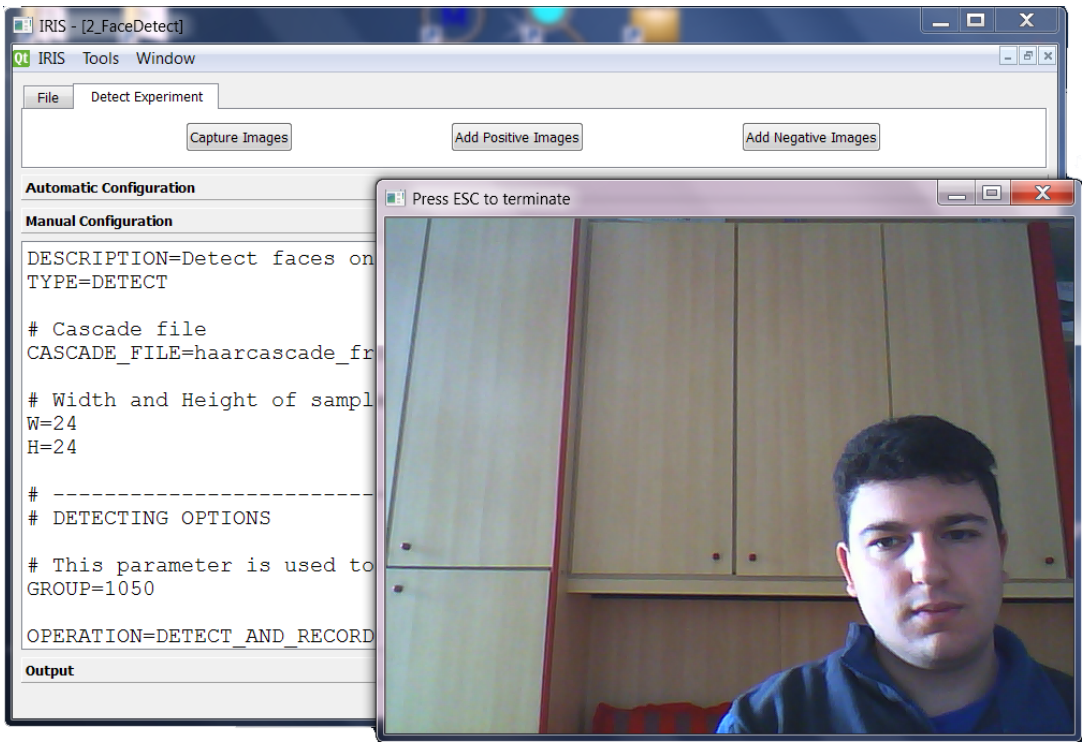


Figura 7.2: *Experiment* tipo `DETECT`. Registrazione di un video per la raccolta dei campioni

vamente di individuare nel fotogramma una campione negativo e un campione positivo (specificando la regione di interesse).

7.3.2 Training. La fase del *training* è sicuramente la meno laboriosa da parte dell'utente, ma la più intensa per il calcolatore. Il processo del training può impiegare anche 10 o 12 ore. Come anticipato, questa fase deve essere compiuta almeno una volta per ogni *Experiment*.

Il risultato di questa fase è un *file* in formato `XML` chiamato in gergo *cascade file*. Per il successivo utilizzo del programma (e quindi per il riconoscimento dell'oggetto in questione) sarà sufficiente disporre di questo file.

Ad esempio, è possibile preparare il *cascade file* con un calcolatore molto potente e poi utilizzarlo su qualsiasi calcolatore (anche meno potente). In questo caso, sul computer *utilizzatore* non sarà necessario effettuare la raccolta dei campioni né il training, ma basterà il *cascade file* per il calcolo di posizione e velocità.

La fase del training inizia automaticamente quando non è presente il *cascade file*. L'*output* di questa fase è visibile nella sezione *Output* dell'*Editor della configurazione*, come mostrato in Figura 7.4.

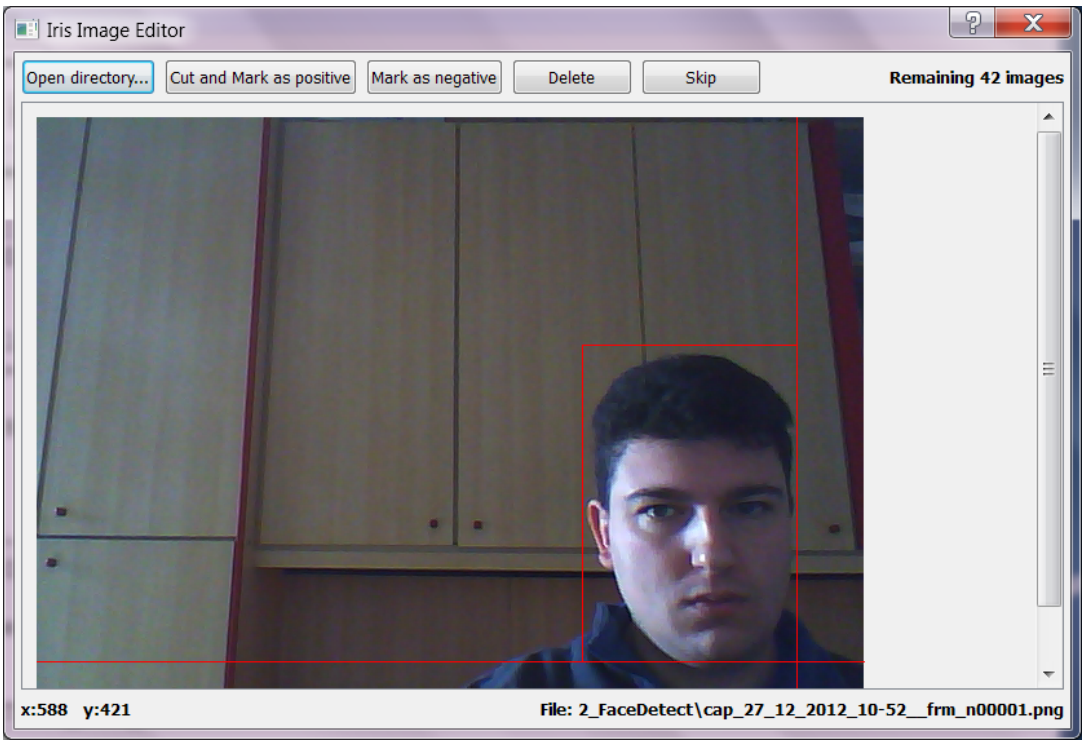


Figura 7.3: *Experiment* tipo `DETECT`. Selezione dei fotogrammi per i campioni negativi e positivi.

7.3.3 Testing. Infine, la fase del *testing* consiste nell'analisi vera e propria. Durante il testing viene individuato l'oggetto e ne viene calcolata la posizione e velocità.

Per il corretto funzionamento di questa fase, è necessaria la presenza del *file* di configurazione e del *cascade file*.

Avviato l'esperimento, se il *cascade file* è presente (perché frutto del *training* o perché copiato da noi nella cartella dell'*Experiment*), il programma avvia la fase del *testing*.

In Figura 7.5 vengono mostrate le finestre del programma durante questa fase.

7.3.4 Il *file* della configurazione. Il cuore di ogni *Experiment*, come detto, consiste nel *file* di configurazione che descrive al programma il da farsi.

Le parole chiave condivise da tutti gli *Experiment* sono state descritte nei paragrafi precedenti. In questa sezione, vedremo in dettaglio le parole chiave per l'*Experiment* di tipo `DETECT`.

`CASCADE_FILE` Questa opzione indica il nome del *cascade file* senza l'estensione XML. Se non è specificata, la fase del training genera il file *cascade.xml* all'interno della cartella dell'*Experiment*. In caso il *cascade file* ci viene fornito (ad es. da studi precedenti) bisognerà utilizzare questa opzione per specificarne il nome.

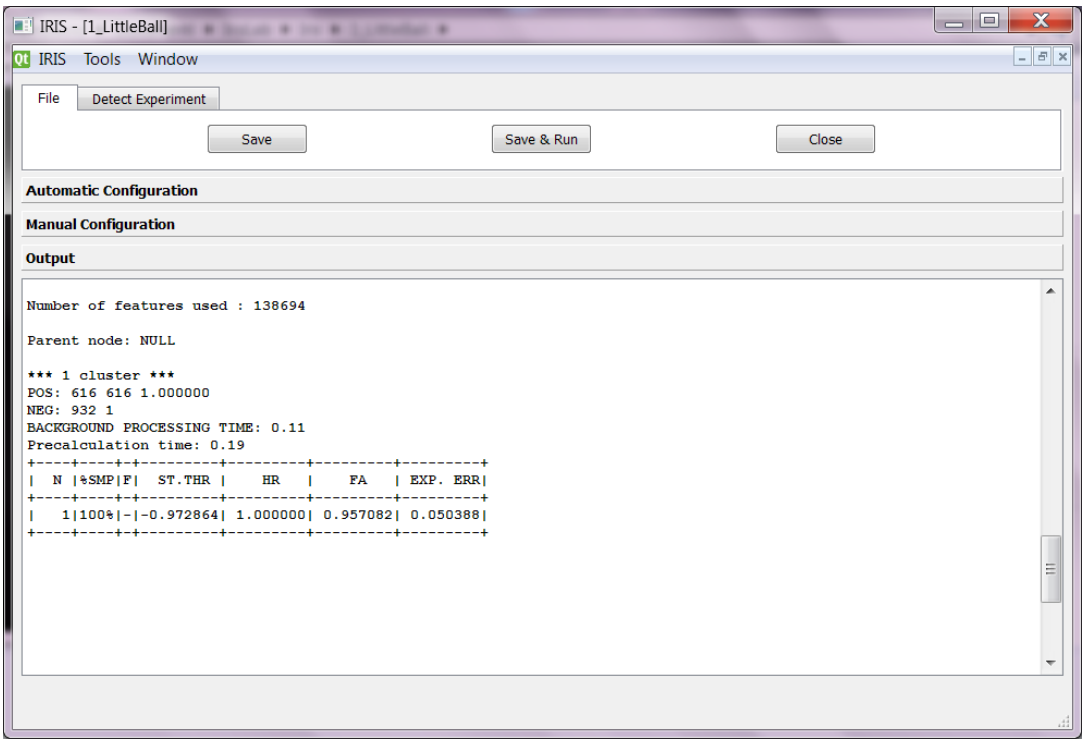


Figura 7.4: *Experiment* tipo `DETECT`. Visualizzazione dell'output del *training*.

`W e H` Indicano rispettivamente la larghezza e l'altezza in `pixel` dei campioni per il *training*. Maggiore dimensione porta a migliori prestazioni, ma anche ad un consumo notevole di memoria. Se non specificati, viene utilizzato per entrambi il valore di 24 `pixel`.

`GROUP` Funge da valore soglia per il riconoscimento di un oggetto. Deve essere riconosciuto un numero di caratteristiche almeno pari a `GROUP` per affermare che l'oggetto sia quello ricercato.

`FORCE_TRAIN` Se questa parola chiave è presente, e se il suo valore è 1, viene sempre effettuata la fase del training anche se il *cascade file* è già presente.

`MOTION_DATA` Se questa parola chiave è presente, avvia il modulo per l'analisi del moto. Il valore indica il nome del *file* in cui salvare i dati.

`OPERATION` Questa parola chiave è la più importante per la fase del *testing*. Determina ciò che il programma farà. Possono esserci quattro valori:

- `OPERATION=DETECT`. I fotogrammi, acquisiti dalla sorgente video, vengono elaborati con l'*Object Detector* e visualizzati in un'apposita finestra;

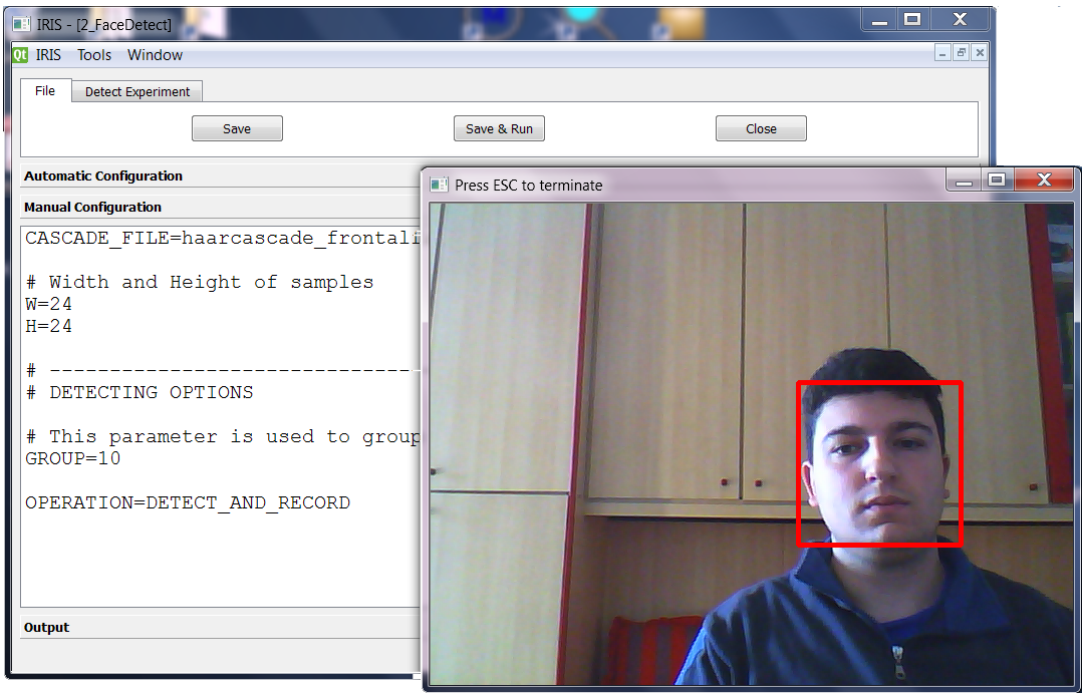


Figura 7.5: *Experiment* tipo `DETECT`. Fase di *testing*.

- `OPERATION=DETECT_AND_RECORD`. I fotogrammi, acquisiti dalla sorgente video, vengono elaborati con l'*Object Detector*, visualizzati in un'apposita finestra e memorizzati in un *file* video;
- `OPERATION=BOOTSTRAP_POS`. Il programma avvia la procedura di *bootstrap* per il miglioramento dei campioni positivi. L'oggetto è sempre presente nel campo visivo. I fotogrammi, acquisiti dall sorgente video, vengono elaborati con l'*Object Detector* e, nel caso non venga rilevato l'oggetto, viene chiesto all'utente di individuarlo. I fotogrammi così trattati, sono inseriti nei campioni positivi;
- `OPERATION=BOOTSTRAP_NEG`. Il programma avvia la procedura di *bootstrap* per il miglioramento dei campioni negativi. L'oggetto è sempre assente dal campo visivo. I fotogrammi, acquisiti dall sorgente video, vengono elaborati con l'*Object Detector* e, nel caso venga rilevato almeno un oggetto, il fotogramma è memorizzato tra i campioni negativi.

Un esempio di *file* di configurazione minimale, può essere il seguente:

Configurazione minimale per *Experiment* di tipo DETECT

```
DESCRIPTION=Detect faces on the webcam
TYPE=DETECT

# Cascade file
CASCADE_FILE=haarcascade_frontalface_default

# Width and Height of samples
W=24
H=24

# ____________________________________________________
# DETECTING OPTIONS

# This parameter is used to group small detected
# objects in bigger ones.
GROUP=10

OPERATION=DETECT_AND_RECORD
```

7.3.5 Le opzioni. Oltre alle parole chiave sopra indicate, sono presenti altre opzioni. Queste servono per gestire i parametri dell'*Object Detector*, della registrazione o acquisizione video.

Nella Tabella 7.3 è presente l'elenco completo delle parole chiave per l'*Experiment* tipo DETECT.

Parola chiave	Valore	Descrizione
`NEG_DIR`	testo	Nome della cartella contenente i campioni negativi
`POS_DIR`	testo	Nome della cartella contenente i campioni positivi
`STAGES_NUM`	numero intero	Numero di *stages* per il *training*
`SPLITS`	numero intero	Numero di frazionamenti nell'algoritmo *Object Detector*
`MEM`	numero intero	Memoria disponibile (in `MByte`) per il *training*
`MIN_HIT_RATE`	numero decimale	Percentuale minima di successi per terminare il *training*
`MAX_FALSE_ALARM`	numero decimale	Percentuale massima di falsi positivi
`MODE`	ALL o CORE	*Features* da utilizzare nell'*Object Detector*
`REC_FILE`	testo	Nome del *file* video da registrare
`FPS`	numero intero	Valore del *Frame Per Second* del video registrato

Tabella 7.3: *Experiment* tipo `DETECT`. Opzioni nel *file* di configurazione.

Configurazione ottimale per *Experiment* di tipo DETECT

```
DESCRIPTION=Detec a little ping-pong ball (including training)
TYPE=DETECT

# ---------------------------------------------------------------
# TRAINING OPTIONS

# Must tain every time
FORCE_TRAIN=1

# Number of training stages
STAGES_NUM=15
SPLITS=4
MIN_HIT_RATE=0.9992
MAX_FALSE_ALARM=0.495
MEM=760
MODE=ALL

# Directory containing the positive images
POS_DIR=pos

# Directory containing the negative images
NEG_DIR=neg


# Cascade file (without XML extention)
CASCADE_FILE=little_ball

# Width and Height of samples
#W=24
#H=24

# ---------------------------------------------------------------
# DETECTING OPTIONS

# This parameter is used to group small deteted
# objects in bigger ones.
GROUP=10

# Motion analysis
MOTION_DATA=data_position.csv

# Record output using video file
REC_FILE=video.avi

# Operation to be done
OPERATION=DETECT_AND_RECORD
```

7.3.6 L'analisi del moto. Con l'opzione MOTION_DATA il programma avvia il modulo per l'analisi del moto.

In particolare questo modulo consente di:

- Salvare i dati di posizione in in formato CSV;
- Avere il grafico istantaneo degli spostamenti;
- Calcolo della velocità e accelerazione mediante derivazione numerica.

La posizione dell'oggetto è memorizzata in maniera adimensionale. In Figura 7.6 sono indicati il sistema di riferimento per l'analisi del moto e le dimensioni di adimensionalizzazione.

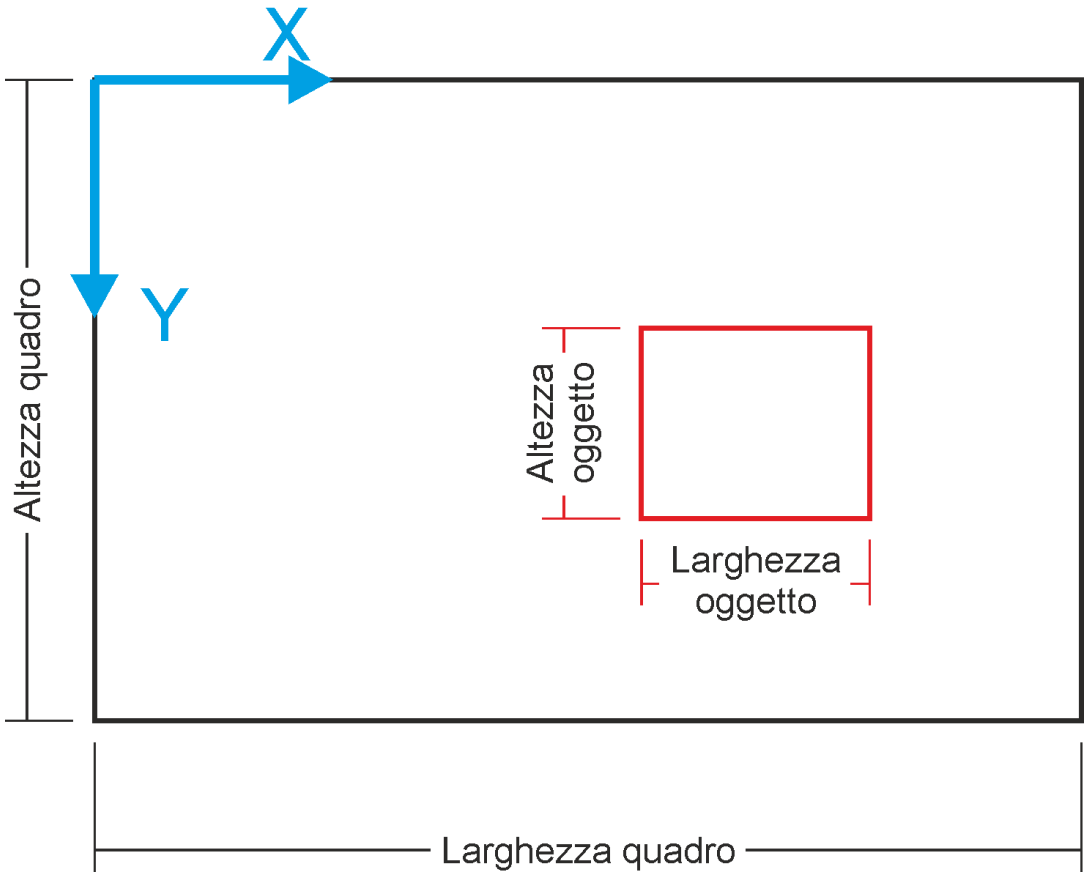
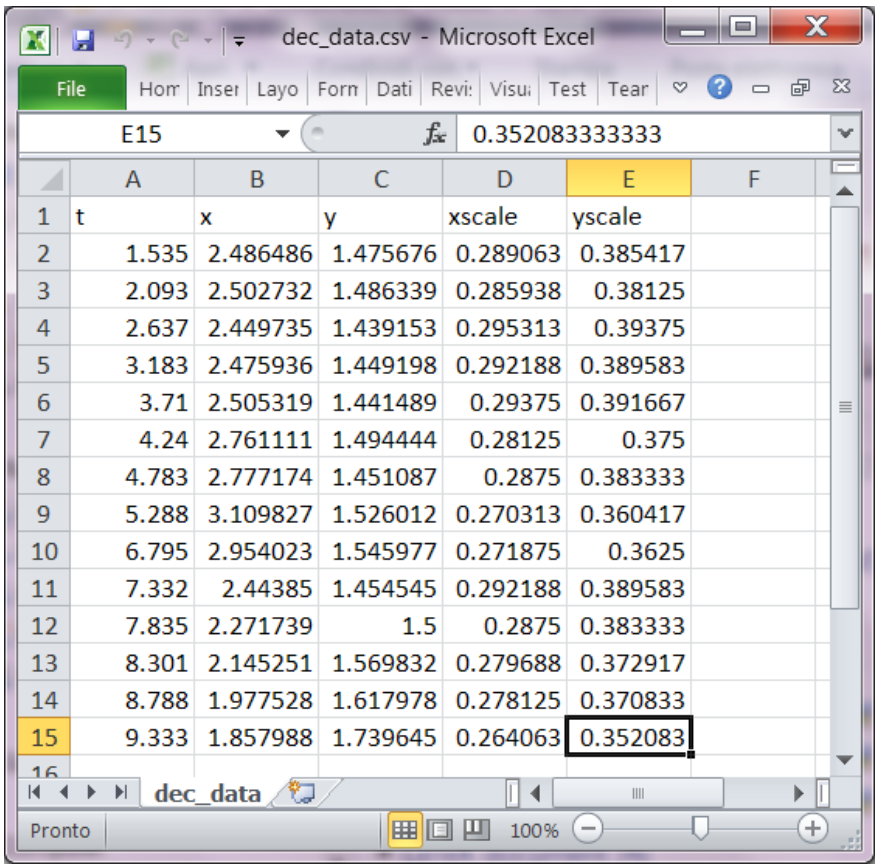


Figura 7.6: Modulo *Motion Analysis*. Convenzione ed adimensionalizzazione.

Ad ogni acquisizione di un fotogramma, viene memorizzato il tempo (in secondi). Le coordinate X ed Y sono adimensionalizzate, rispettivamente, rispetto alla larghezza e all'altezza del quadro di acquisizione. Questi rapporti, sono anch'essi memorizzati nei dati del *Motion Analyzer*, per una verifica successiva di complanarità.

I dati possono essere elaborati in programmi esterni (ad es. Excell) come visibile nella Figura 7.7.

Alla fine della fase di *testing* il modulo presenterà, automaticamente, il grafico di spostamento, velocità ed accelerazione come mostrato in Figura 7.8.



	A	B	C	D	E	F
1	t	x	y	xscale	yscale	
2	1.535	2.486486	1.475676	0.289063	0.385417	
3	2.093	2.502732	1.486339	0.285938	0.38125	
4	2.637	2.449735	1.439153	0.295313	0.39375	
5	3.183	2.475936	1.449198	0.292188	0.389583	
6	3.71	2.505319	1.441489	0.29375	0.391667	
7	4.24	2.761111	1.494444	0.28125	0.375	
8	4.783	2.777174	1.451087	0.2875	0.383333	
9	5.288	3.109827	1.526012	0.270313	0.360417	
10	6.795	2.954023	1.545977	0.271875	0.3625	
11	7.332	2.44385	1.454545	0.292188	0.389583	
12	7.835	2.271739	1.5	0.2875	0.383333	
13	8.301	2.145251	1.569832	0.279688	0.372917	
14	8.788	1.977528	1.617978	0.278125	0.370833	
15	9.333	1.857988	1.739645	0.264063	0.352083	

Figura 7.7: Modulo *Motion Analysis*: *file* dati elaborato in Excell.

7.4

Tipologia *CAPTURE*

La tipologia `CAPTURE` ha come obiettivo la gestione dei *file* video, un problema ricorrente nella *Digital Image Processing*. É possibile registrare video, visualizzare la zona inquadrata dalla webcam oppure salvare i singoli fotogrammi acquisiti come immagini.

Le parole chiave disponibili per questo *Experiment* sono poche, se confrontate con quelle della tipologia `DETECT`.

7.4.1 Il *file* della configurazione. Un tipico *file* di configurazione, è quello che prevede la registrazione di un video ed il salvataggio dei singoli fotogrammi come immagini.

Experiment *tipo `CAPTURE`: configurazione minimale*

```
DESCRIPTION=Capture some images.
TYPE=CAPTURE

SOURCE=rec_video.avi
```

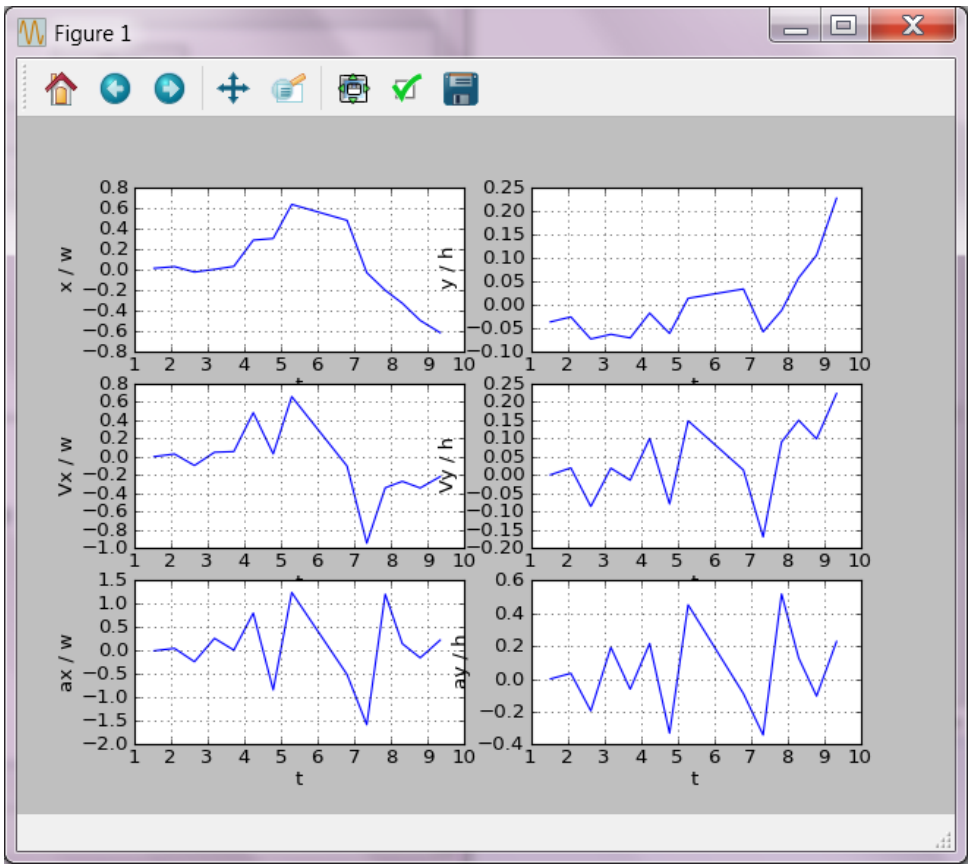


Figura 7.8: Modulo *Motion Analysis*: grafici istantanei.

Le immagini, sono salvate nella cartella dell'*Experiment*. Poiché solitamente questa procedura è utilizzata per la raccolta dei campioni dell'*Object Detector*, al termine della registrazione, il programma chiede automaticamente di catalogare le immagini.

Tra le parole chiave disponibili, ne elenchiamo alcune di notevole importanza.

`NAME` Permette di scegliere il nome dei *file* immagine. Se assente, ogni fotogramma verrà salvato utilizzando come nome del *file* la data e ora corrente.

`FPS` Stabilisce il valore del *Frames Per Second* dell'acquisizione video. Un alto valore del `FPS` determina una grande frequenza di acquisizione e quindi una notevole quantità di immagini prodotte.

`OUTDIR` Impone al programma di salvare i *file* immagine in una determinata cartella. Se tale cartella non esiste, verrà creata.

7.4.2 Le opzioni. Oltre alle parole chiave, sono disponibili due opzioni per la catalogazione delle immagini come campioni positivi e negativi.

`POS_DIR` La parte delle immagini contrassegnata come campione positivo verrà salvata in questa cartella.

`NEG_DIR` Le immagini contrassegnate come campioni negativi verranno memorizzate in questa cartella.

Experiment *tipo* `CAPTURE`*: configurazione completa*

```
DESCRIPTION=Capture some images.
TYPE=CAPTURE

# Record from the 2nd webcam which is connected to the PC
SOURCE=2

# Frames per second
FPS=2

# Output directory
OUTDIR=samples

# Sub-directory for storing positive and negative samples
POS_DIR=pos
NEG_DIR=neg
```

7.5 Tipologia *EDGE*

La tipologia di *Experiment* `EDGE` utilizza l'*Edge Detector* in tempo reale. L'obiettivo principale, è pertanto il *tuning* (regolazione) dei parametri di questo algoritmo.

La sorgente video può essere la webcam oppure, un *file* video. Il risultato, visibile sullo schermo, sarà frutto dell'applicazione istantanea del *Canny Detector* e delle *Linee di Hough*. Le linee fondamentali saranno evidenziate in rosso.

In Figura 7.9 è possibile vedere un *Experiment* in corso. I parametri sono regolati nel *file* di configurazione ed il risultato è mostrato in una finestra a parte.

I parametri regolati dalle parole chiave, sono i medesimi descritti nella Tabella 5.1.

7.5.1 Il *file* della configurazione. Elenchiamo qui le parole chiavi più importanti per la tipologia `EDGE`.

`DIST_RES` Risoluzione spaziale. Collegato direttamente con l'errore sulla misura di ρ per l'*Edge Detector*. Più piccolo è questo valore, più sarà precisa la determinazione delle linee base della geometria. Valori troppo piccoli possono portare a tempi di calcolo eccessivi.

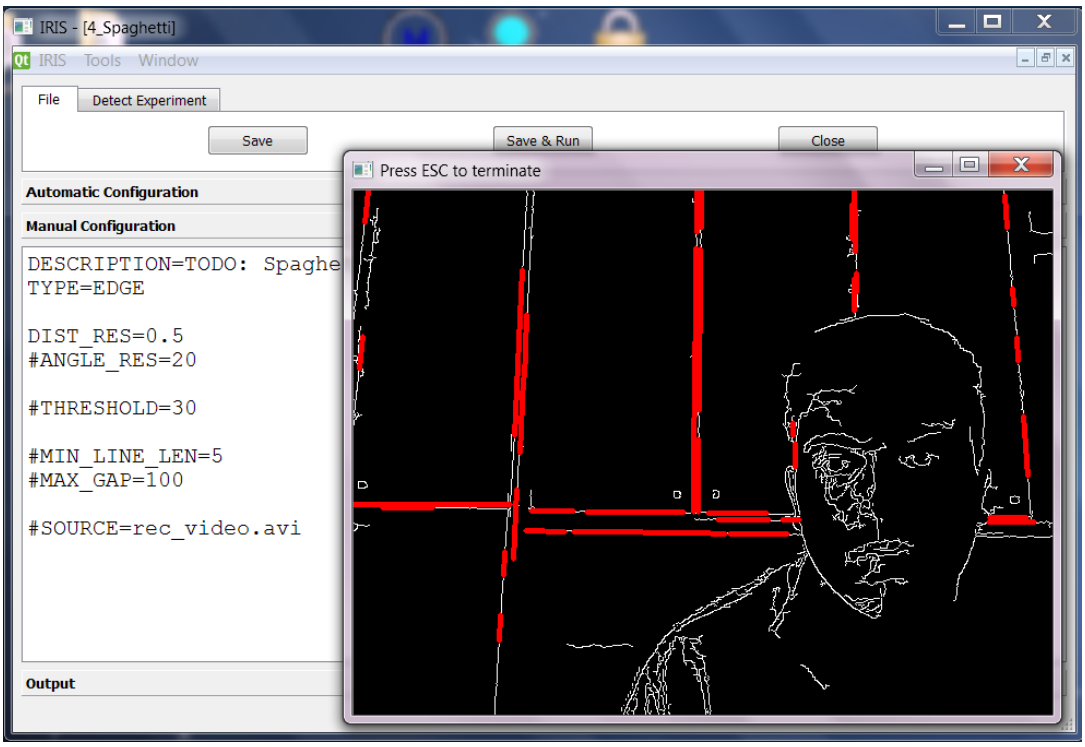


Figura 7.9: *Experiment* tipo `EDGE`: rilevazione dei bordi in tempo reale.

`ANGLE_RES` Risoluzione angolare. Collegato direttamente con l'errore sulla misura di θ per l'*Edge Detector*. Anche qui, un valore minore porta a calcoli più accurati ma più lunghi.

`THRESHOLD` Soglia di probabilità oltre la quale una linea è considerata come tale.

`MIN_LINE_LEN` Lunghezza minima della linea per la geometria di base.

`MAX_GAP` Distanza massima tra due linee, prima di raggrupparle.

Un esempio di *file* di configurazione è il seguente:

Experiment *tipo `EDGE`: configurazione ottimale*

```
DESCRIPTION=Real-time Edge Detector
TYPE=EDGE

DIST_RES=1
ANGLE_RES=1

THRESHOLD=30

MIN_LINE_LEN=5
MAX_GAP=100

SOURCE=0
```

7.6

La *Wizard* o procedura automatica

Fin ora, abbiamo descritto il funzionamento del programma e i *file* di configurazione. Per la creazione di un nuovo *Experiment* non è necessario compiere manualmente queste operazioni: è stata progettata appositamente una proceduta automatica (in gergo, *Wizard*) per la creazione di nuovi *Experiment*.

Alla *Wizard* si accede dal Menù principale, come in Figura 7.10.

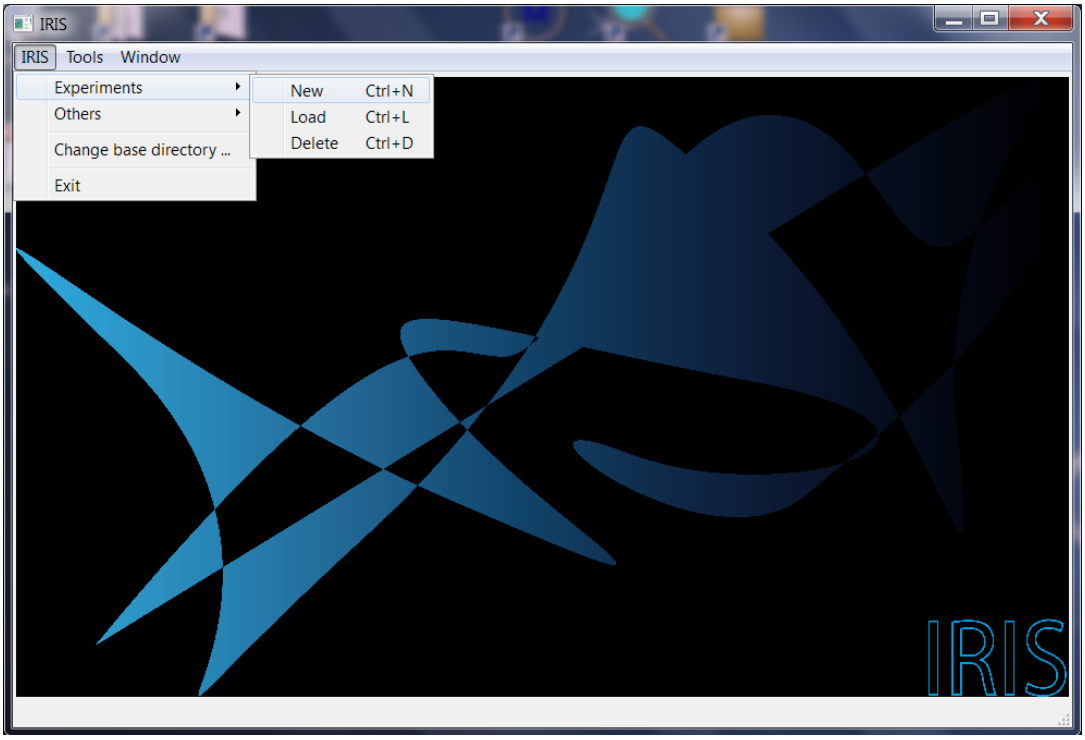


Figura 7.10: Wizard. Creazione di un nuovo *Experiment*

Compare la finestra di benvenuto che descrive i tipi di *Experiment* e quello che possiamo fare con essi (Figura 7.11).

Viene scelto il tipo di *Experiment* (Figura 7.12). Il nome è un parametro necessario e deve rispettare le regole per i nomi dei *file* del sistema operativo in uso. La descrizione è invece facoltativa.

La schermata successiva è differente per ogni tipo di *Experiment*. In questa fase viene generato automaticamente il *file* di configurazione, scegliendo il valore dei parametri che riteniamo più opportuni. In Figura 7.13 viene mostrata la schermata per il tipo `DETECT`.

Il buon fine della procedura è segnalata dall'ultima schermata (Figura 7.14). Se in precedenza è stata selezionata l'opzione *Capture Now!* il programma inizierà la fase di

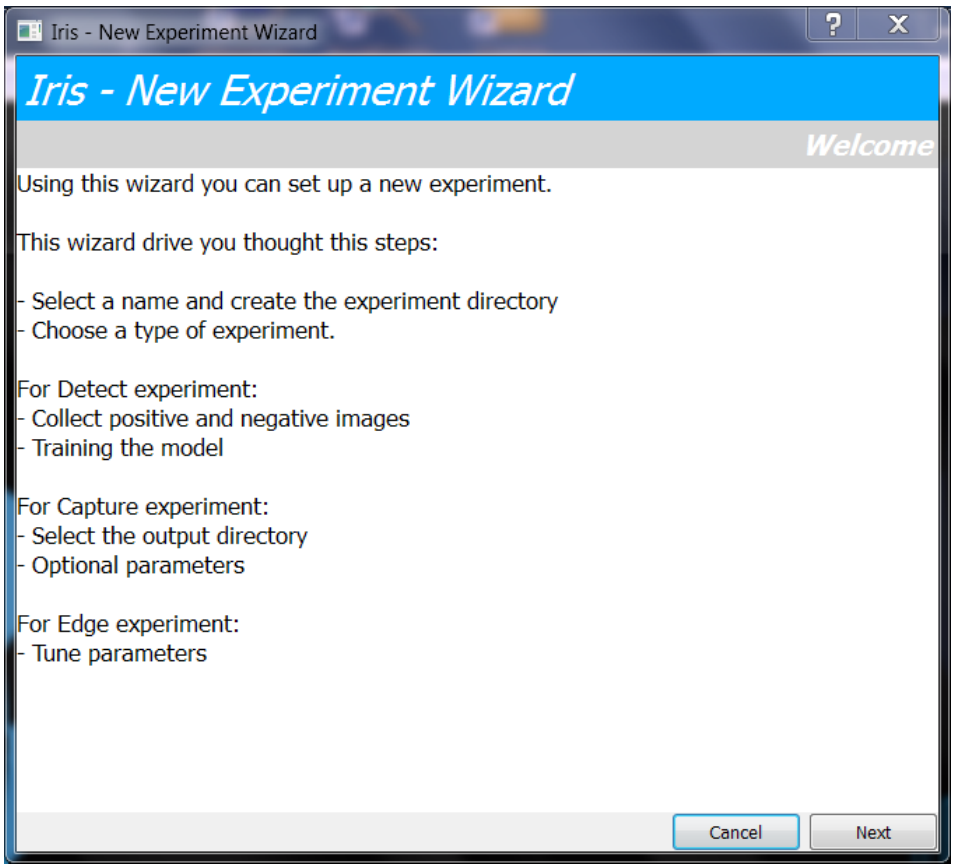


Figura 7.11: Wizard. Finestra di benvenuto.

registrazione o di raccolta dei campioni.

Da questo punto in poi, l'*Experiment* è pronto.

7.7 Lavorare ad un *Experiment* già esistente

Quando un *Experiment* è già presente nella cartella predefinita, possiamo caricarne il contenuto utilizzando l'*Experiment Selector*.

Dal Menù principale si seleziona la voce `Experiment` ed infine, `Load`. In Figura 7.15 è possibile vedere la finestra per la scelta dell'*Experiment*. In alto è presente l'opzione per la selezione della cartella predefinita. In basso, l'elenco degli *Experiment* presenti in tale cartella. Accanto al nome, è presente il tipo e la descrizione.

Selezionato l'*Experiment*, si aprirà l'*Editor della configurazione* già descritto in questo capitolo.

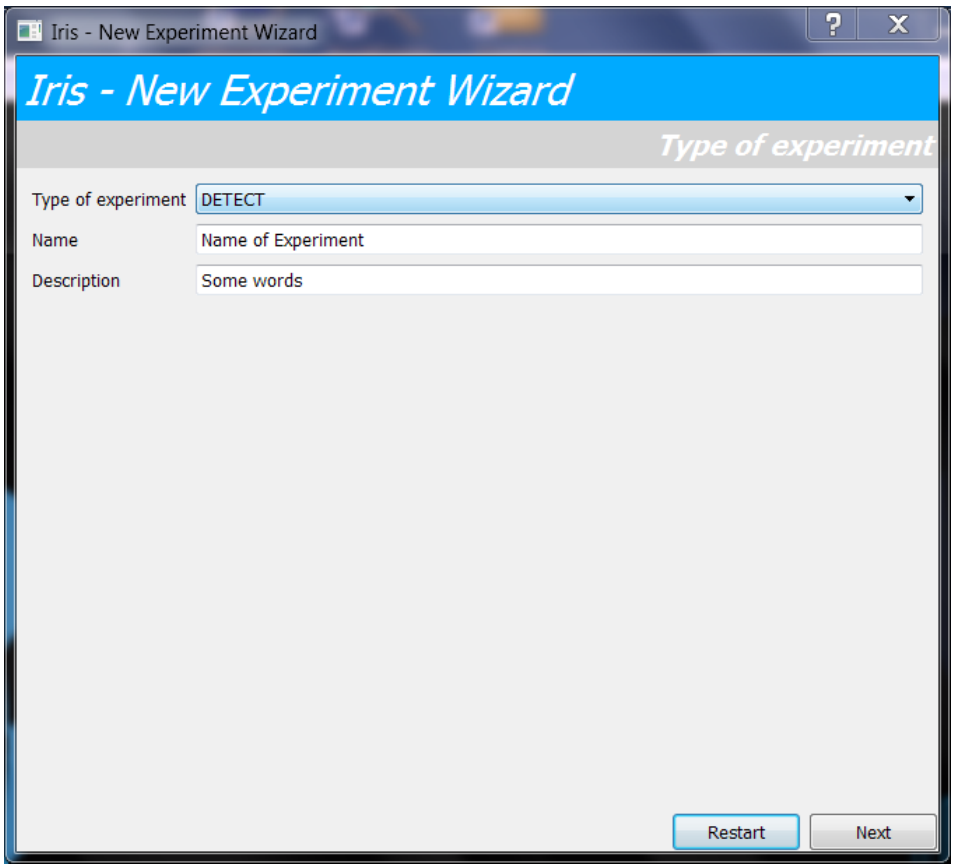


Figura 7.12: Wizard. Scelta del tipo di *Experiment*.

7.8

Importare un *cascade file*

Per un *Experiment* tipo `DETECT` il *cascade file* può essere fornito da fonti esterne (ad esempio il training è stato svolto su un altro calcolatore oppure è stato svolto da altri).

É questo il caso di alcuni *cascade file* prodotti in studi precedenti. La libreria OpenCV fornisce, ad esempio, i *cascade file* pronti all'uso per il riconoscimento dei volti. In questa Tesi, invece, per la validazione dei risultati, è stato utilizzato un *cascade file* fornito dallo studio in [5].

Per utilizzare un *cascade file* la procedura è semplice:

1. Creare una cartella per l'*Experiment*;
2. Spostare il *cascade file* in tale cartella;
3. Scrivere un *file* di configurazione per l'*Object Detector* facendo attenzione a specificare il nome del *file* importato.

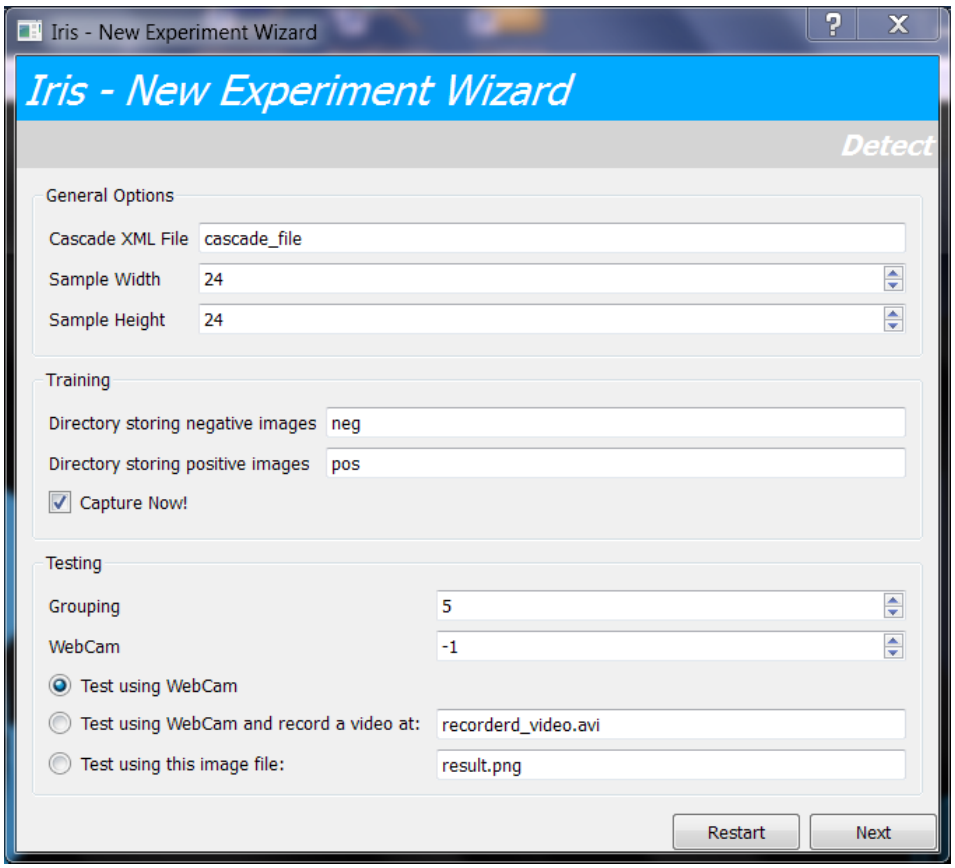


Figura 7.13: Wizard. Creazione automatica del *file* di configurazione.

Un esempio di configurazione minimale è la seguente:

File *di configurazione per l'importazione di un* cascade file

```
DESCRIPTION=Detect faces on the webcam (import a cascade file)
TYPE=DETECT

# Write here the file name
CASCADE_FILE=haarcascade_frontalface_default

# ------------------------------------------------------------
# DETECTING OPTIONS

# This parameter is used to group small deteted objects
# in bigger ones.
GROUP=10

# Write this line to do motion analysis
MOTION_DATA=dec_data.csv

# "Operation" is needed. You can record or just detect
OPERATION=DETECT
```

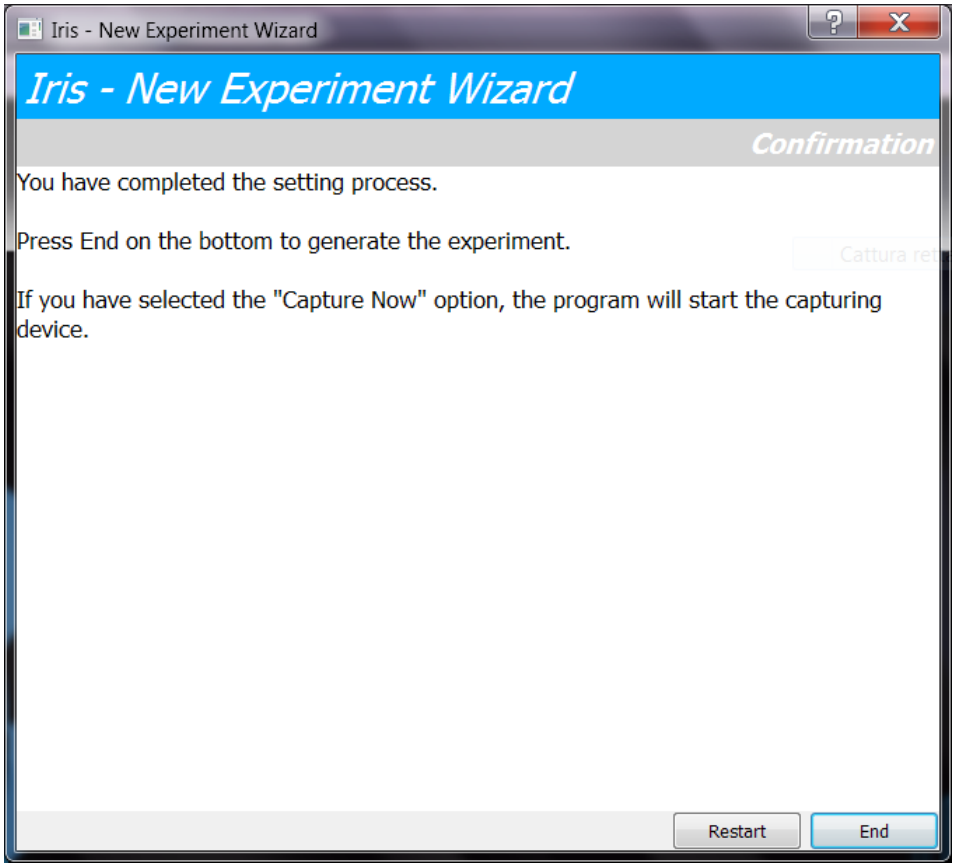


Figura 7.14: Wizard. Fine della procedura automatica.

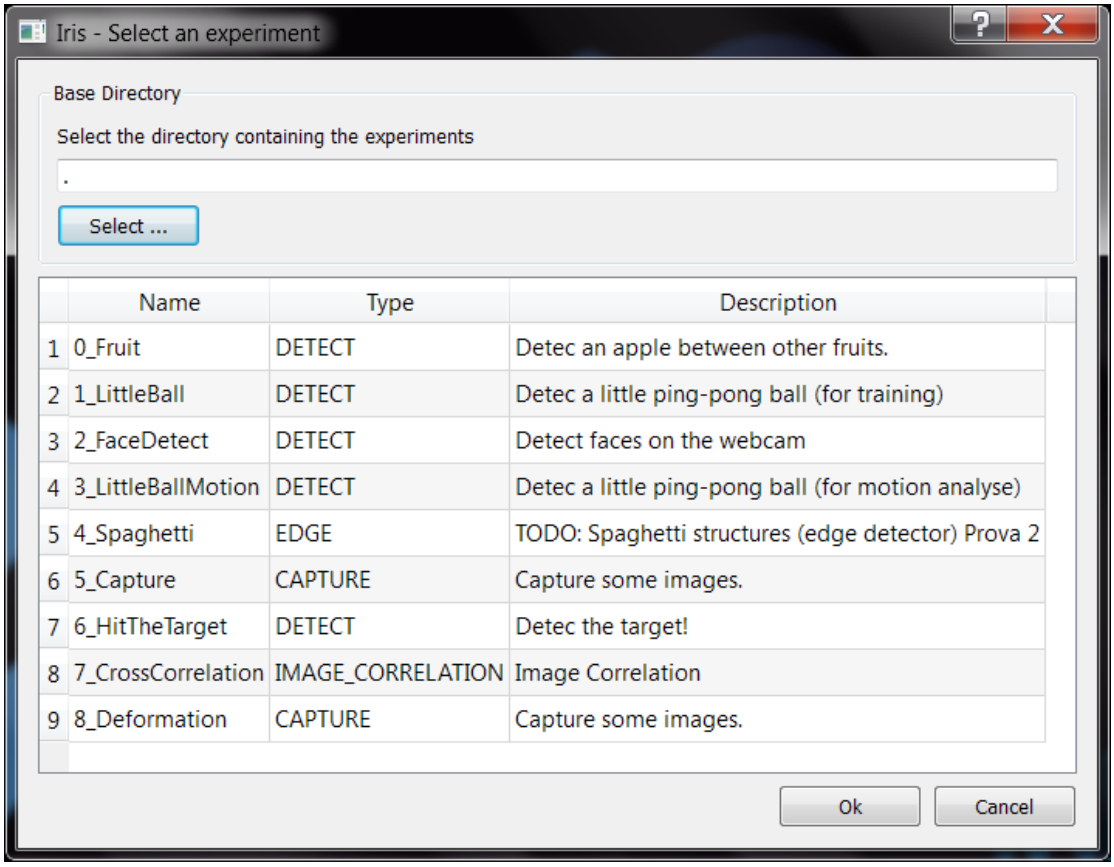


Figura 7.15: *Experiment Selector* per la selezione dell'*Experiment*.

8

Modulo *Deformation From Image*

L'obiettivo di questo modulo è comparare due immagini, ottenendo il campo degli spostamenti e/o deformazione. Viene adoperata la tecnica dell'*Image Correlation* per confrontare l'immagine di stato indeformato, con l'immagine di stato deformato.

Il programma implementa queste caratteristiche in un'interfaccia grafica dedicata. Sono disponibili una serie di procedure automatiche per l'analisi comparata tra più campioni.

Infine, uno sguardo attento è rivolto all'analisi e alla gestione dei dati.

8.1 Analisi tra due immagini

L'analisi base che il *software* deve condurre è sicuramente il confronto tra due immagini.

L'algoritmo si basa sulla divisione dell'immagine iniziale (di stato non deformato) in finestre di campionamento. Ognuna di queste finestre è poi ricercata nell'immagine di destinazione, come descritto nella Parte I.

In Figura 8.1 è possibile vedere la finestra del programma per l'utilizzo di questa tecnica. Tra i comandi, in alto a sinistra, si notano i pulsanti per la selezione delle immagini di stato deformato ed indeformato.

In dettaglio, la ricerca del campione avviene utilizzando la libreria OpenCV che mette a disposizione la tecnica dell'*Image Correlation*. É possibile scegliere la modalità con cui normalizzare la distribuzione di probabilità risultate.

L'algoritmo di analisi è riportato in questa sezione. Il codice fornisce una serie di informazioni:

- Confidenza. Per ogni finestra di campionamento ci fornisce la probabilità con cui è stata identificata;

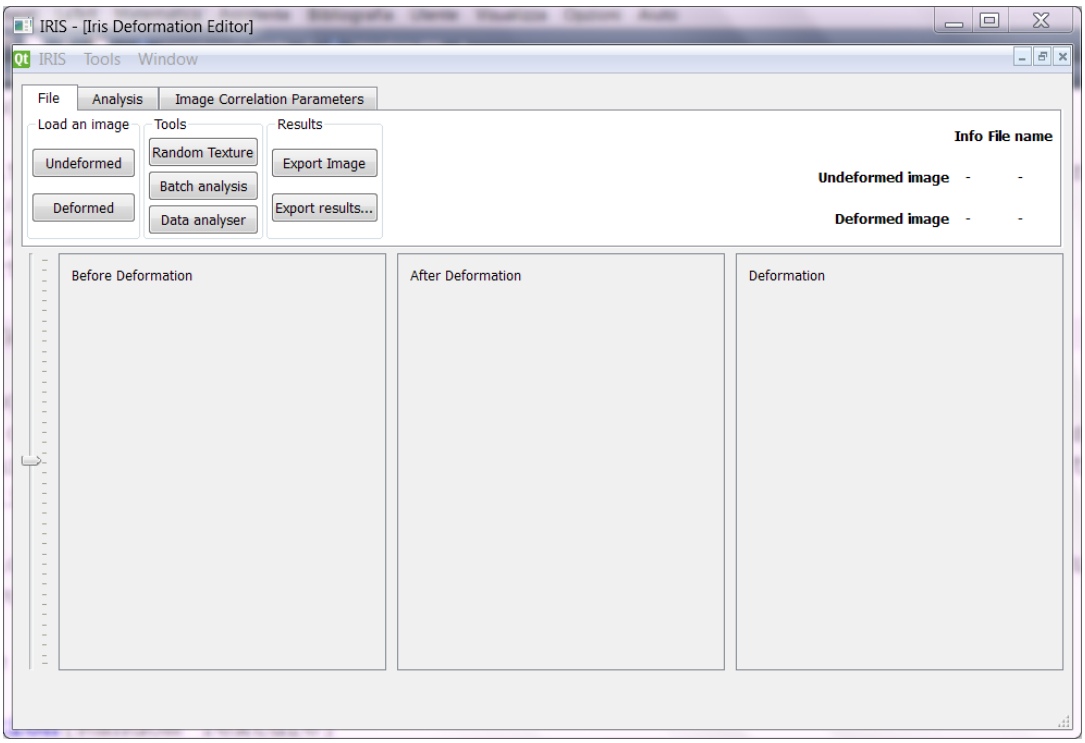


Figura 8.1: Modulo *Deformation From Image*

- Traslazione lungo X ed Y. Calcola la traslazione in direzione X ed Y sia in `pixel` che in millimetri;
- Traslazione totale. Ricavata dalle traslazioni lungo X ed Y mediante il teorema di Pitagora.

8.1.1 Random Texture. La tecnica dell'*Image Correlation* restituirà risultati sempre più accurati quanto più univocamente è possibile individuare il campione tratto dall'immagine di stato indeformato. La condizione migliore presenta in ogni campione una trama univoca. Questo si ottiene riproducendo una trama *casuale* nell'immagine di stato indeformato.

Per provini metallici, è possibile riprodurre su di essi una trama causale con delle vernici spray. Questa tecnica è stata utilizzata ampiamente in bibliografia, ad esempio in [1]. Rivestono un ruolo importante, la dimensione e la distribuzione delle macchie di pittura.

Il programma, a tal fine, è stato munito di generatore di *Random Texture*. Si produce un'immagine con distribuzione casuale di punti neri, decidendone la dimensione e la densità. Tale immagine può essere, ad esempio, stampata sul provino.

Un esempio di *random texture* è riportata in Figura 8.2.

Codice 8.1: *Algoritmo per* Deformation From Image

```
def analysingProcess(
    deformedImg,
    undeformed,
    threshold,
    contour,
    method,
    rects):

  if len(rects) <= 0:
    return []

  uImg = cv.LoadImage(undeformed)
  dImg = cv.LoadImage(deformedImg)

  wD, hD = cv.GetSize(dImg)

  values = []

  for (x, y, wT, hT) in rects:
    template = cv.GetSubRect(uImg, (x, y, wT, hT))

    result = cv.CreateImage((wD - wT + 1, hD - hT + 1), 32, 1)
    cv.MatchTemplate(dImg, template, result, method)
    minVal,maxVal,minLoc,maxLoc = cv.MinMaxLoc(result)

    if math.fabs(maxVal- minVal) >= threshold:
      newx = maxLoc[0]
      newy = maxLoc[1]

      if contour == 0:
        val = math.fabs(maxVal- minVal) * 100.0
      elif (contour == 1) or (contour >= 4):
        val = math.sqrt( (x - newx)**2 + (y - newy)**2 )
      elif contour == 2:
        val = newx - x
      elif contour == 3:
        val = -(newy - y)

      values.append((x, y, wT, hT, newx, newy, val))

  return values
```

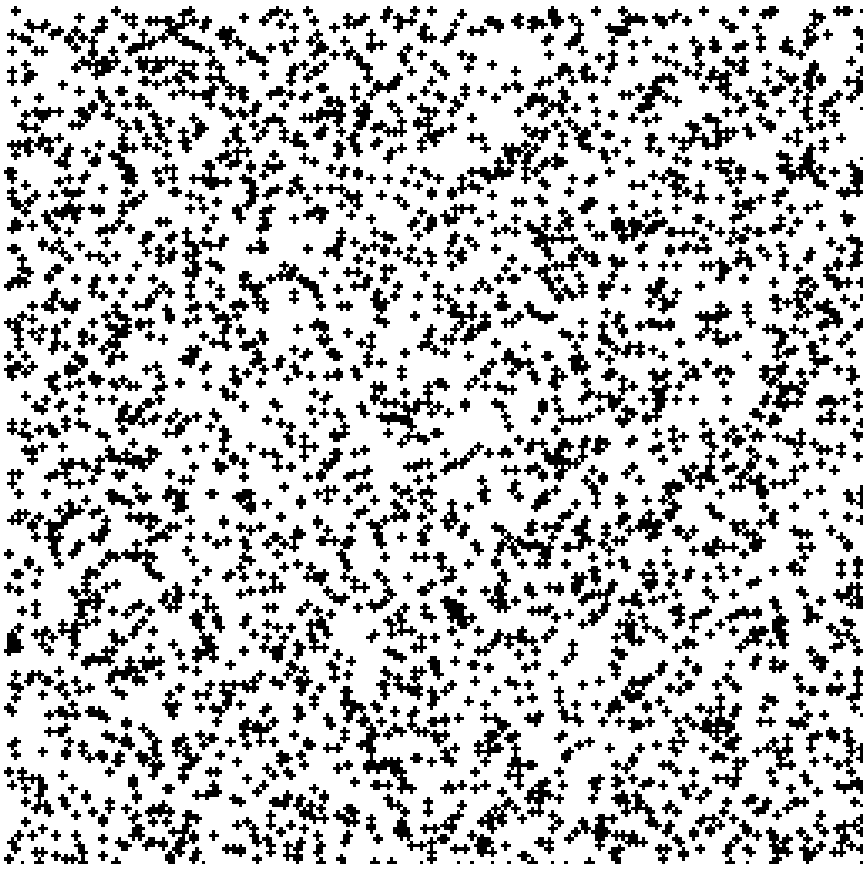


Figura 8.2: Esempio di *Random Texture* per l'utilizzo con l'*Image Correlation*.

8.1.2 Parametri della Image Correlation. Come le altre tecniche, anche per l'*Image Correlation* è possibile regolare alcuni parametri. In Figura 8.3 si può vedere come il programma gestisce la scelta dei parametri.

Di primaria importanza è la dimensione delle finestre di campionamento. Queste dimensione è sempre espressa in `pixel`. Dimensioni troppo grandi o troppo piccole della finestra di acquisizione forniscono dei risultati non attendibili. Nel primo caso perché poche finestre non riproducono il campo degli spostamenti, nel secondo perché perdono di univocità. Dimensioni pari a $\frac{1}{10}$ della larghezza dell'immagine iniziano a fornire dei risultati utili. Per immagini di 5 `MP` finestre di campionamento di 24x24 `pixel` sono un buon compromesso tra tempi di calcolo (circa 10 minuti su un moderno calcolatore) e accuratezza.

L'altro parametro importante è la soglia (*threshold*) di probabilità. I campioni con probabilità al di sotto della soglia non sono presi in considerazione nell'analisi.

É possibile filtrare uno spostamento costante impostando una soglia minima di spostamento.

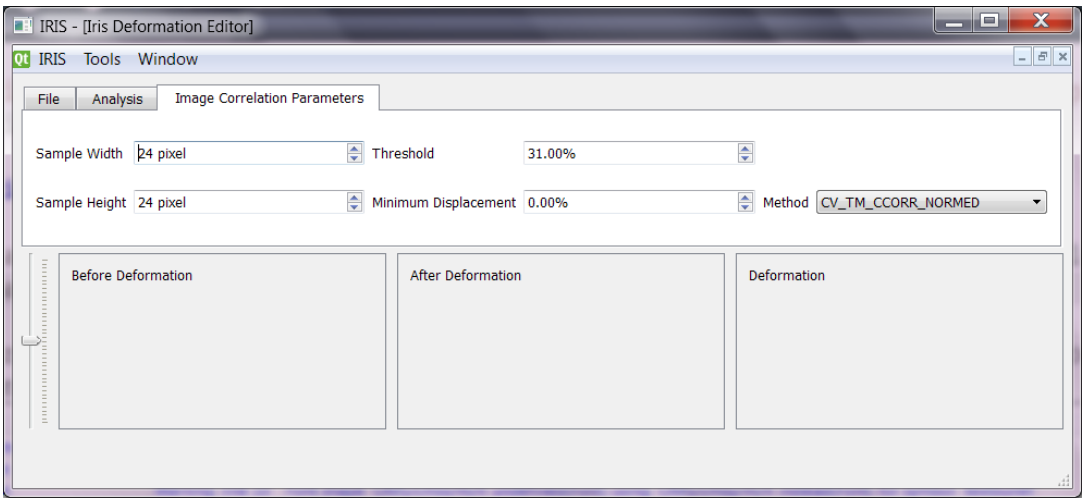


Figura 8.3: Modulo *Deformation From Image*: modifica dei parametri

8.1.3 Stima della confidenza. Prima di calcolare i valori della traslazione, è necessario ricercare la soglia di probabilità da imporre al programma.

Una soglia troppo bassa porta ad errori grandi. Ad esempio, tutti i campioni avranno una probabilità non nulla. Scegliendo una soglia nulla, ciascun campione andrebbe bene dappertutto.

Una soglia troppo alta, invece, darebbe l'effetto opposto: non è detto che ci siano campioni che superino quella soglia di probabilità.

La prima analisi da compiere, dunque, è quella di confidenza. Si procede effettuando delle analisi con finestre di campionamento molto ampie (per ridurre i tempi di calcolo) e con un valore soglia intorno al 30%. Successivamente, in base ai risultati ottenuti, si modifica tale valore e si può ridurre la dimensione delle finestre di campionamento.

In Figura 8.4 è possibile notare i comandi da utilizzare per far iniziare un'analisi di confidenza.

8.1.4 Stima della traslazione. Successiv all'analisi di confidenza, si effettua l'analisi per la stima della traslazione.

Impostata la soglia di probabilità, basterà selezionare l'opzione `Translation` nelle varie forme (lungo X, lungo Y o totale) per ottenere i risultati desiderati.

La stima della traslazione è sempre effettuata, dal programma, in `pixel`. L'eventuale trasformazione in altre unità di misura, sarà discussa in seguito.

8.1.5 Stima della rotazione. Il metodo del'*Image Correlation* fornisce soltanto la traslazione, nelle due direzioni, del campione. Per una stima della rotazione c'è bisogno di

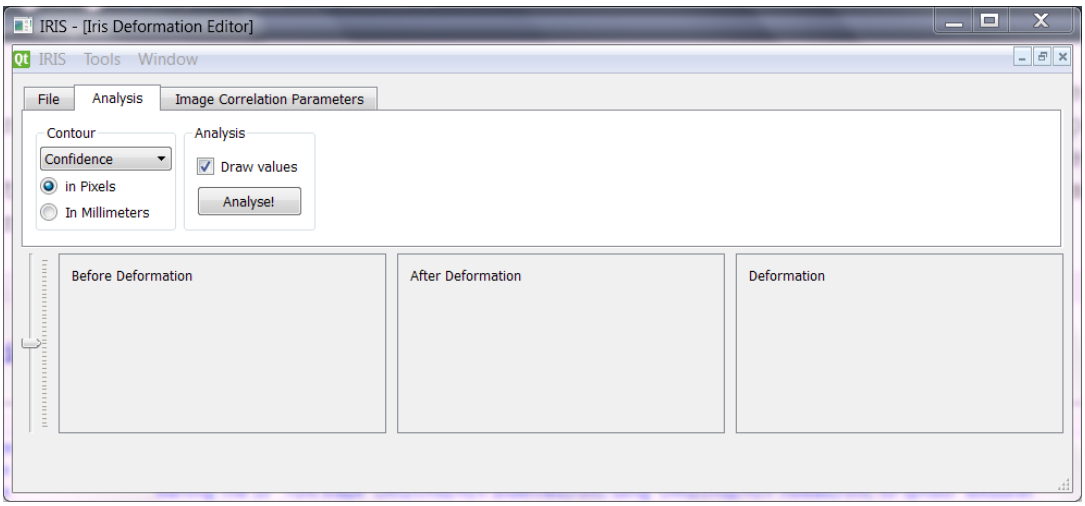


Figura 8.4: Modulo *Deformation From Image*: opzioni per l'analisi di confidenza.

integrare i dati con una formulazione matematica appropriata.

In Figura 8.6 sono rappresentate le convenzioni sugli assi di riferimento. Conoscendo le coordinate del punto A, B e C possiamo stimare le rotazioni rispetto al punto A.

Ad esempio, ipotizziamo di considerare B come il punto nel quale inizia la finestra di campionamento per l'immagine di stato indeformato. Dopo l'analisi, sappiamo che le coordinate del campione sono quelle del punto C nell'immagine di stato deformato. Volendo calcolare l'angolo di rotazione β di B intorno ad A, otteniamo:

$$\begin{cases} \gamma &= \arctan\left(-\frac{y_C - y_A}{x_C - x_A}\right) \\ \alpha &= \arctan\left(-\frac{y_B - y_A}{x_B - x_A}\right) \end{cases} \rightarrow \beta = \gamma - \alpha \tag{8.1}$$

Questa elaborazione può essere effettuata agevolmente con i dati risultanti dall'analisi della traslazione.

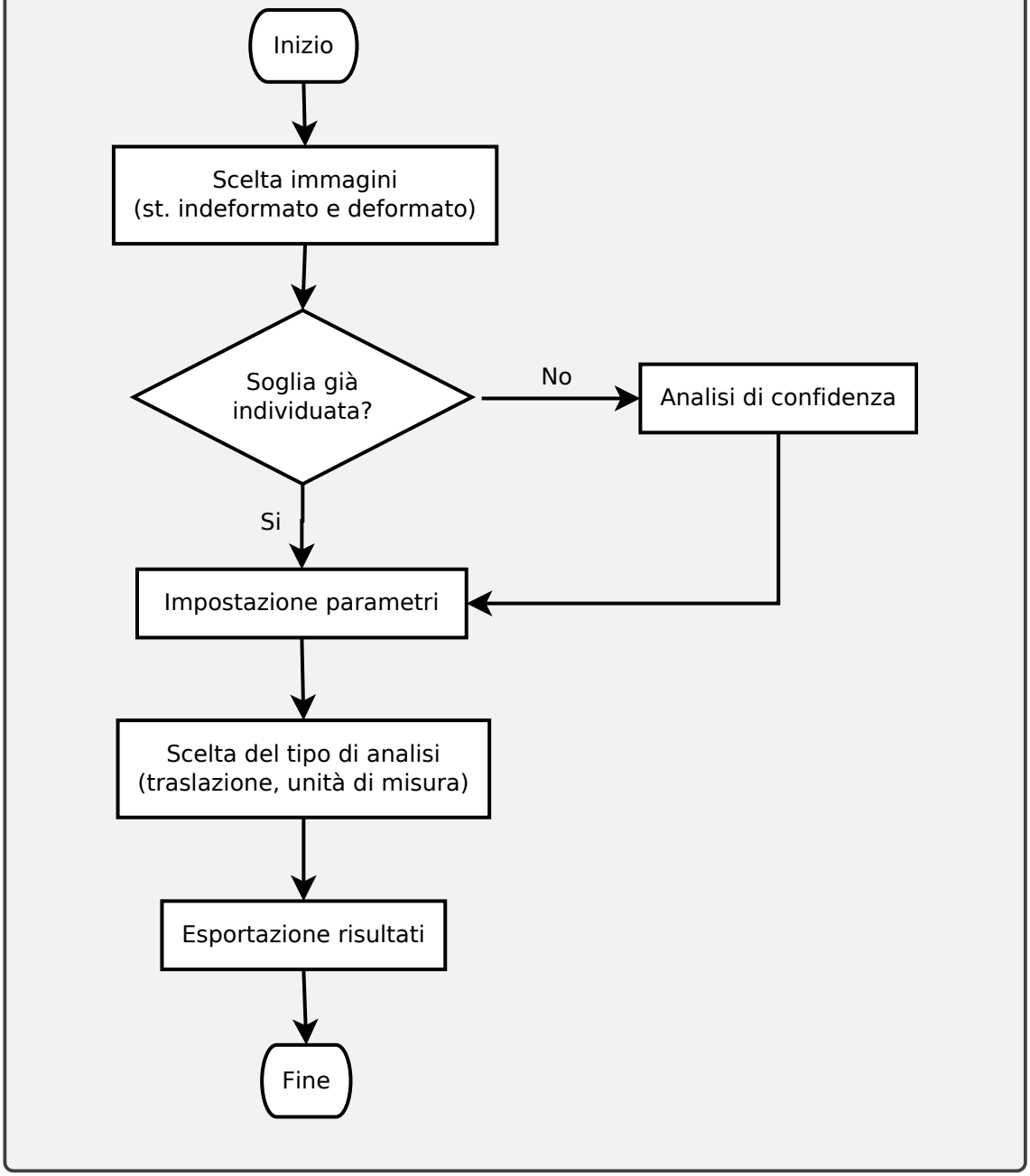
8.1.6 Conversione Pixel in Millimetri. L'unità di misura predefinita è il `pixel`. Per dare significato fisico, e quindi stimare lo spostamento realisticamente, possiamo convertire questa unità di misura in millimetri.

Mediante la risoluzione dell'immagine (misurata in `DPI`, *Dot Per Inch*) possiamo calcolare quanti `pixel` dell'immagine equivalgono ad un millimetro.

Considerando che $1\text{inch} = 25.4\text{millimetri}$ si convertono gli spostamenti in millimetri.

Questa trasformazione è valida a meno che il dispositivo di acquisizione non presenti degli ingrandimenti e che sia posizionato molto vicino al provino.

Diagramma 8.5: Analisi con il modulo Deformation From Image



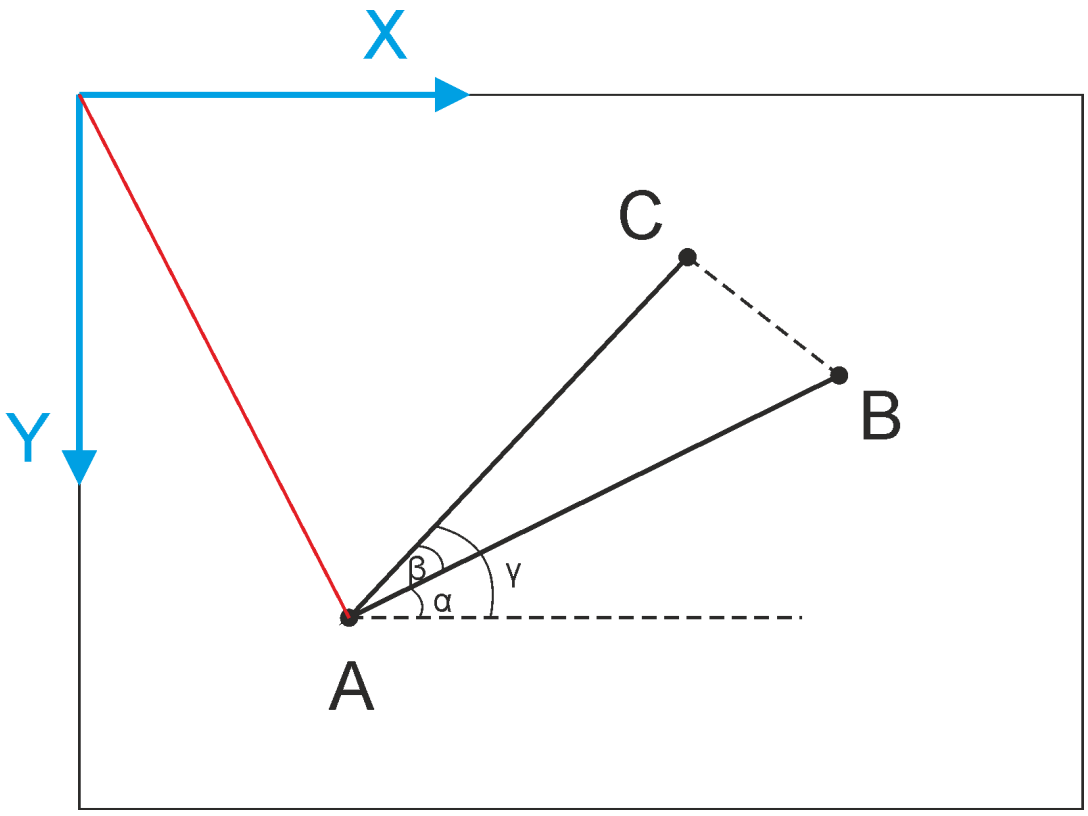


Figura 8.6: Stima della rotazione, conoscendo le coordinate di tre punti.

Nel programma, il metodo automatico per il calcolo degli spostamenti in millimetri utilizza questa procedura. É bene, quindi, rispettare le condizioni sopra indicate o in alternativa utilizzare i risultati in `pixel`.

Un approccio più empirico, ed in generale più valido, consiste nel trasformare i valori `pixel` mediante l'utilizzo di una scala graduata posta per confronto nella fotografia.

8.1.7 Esportazione dei dati. I risultati dell'analisi consistono nella rappresentazione a colori del campo di deformazione e nei dati di spostamento delle finestre di campionamento.

I dati possono essere esportati (per un'analisi più approfondita o per l'uso con altri *software*) in formato CSV. Nella Tabella 8.1 vengono descritte le informazioni presenti in ogni colonna del *file* CSV.

8.1.8 Altre opzioni. Sono disponibili altre opzioni per l'analisi di traslazione. In particolare l'opzione *Draw Values* scrive, sulla rappresentazione a colori, il valore del parametro richiesto.

Questa funzionalità è molto utile per una stima qualitativa dei risultati.

Colonna	Valore	Descrizione
1 e 2	x ed y	Coordinata x ed y della finestra di campionamento sull'immagine di stato indeformato
3 e 4	δ_x e δ_y	Spostamento in direzione x ed y del campione nell'immagine di stato deformato
5 e 6	W e H	Dimensione della finestra di campionamento
7 e 8	Valore e descrizione	Valore richiesto dall'utente per la rappresentazione a colori (può essere la traslazione, in millimetri o `pixel`, o la confidenza in percentuale)

Tabella 8.1: Modulo *Deformation From Image*: formato dei *file* per l'esportazione dei risultati.

8.2 Analisi multipla

L'analisi tra due immagini, descritta fin ora, può estendersi al caso di immagini che descrivono un'evoluzione del tempo.

Ad esempio, per un oggetto in moto, è possibile confrontare diverse posizioni ad instanti temporali diversi con la posizione al tempo iniziale.

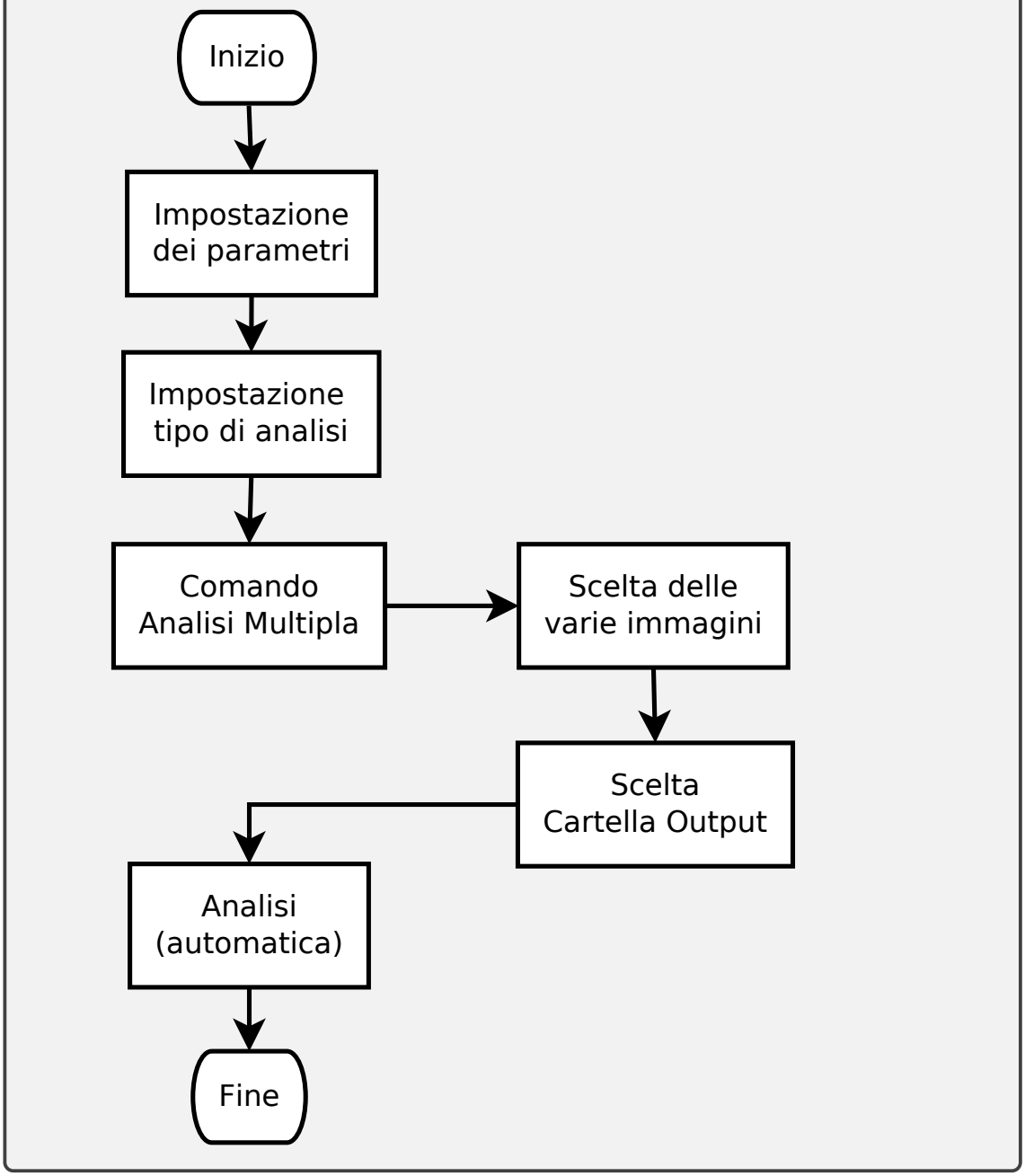
L'analisi multipla (indicata come *batch analysis*) permette quindi di confrontare, a parità di parametri, più immagini con un'immagine predefinita.

8.2.1 Parametri per l'analisi comparata. Per l'analisi multipla le operazioni procedono nel seguente ordine:

1. Scegliere i parametri per l'analisi (ad esempio, per l'analisi di confidenza);
2. Selezionare l'immagine predefinita;
3. Utilizzare il comando per l'analisi multipla. In questo caso verrà chiesto di:
   (a) Selezionare le varie immagini da confrontare con l'immagine predefinita;
   (b) Selezionare una cartella di *output* dei risultati.

In Figura 8.8 è possibile vedere le prime fasi dell'analisi multipla.

Diagramma 8.7: Analisi multipla



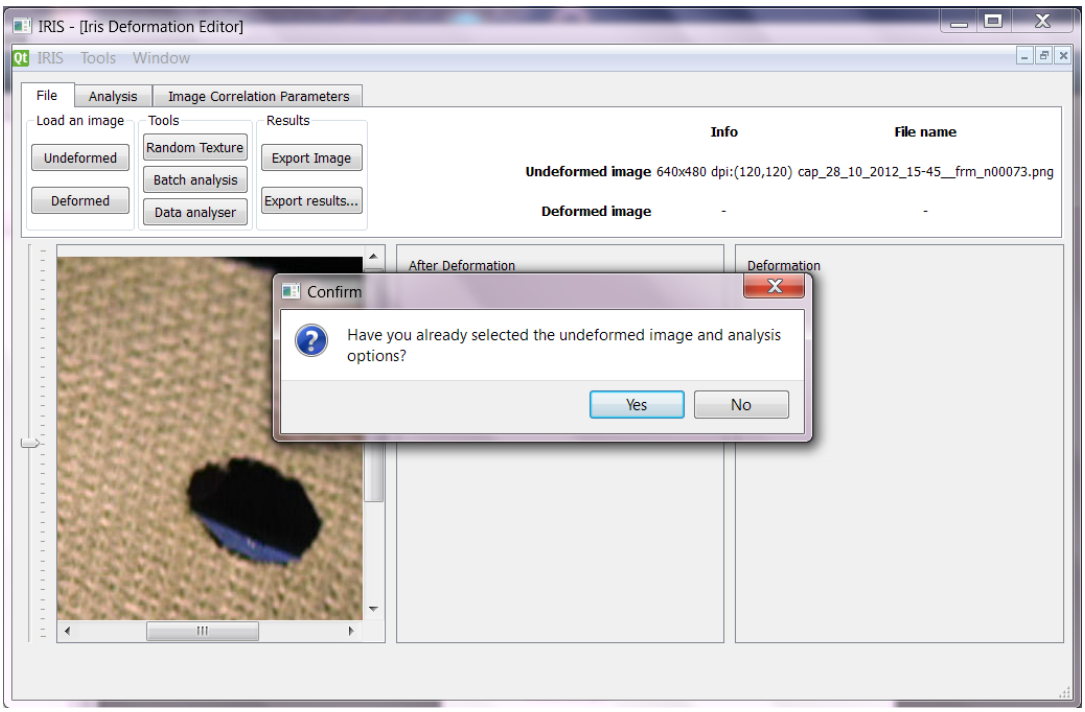


Figura 8.8: Modulo *Deformation From Image*: avvio dell'analisi multipla.

L'analisi multipla è progettata per l'utilizzo contemporaneo del programma da parte dell'utente. Questa funzionalità risulta utile nelle circostanze in cui le immagini da analizzare sono molte e quindi l'analisi multipla impiegherà ore.

8.2.2 Output dei dati. Man mano che le immagini vengono analizzate i risultati sono salvati nella cartella selezionata. Sono memorizzate due tipi di informazioni: la rappresentazione a colori e i dati in formato CSV.

Per la rappresentazione a colori, il parametro riportato è quello scelto dall'utente al momento dell'avvio dell'analisi multipla. Come per l'analisi singola è opportuno, prima di avviare l'analisi di traslazione, effettuare un'analisi di confidenza.

I dati in formato CSV sono identici a quelli descritti per l'analisi singola. In Tabella 8.1 è riassunto il significato di ogni colonna del *file*.

La caratteristica peculiare dell'analisi multipla è, senza dubbio, il confronto tra immagini a parità di parametri.

8.3 Il *Data Analyzer*

L'analisi dei dati, soprattutto per l'analisi multipla, sarebbe lunga e laboriosa senza uno strumento adatto. Il programma è stato dotato di un *Data Analyzer* semi-automatico.

Tale strumento presenta una GUI dedicata che permette facilmente la selezione dei dati da analizzare. Per il corretto funzionamento, il *Data Analyzer* ha bisogno di lavorare con i *file* CSV prodotti dal modulo *Deformation From Image* come risultato dell'analisi (singola o multipla).

8.3.1 Immagine e regione di interesse. Lo studio dei risultati inizia con la scelta dell'immagine di stato indeformato e, in dettaglio, dalla scelta della regione che maggiormente ci interessa.

Ad esempio, supponiamo che una finestra di campionamento presenti il suo vertice alle coordinate x ed y sull'immagine A di stato indeformato. I risultati (memorizzati nel *file* CSV) ci dicono che nell'immagine B lo stesso campione non si troverà alle coordinate x ed y ma alle coordinate $x + \delta x$ ed $y + \delta y$ con δx e δy specificati del *file* dei dati.

In generale, non siamo interessati ai δx e δy di tutte le finestre di campionamento dell'immagine di stato indeformato, ma soltanto ai loro valori per una determinata regione di interesse. In un'immagine di un pendolo, potremmo non essere interessati ad eventuali spostamenti sullo sfondo ma solo a quelli del peso.

Quindi il *Data Analyzer* ci chiede le seguenti informazioni:

- Immagine di stato indeformato. Servirà al programma per il calcolo degli spostamenti relativi;
- Regione di interesse. Sull'immagine di stato indeformato, il programma chiederà di individuare la regione a cui siamo interessati. Saranno analizzati gli spostamenti solo per tale regione;
- Dati in formato CSV. Questi dati, frutto dell'analisi singola o multipla, contengono gli spostamenti δx e δy per ogni finestra di campionamento.

In Figura 8.9 vediamo il *Data Analyzer* mentre richiede le informazioni sopra indicate.

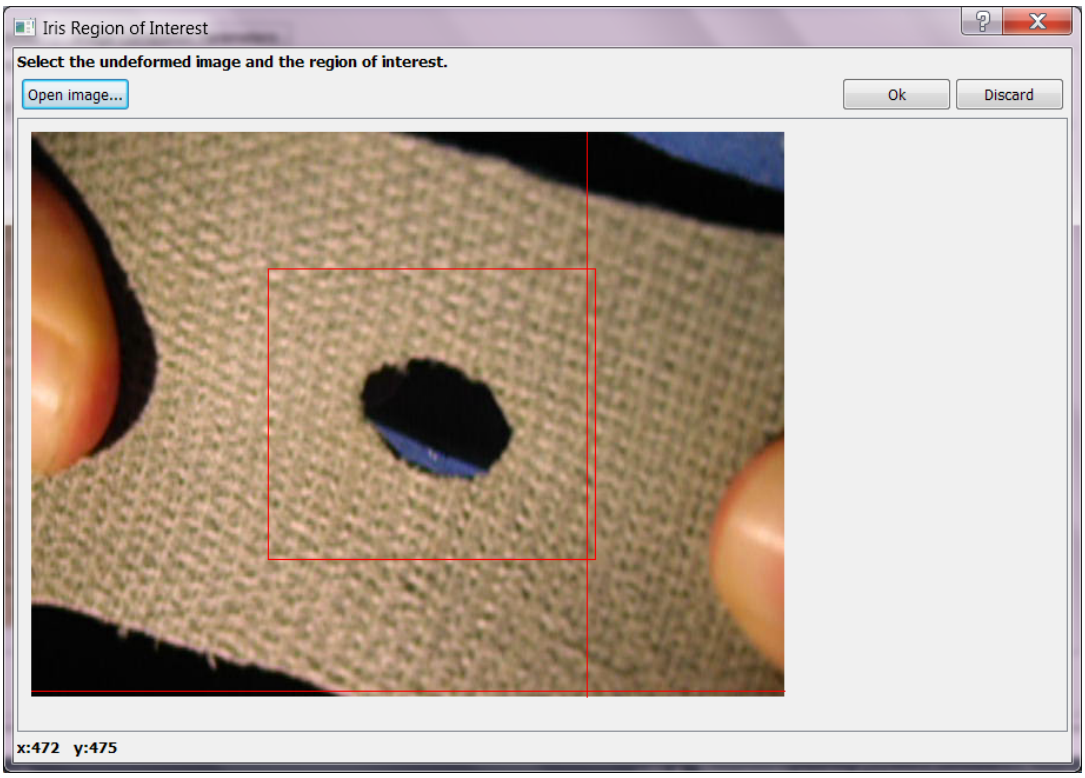


Figura 8.9: *Data Analyzer*. Selezione dell'immagine di stato indeformato e della regione di interesse.

8.3.2 Filtro dei dati. Dei dati forniti dal *Data Analyzer* verrà calcolata la media $\bar{x}$, $\bar{y}$ e lo scarto quadratico medio σ. Verrà segnalata la presenza di dati anomali che, cioè, non appartengono alla zona $[\bar{x} - 3\sigma, \bar{x} + 3\sigma]$ o $[\bar{y} - 3\sigma, \bar{y} + 3\sigma]$.

I dati anomali possono essere filtrati: considerando una distribuzione *standard* degli errori, i valori di spostamento rispetto alla media di oltre 3σ hanno probabilità pari a 0.3% di comparire. Possono essere ritenuti frutto di errori.

Il filtro dei dati, consiste quindi, nell'ignorare gli spostamenti rispetto al valore medio che vanno oltre 3σ. Questo filtro può essere applicato ricorsivamente, per ottenere dei dati senza anomalie.

L'applicazione dei filtri è a discrezione dell'utente.

8.3.3 Grafici automatici. I dati sono, infine, rappresentati graficamente. Lo strumento è molto utile soprattutto per quanto riguarda le analisi multiple: per ogni *file* verrà generata una coppia di grafici dello spostamento (lungo x ed y).

Sono ben visibili in Figura 8.10 due finestre con i grafici dello spostamento per un'analisi comparata.

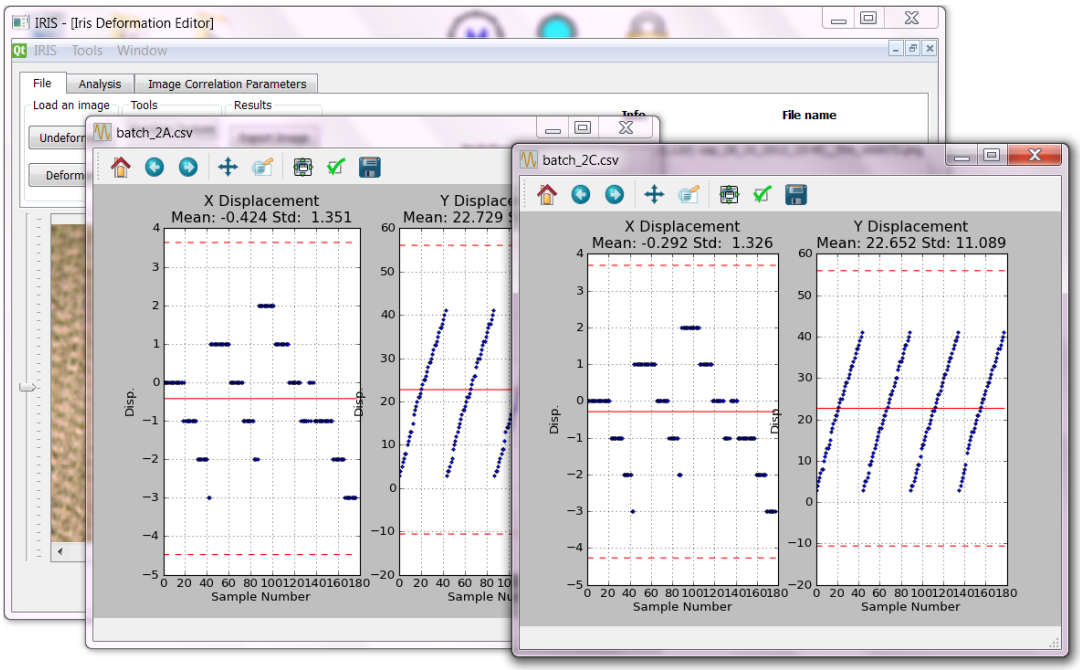


Figura 8.10: *Data Analyzer*. Grafici risultanti di spostamento per analisi multipla.

9 Modulo *Geometry From Image*

Il modulo *Geometry From Image* permette di ricavare, partendo da un'immagine, le linee base di una struttura.

Il modulo utilizza l'*Edge Detector* descritto nella Parte I. Il riconoscimento delle linee di maggiore importanza è poi migliorato, mediante un algoritmo di semplificazione.

In Figura 9.1 si può vedere la finestra principale del modulo. In alto, sono presenti i parametri per l'*Edge Detector* come descritti in Tabella 5.1.

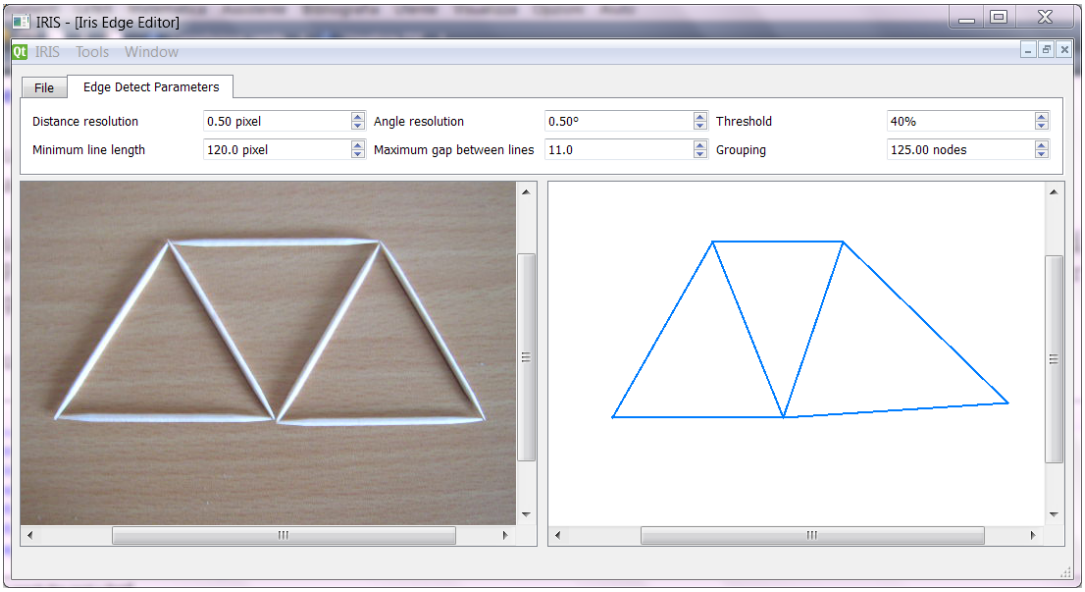


Figura 9.1: Modulo *Geometry From Image*: finestra principale

9.1 Algoritmo di semplificazione

L'*Edge Detector* ricava la geometria base in una struttura dalla sinergia tra *Canny Detector* e *Linee di Hough*. Il risultato è un *insieme* di linee base e non le *uniche* linee base.

In ausilio di queste tecniche bisogna affiancare un algoritmo di semplificazione. L'algoritmo cerca i *nodi* di maggiore importanza nella struttura.

I nodi si trovano all'intersezione di due o più linee e coincidono con gli estremi dei segmenti. L'*Edge Detector* fornisce i punti d'estremo delle linee non raggruppandoli se vicini.

L'algoritmo ha l'obiettivo di raggruppare i nodi vicini modulando il risultato mediante un parametro fornito dall'utente.

9.1.1 Individuazione dei nodi. Il primo passo dell'algoritmo è creare un elenco dei nodi. Per ogni linea, vengono analizzati gli estremi: se la lunghezza della linea è maggiore del parametro di modulazione, gli estremi vengono inseriti tra i nodi.

Successivamente, i nodi dell'elenco vengono confrontati a due a due. Se la distanza tra di essi è minore della distanza minima, uno dei due nodi è soppresso e tutti i riferimenti a tale nodo sono sostituiti.

Sono eliminati i nodi duplicati. Infine, si scorre di nuovo l'elenco delle linee. Per ogni linea, si cercano i due nodi più vicini a ciascun vertice, inserendone i riferimenti.

Il parametro di modulazione è scelto dall'utente.

9.2 Esportazione delle immagini

L'elaborazione dell'immagine produce un'immagine risultante mostrata sul lato destro dello schermo.

In Figura 9.2 vediamo come il programma ne gestisce l'esportazione.

Lo scopo dello spazio laterale è mostrare, qualitativamente, il risultato dell'analisi. Modificando i parametri si nota come cambia la geometria riscontrata.

9.3 Esportazione della geometria

Una funzionalità interessante è l'esportazione della geometria in formato `DXF`. Questo formato è utilizzato per lo scambio di disegni geometrici tra i vari *software* tecnici.

Codice 9.1: *Algoritmo di semplificazione dei nodi*

```
# Calculate the distance between two nodes
def calcGap(self, p1, p2):
  return sqrt((p1[0] - p2[0])**2 + (p1[1] - p2[1])**2)

# Find nearest node from one
def findNearest(self, sel_node, nodes, max_dist):
  for i in range(0, len(nodes)):
    if self.calcGap(sel_node, nodes[i]) <= max_dist:
      return i

  return -1

# Change reference frame
def translate(self, nodes):
  if len(nodes) <= 0:
    return []

  (minX, maxY) = nodes[0]

  for node in nodes:
    (x, y) = node

    if x < minX:
      minX = x

    if y > maxY:
      maxY = y

  new_nodes = []

  for node in nodes:
    (x, y) = node

    new_nodes.append((x - minX, maxY - y))

  return new_nodes

# Group lines
def groupingLines(self, lines, max_dist):
  nodes = []

  # Create a list of nodes
  for line in lines:
    pt1 = line[0]
    pt2 = line[1]

    if self.calcGap(pt1, pt2) > max_dist:
      nodes.append(pt1)
      nodes.append(pt2)


  # Gruping this list
  for i in range(0, len(nodes)):
    for k in range(i+1, len(nodes)):
      if self.calcGap(nodes[i], nodes[k]) <= max_dist:
        nodes[k] = nodes[i]

  # Delete duplicate values
  nodes = list(set(nodes))
  list_lines = []

  # Create line referencing the nodes list
  for i in range(0, len(lines)):
    A = self.findNearest(lines[i][0], nodes, max_dist)
    B = self.findNearest(lines[i][1], nodes, max_dist)

    if not (A == B):
      list_lines.append((A,B))

  # Delete not utilized nodes
  new_nodes = self.translate(nodes)
  list_lines = list(set(list_lines))


  return (new_nodes, list_lines, nodes)
```

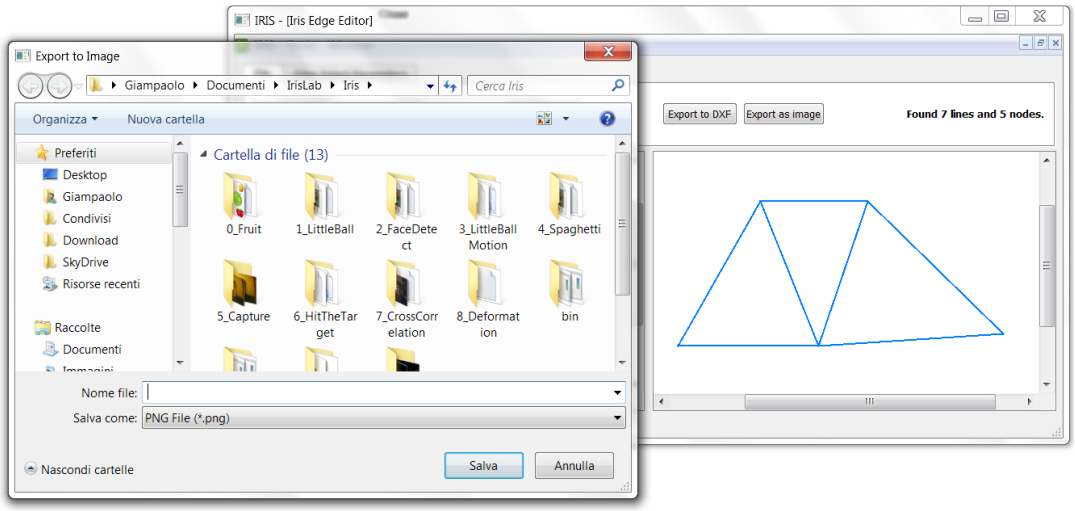


Figura 9.2: Modulo *Geometry From Image*: esportazione dell'immagine

Per il campo di nostro interesse, è possibile utilizzare il formato `DXF` in *AutoCAD* ed in *Femap*.

L'esportazione avviene partendo dall'insieme di linee e nodi. Utilizzando la libreria `dxfwrite` si genera un *file* contenente i vertici da collegare, a due a due, ed il colore del *layer*.

9.4 Problematiche connesse al metodo

Il modulo *Geometry From Image* raggiunge l'obiettivo con sufficiente precisione. La qualità della geometria dipende fortemente dai parametri dell'utente.

Questo porta un problema: a seconda dell'esperienza dell'utente, il risultato può essere buono o pessimo. Una variazione modesta dei parametri porta ad un cambio vistoso nel risultato: sono necessarie una serie di prove per determinare la giusta combinazione dei parametri. Il fattore umano è ancora troppo influente nel metodo, rendendolo non conveniente, in termini temporali, rispetto al disegno *manuale* della geometria.

Un miglioramento verrà introducendo un nuovo modo per scegliere i nodi: ad ogni nodo sarà associato un *peso* che ne misuri l'importanza, valutando il numero di linee che convergono nelle sue vicinanze. Solo i nodi più significativi saranno individuati, spostando l'obiettivo del metodo dalle linee base della struttura ai nodi.

Parte III

Applicazioni

10 Testing del programma

Nella Parte II è stata descritta la realizzazione, graduale, del programma.

In questa Parte III verrà descritto come il programma è stato sottoposto ad una serie di test, per accertarsi del suo corretto funzionamento.

É stato effettuata una prova per ciascun modulo. In particolare si verificato il corretto funzionamento per tutti gli aspetti legati all'ingegneria strutturale.

10.1 Grandi spostamenti: il moto del pendolo

La prima verifica ha riguardato lo studio del moto di un pendolo. L'obiettivo è stato raggiunto utilizzando il modulo *Experiment* con tipologia `DETECT`.

Gli aspetti approfonditi sono stati:

- Analisi e registrazione video;
- Numero di campioni positivi e negativi necessari;
- Campo dei grandi spostamenti: il pendolo poteva oscillare liberamente;
- Verifica automatica di complanarità del moto;
- Analisi in frequenza: il *software* ha individuato, automaticamente, la frequenza di oscillazione.

10.2 Spostamenti e deformazioni

I campi di spostamento e deformazione sono stati misurati con il modulo *Deformation From Image*. Sono state effettuate tre tipi di prove.

La prima è stata una simulazione computerizzata della deformazione di una piastra. Le altre due prove hanno riprodotto risultati di bibliografia. Sono state stimate le traslazioni e le rotazioni di un campione reale fotografato durante alcuni spostamenti controllati.

Gli aspetti approfonditi sono stati:

- Corretta stima delle deformazioni;
- Deformazione qualitativamente realistica;
- Apparato fotografico necessario per una buona accuratezza;
- Precisione nella stima degli spostamenti;
- Precisione nella stima delle rotazioni;
- Corretto funzionamento dell'analisi multipla.

10.3 Struttura reticolare

L'ultimo test ha messo alla prova il modulo *Geometry From Image*. Una struttura reticolare è stata fotografata ed analizzata.

Gli aspetti approfonditi sono stati:

- Scelta dei parametri;
- Influenza dell'algoritmo di semplificazione;
- Accuratezza ottenuta.

11 Il moto del pendolo

In questo esperimento il programma ha studiato, automaticamente, il moto di un pendolo ripreso da una webcam.
Gli spostamenti del pendolo possono essere anche di grande entità. Questa considerazione ha spinto per l'adozione di Experiment *di tipo* `DETECT` *per risolvere il problema.*

11.1 Obiettivi

L'obiettivo principale dell'esperimento è stato verificare il corretto funzionamento del modulo *Experiment* per lo studio dei grandi spostamenti nodali, utilizzando un pendolo.

Creando un nuovo *Experiment* di tipo `DETECT` sono stati forniti un numero sufficiente di campioni positivi e negativi affinché il programma riconoscesse una pallina, l'elemento fondamentale del pendolo.

Registrando un video con una webcam collegata al computer, sono stati calcolati gli spostamenti in funzione del tempo.

Il *Motion Analyzer* ha svolto il resto del lavoro analizzando i dati e ricavandone tutte le informazioni possibili.

In questo modo si è potuto capire il numero di campioni necessari, il tempo richiesto per il training, la qualità dell'apparato fotografico per effettuare un'analisi del moto soddisfacente.

Infine, si è stabilità l'accuratezza di questo metodo non invasivo.

Caratteristica	Dimensione
Processore	Intel Core I3 M350
Frequenza (GHz)	2.27
RAM	4 GByte

Tabella 11.1: Caratteristiche *hardware* del calcolatore utilizzato nella prova.

11.2 Apparato strumentale

L'apparato strumentale è stato raggruppato in due categorie: una informatica, l'altra fotografica.

Della prima abbiamo già accennato: per svolgere queste analisi istantaneamente è necessario un computer relativamente recente. In Tabella 11.1 sono riportate le caratteristiche del *laptop* utilizzato per la prova.

In Figura 11.1 si può vedere il computer in azione mentre esegue il programma.

Figura 11.1: Il *laptop* mentre esegue il programma.

Caratteristica	Dimensione
Quadro	640x480 `pixel`
Frequenza (FPS)	30
Protocollo	USB

Tabella 11.2: Caratteristiche *hardware* della webcam utilizzata nella prova.

Alla seconda categoria appartengono le webcam o le fotocamere utilizzate.

11.2.1 Dispositivo fotografico. Per l'acquisizione video è stata utilizzata una webcam (Figura 11.2) collegata al computer. Il quadro di ripresa presenta una dimensione pari a 640x480 `pixel` con FPS pari a 30.

Figura 11.2: La webcam utilizzata per l'esperimento.

In Tabella 11.2 sono presenti in dettaglio le caratteristiche tecniche.

11.2.2 Dimensioni del pendolo. Il pendolo è composto da una palla da *tennis da spiaggia*. Il diametro della pallina è di 4 centimetri.

La lunghezza complessiva del pendolo è stata calcolata in base alla legge fisica del pendolo: è stata scelta una lunghezza corrispondente ad un periodo di oscillazione di circa 2 secondi.

$$T = 2\pi\sqrt{\frac{l}{g}} \rightarrow l = g\left(\frac{T}{2\pi}\right)^2 \tag{11.1}$$

La lunghezza complessiva è stata quindi stimata in 102.5 centimetri. In Figura 11.3 e 11.4 è possibile vedere il pendolo come appare da lontano e nel dettaglio.

Figura 11.3: Vista complessiva del pendolo.

Il periodo atteso per questa lunghezza è di 2.03 secondi.

11.3

Procedimento

Inizialmente, il procedimento è stato simile a quello descritto nella Parte II. La gestione dell'esperimento ha seguito le fasi già note.

La fase di *apprendimento* (training) è terminata quando il programma ha riconosciuto la pallina con confidenza tale da avere una bassissima percentuale di falsi positivi.

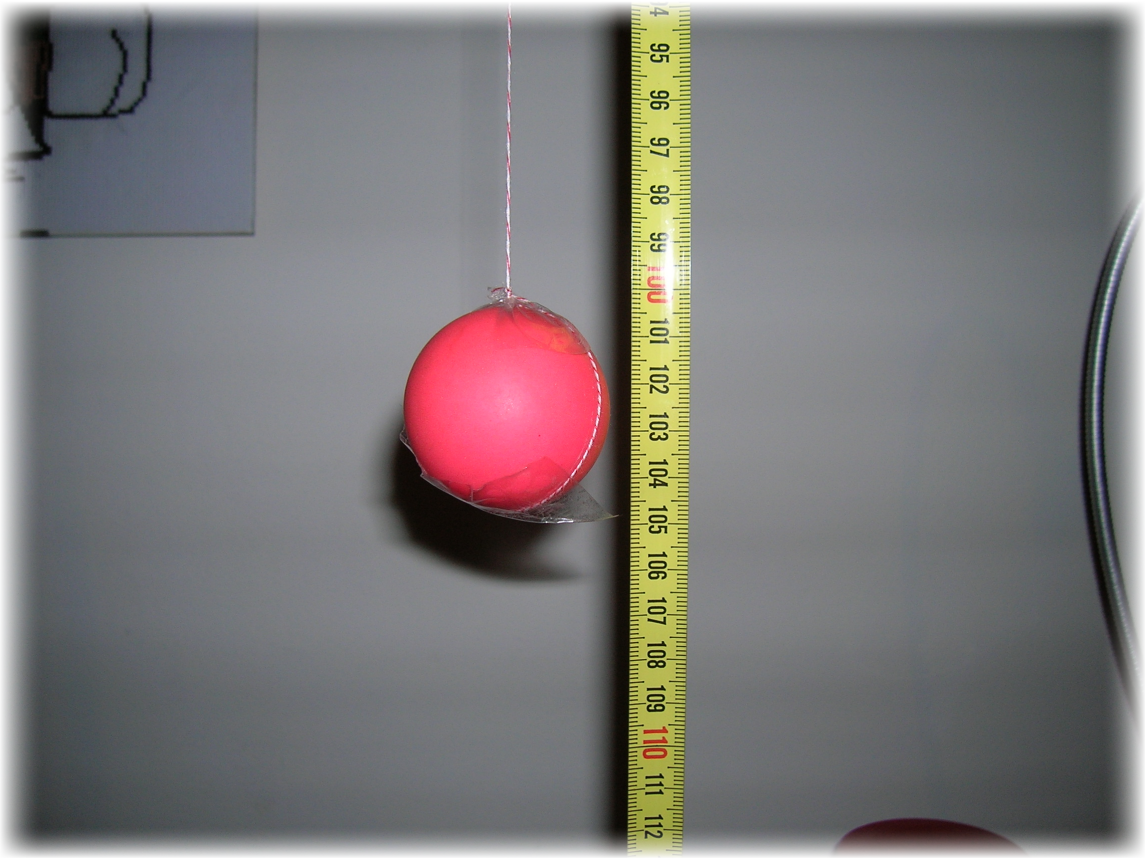


Figura 11.4: Vista in dettaglio del pendolo con indicazione della lunghezza complessiva.

Tipo di campioni	Numero	Dimensione complessiva
Positivi	616	12.6 `MByte`
Negativi	932	341 `MByte`

Tabella 11.3: Riepilogo dei campioni utilizzati per l'esperimento.

11.3.1 Creazione dell'esperimento. Inizialmente è stato creato, utilizzando la *Wizard*, un *Experiment* di tipo `DETECT`. Successivamente, è iniziata la raccolta dei campioni positivi e negativi, seguendo le indicazione in [5].

Per i campioni negativi, è importante raccogliere delle immagini che rispecchino gli sfondi dell'esperimento, sotto le più disparate condizioni di illuminazione.

Per i campioni positivi, la raccolta è stata maggiormente complicata. La pallina è stata fotografata sotto i più disparati punti di vista e variando la condizione di illuminazione, come suggerito in [6].

Dopo la prima raccolta, è stato effettuato due volte il procedimento *bootstrap* per i campioni negativi e positivi.

Nella Tabella 11.3 è presente un riepilogo del numero di campioni utilizzati.

11.3.2 Training. La fase del *training* è stata sicuramente la più complessa per il calcolatore. Tale fase è durata tra le 3 e le 4 ore ed è stata ripetuta più volte.

I campioni inizialmente raccolti fornivano un buon livello di riconoscimento per la pallina, ma il numero di falsi positivi era ancora inaccettabile.

Applicando il metodo del *bootstrap* i campioni (positivi e negativi) sono aumentati rendendo, di fatto, necessaria una nuova fase di *training*.

Il risultato del training è stato un *cascade file* di 167 `KByte`.

11.3.3 Registrazione del moto. L'ultima fase dell'*Experiment*, prima di passare all'analisi dei dati, è stata la registrazione del video.

Il pendolo è stato inquadrato per circa 1288 secondi (pari a 21 minuti e 28 secondi) registrandone i movimenti. Il programma, in tempo reale, ha analizzato i fotogrammi memorizzando gli spostamenti del pendolo in un *file* CSV.

La frequenza di registrazione, pari a 2 `Hz` ovvero 2 FPS, risulta maggiore di un fattore 4 alla frequenza prevista di oscillazione. Il teorema di Nyquist ci assicura che il campionamento è svolto correttamente.

Nella Figura 11.5 si può vedere la successione di tre fotogrammi che descrivono il moto del pendolo. Il rettangolo in rosso indica che il programma ha individuato (correttamente) la pallina e ne sta registrando i movimenti.

11.4 Analisi dei dati

Terminata la registrazione e, contestualmente, l'analisi del moto, i dati sono stati esaminati con un apposito modulo in linguaggio Python.

11.4.1 Oscillazioni registrate. In Figura 11.7 possiamo vedere il grafico delle oscillazioni in funzione del tempo. Lo studio è stato effettuato per i primi 500 secondi di oscillazione.

11.4.2 Complanarità. Sui dati delle oscillazioni è stata effettuata la verifica di complanarità. Tale verifica è effettuata sul rapporto tra la larghezza dell'oggetto rilevato (la pallina) e il quadro di registrazione.

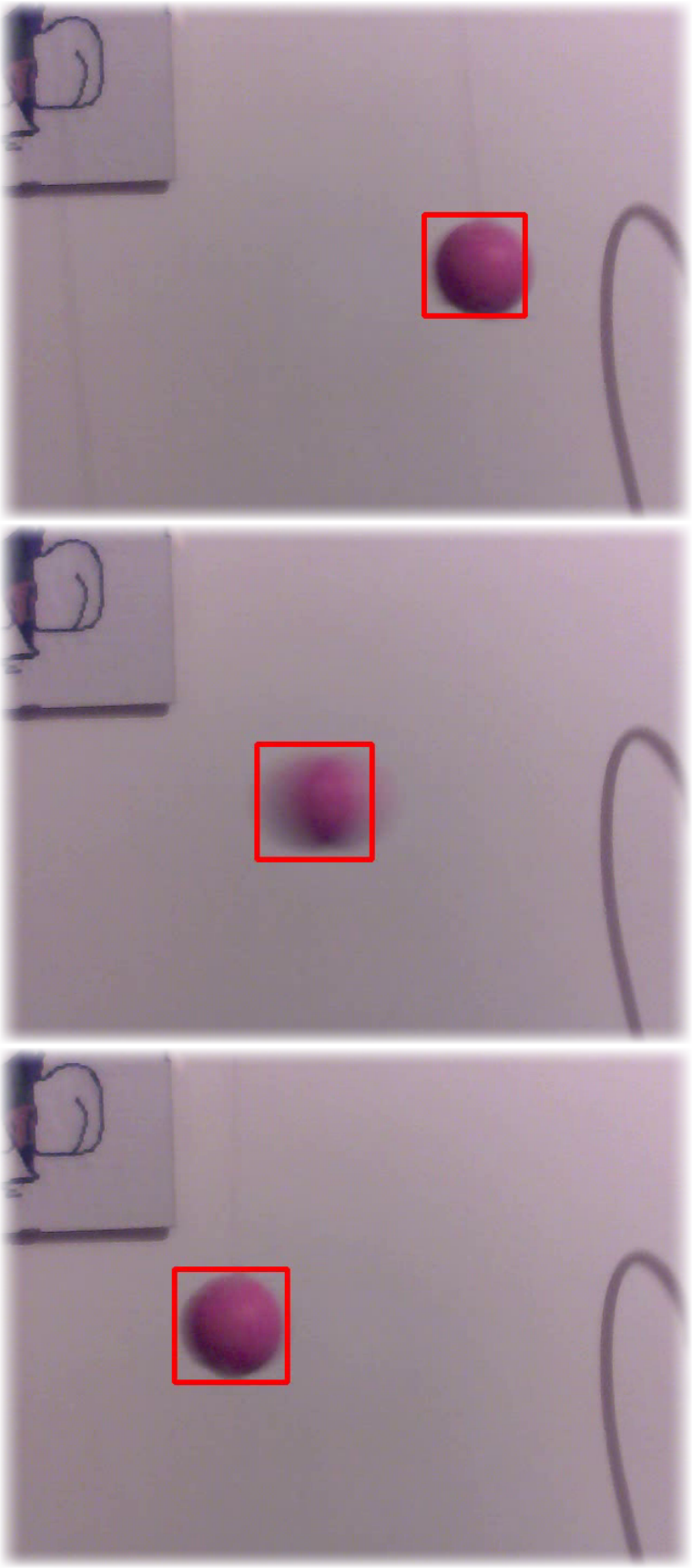
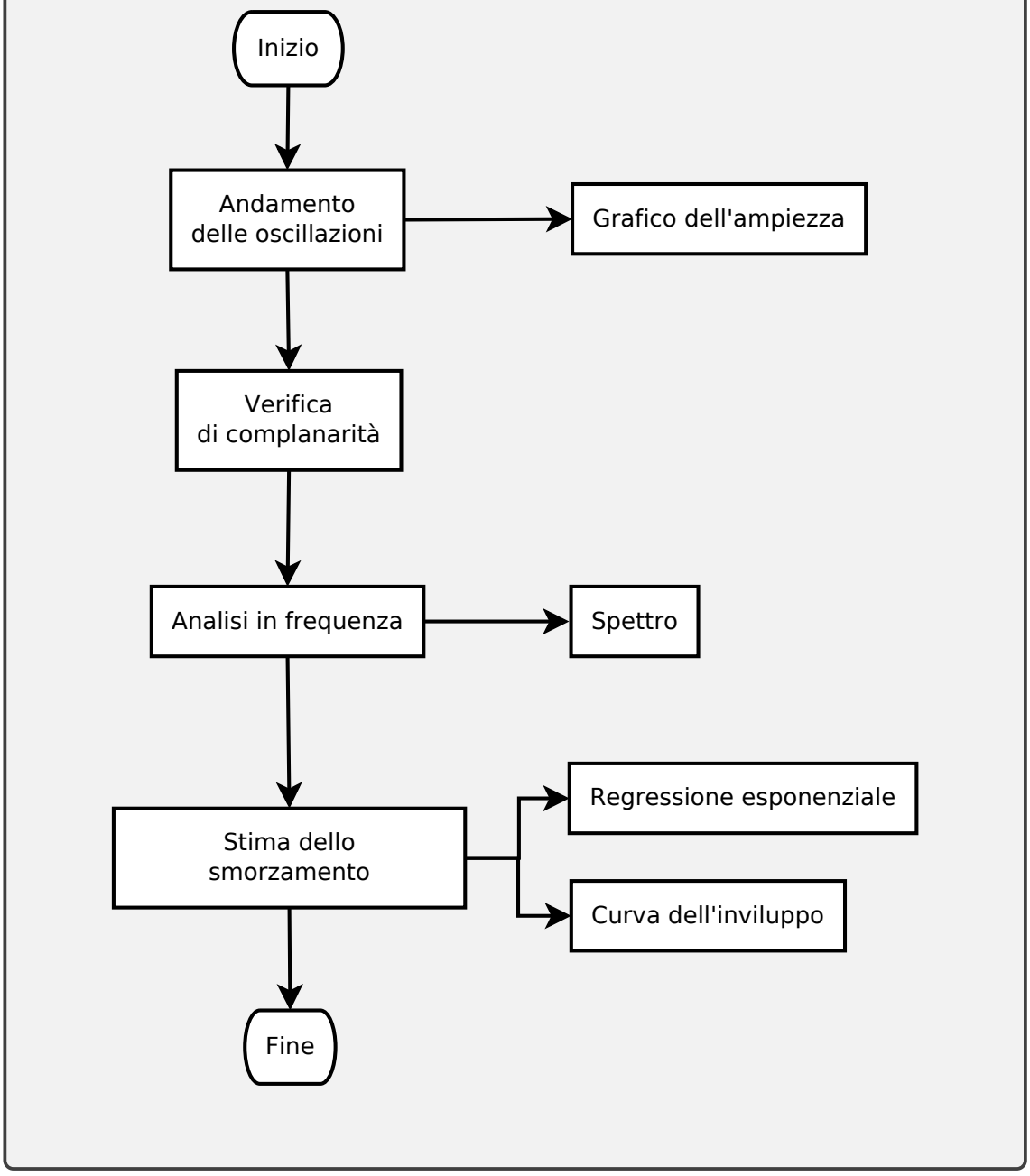


Figura 11.5: Tre fotogrammi successivi che mostrano il pendolo in moto. Il rettangolo rosso indica che il programma ne ha registrato, con successo, il movimento.

Diagramma 11.6: Schema per l'analisi dei dati nell'esempio del pendolo



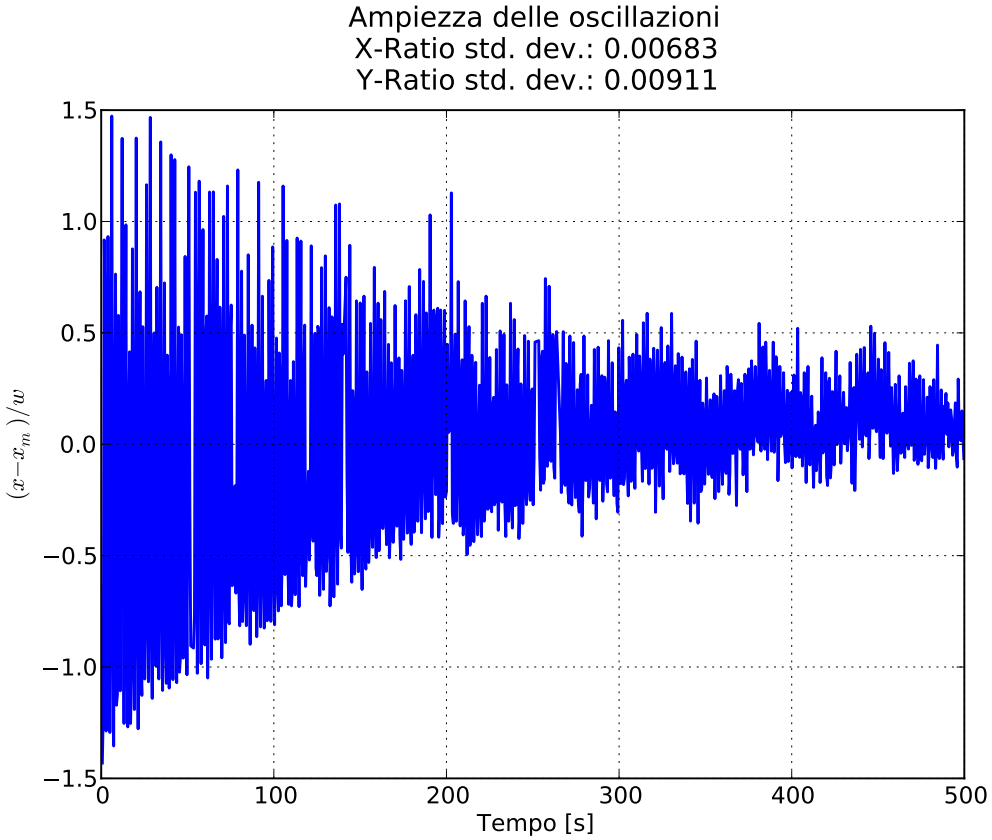


Figura 11.7: Ampiezza delle oscillazioni.

Se il moto avviene in un piano perpendicolare all'obbiettivo fotografico, tale rapporto non varia nel tempo. Al contrario, se il moto della pallina non avviene nel piano si avrà una variazione non trascurabile.

La verifica è quindi condotta calcolando la deviazione standard del rapporto tra la larghezza del pendolo e la larghezza del quadro di ripresa. La deviazione risulta pari a 0.00683 confermando la complanarità.

In Figura 11.7 vengono riportate, nel titolo, le deviazioni standard per la larghezza e per l'altezza.

11.4.3 Frequenza e periodo. L'analisi in frequenza è stata condotta mediante l'algoritmo FFT (*Fast Fourier Transformation*).

In Figura 11.8 è visibile lo spettro normalizzato rispetto al valore massimo.

I picchi, compresi nell'intervallo di frequenze tra 0.4 e 0.6 `Hz`, riportano un periodo di oscillazione pari a 2.07 secondi, come indicato nel grafico.

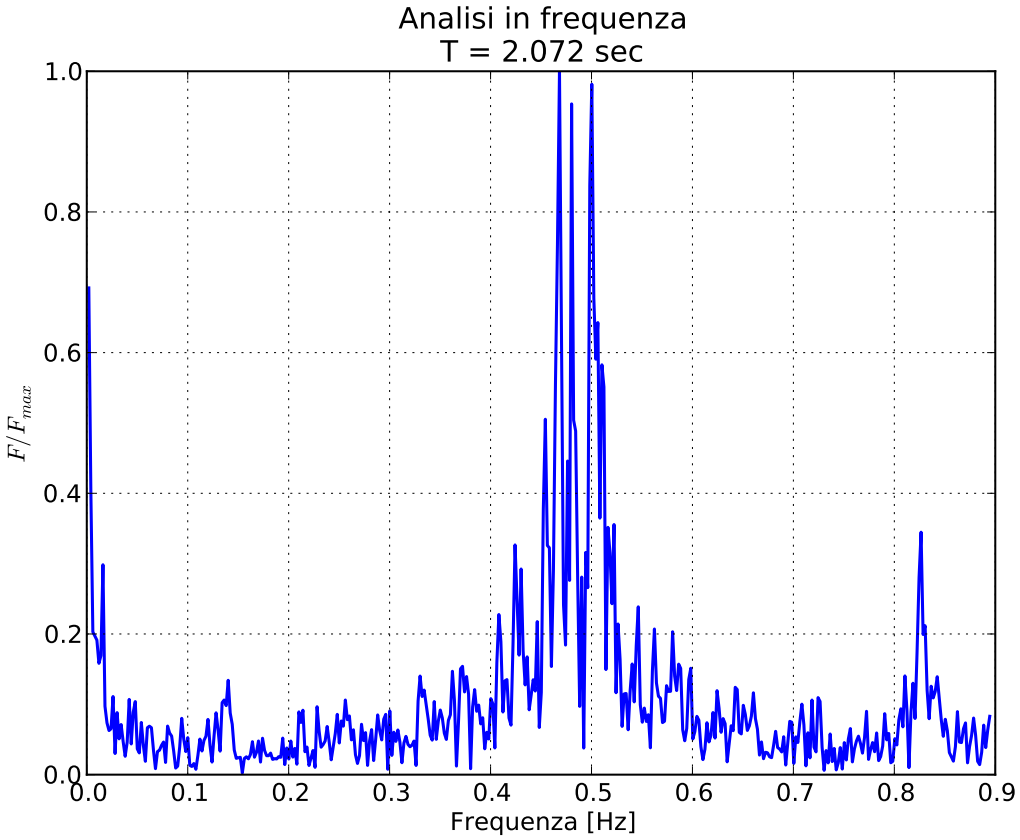


Figura 11.8: Spettro dell'analisi in frequenza.

Confronto con la legge fisica. La legge fisica (eq. 11.1) prevede un periodo pari a 2.03 secondi. Considerando i possibili errori nella realizzazione pratica del pendolo, una divergenza pari a 1.9% sembra accettabile.

11.4.4 Stima dello smorzamento. La legge del pendolo, prevede un decadimento esponenziale dell'ampiezza di oscillazione in funzione del tempo.

I dati relativi ai primi 500 secondi delle oscillazioni, sono stati divisi in 89 campioni. Per ogni campione è stata calcolata l'ampiezza massima (in valore assoluto).

L'insieme delle ampiezze massime per ogni campione ha permesso di effettuare una regressione esponenziale nella forma data dall'Equazione 11.2.

$$A(t) = ae^{-bt} \tag{11.2}$$

In Figura 11.9 è presente la rappresentazione grafica delle curva di regressione con indicazione dell'equazione. I parametri a e b sono rispettivamente uguali a 1.332 e 0.00313.

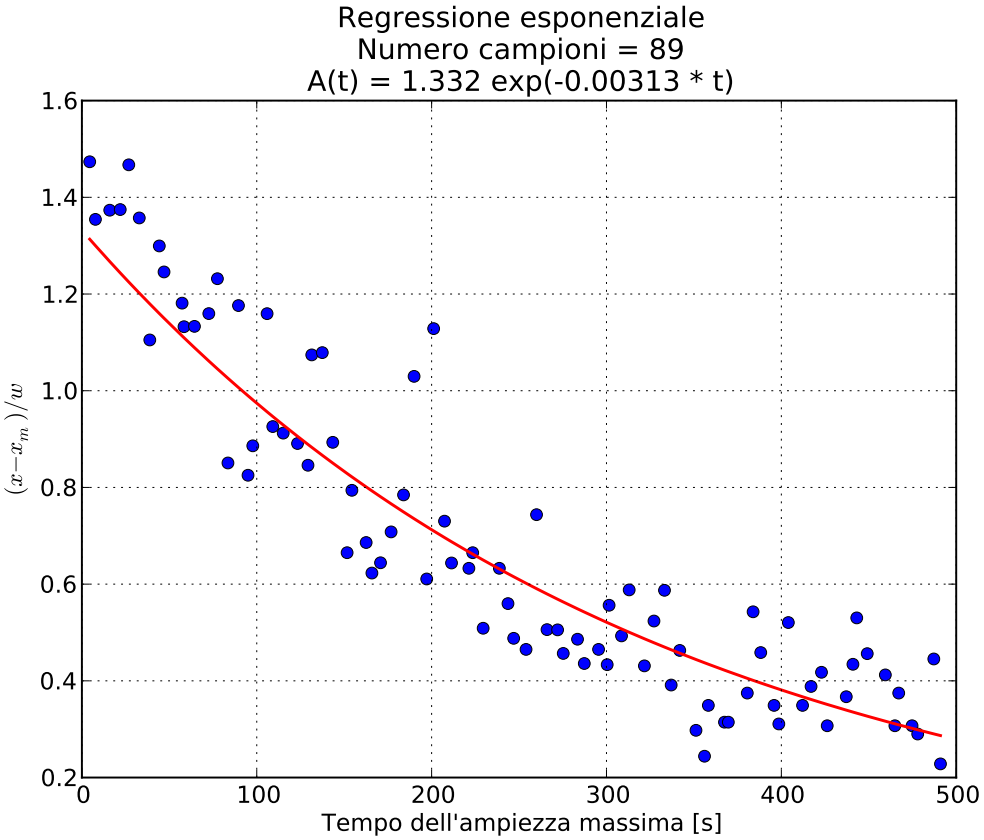


Figura 11.9: Regressione esponenziale per la stima dello smorzamento.

11.4.5 Curva dell'inviluppo. La curva indicata dall'equazione 11.2 è stata, infine, sovrapposta al grafico delle oscillazioni, ottenendo così la curva inviluppo delle ampiezze.

É possibile notare come la curva rispecchi fedelmente il decadimento dell'ampiezza delle oscillazioni del pendolo.

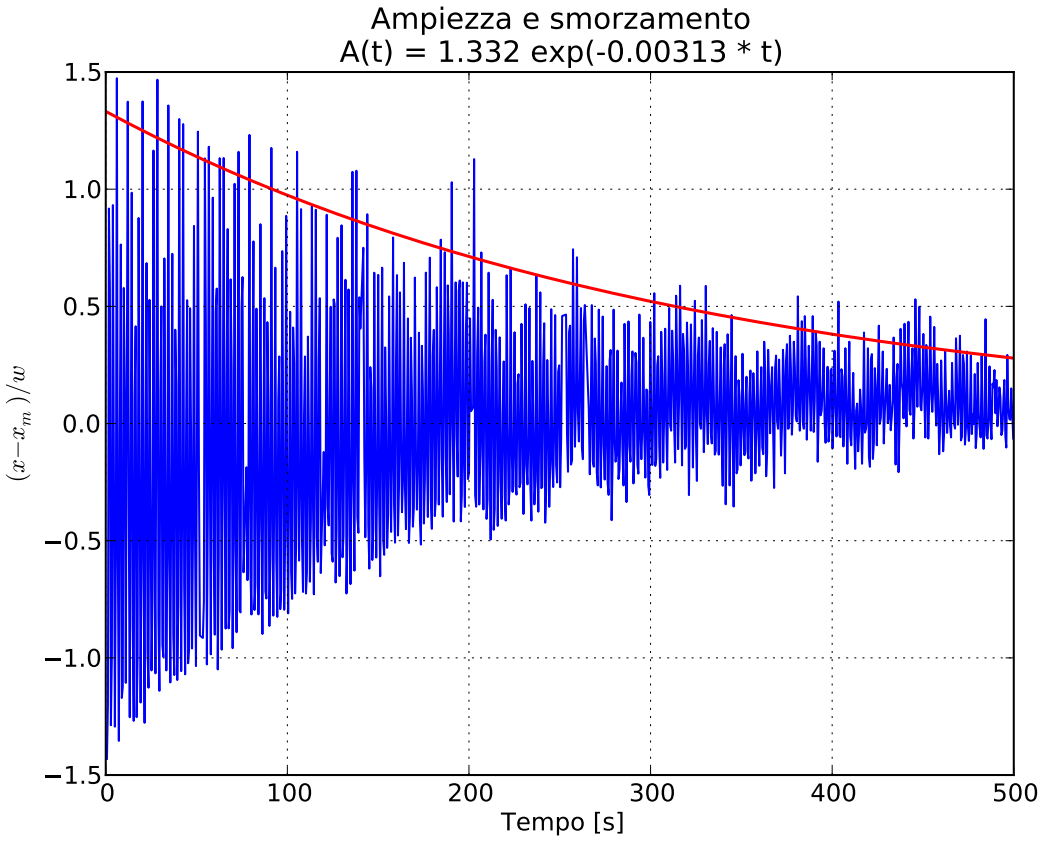


Figura 11.10: Curva inviluppo delle ampiezze.

12 Spostamento e deformazione

Con queste prove si è testato il corretto funzionamento del modulo Deformation From Image.
Le prove sono state tre: una simulazione computerizzata delle deformazioni di una piastra forata, la stima di traslazione e rotazione di un provino reale.
In tutte queste prove, il modulo si è comportato egregiamente.

12.1 Obiettivi

Lo scopo delle prove descritte in questo capitolo è stato il testare, sotto tutti i punti di vista, il modulo *Deformation From Image*.

L'entità degli spostamenti misurati varia da pochi `pixel` ad alcuni centimetri.

Il modulo, con le sue varie sfaccettature, è stato sottoposto a tre prove: una simulazione computerizzata e due stime reali.

12.2 Procedure

Le procedure seguite per le prove sono state in parte tratte dalla bibliografia. In [1] viene applicata una specifica procedura per stabilire la precisione del metodo per la stima della traslazione e della rotazione di un provino.

D'altronde la simulazione computerizzata è servita per stabilire se, qualitativamente, il programma calcola risultati accettabili.

12.3 Simulazione virtuale

La simulazione virtuale ha permesso di verificare che le deformazioni calcolate dal programma siano qualitativamente esatte.

A questo scopo si è utilizzato il *Random Texture Generator* per creare l'immagine di una piastra forata con motivo casuale.

12.3.1 Piastra virtuale indeformata. La piastra indeformata ha dimensioni pari a 600x350 `pixel`. La *random texture* ha densità del 6% con macchie simulate di 2 `pixel`.

In Figura 12.1 è mostrata la piastra. Si può chiaramente notare il motivo casuale su di essa.

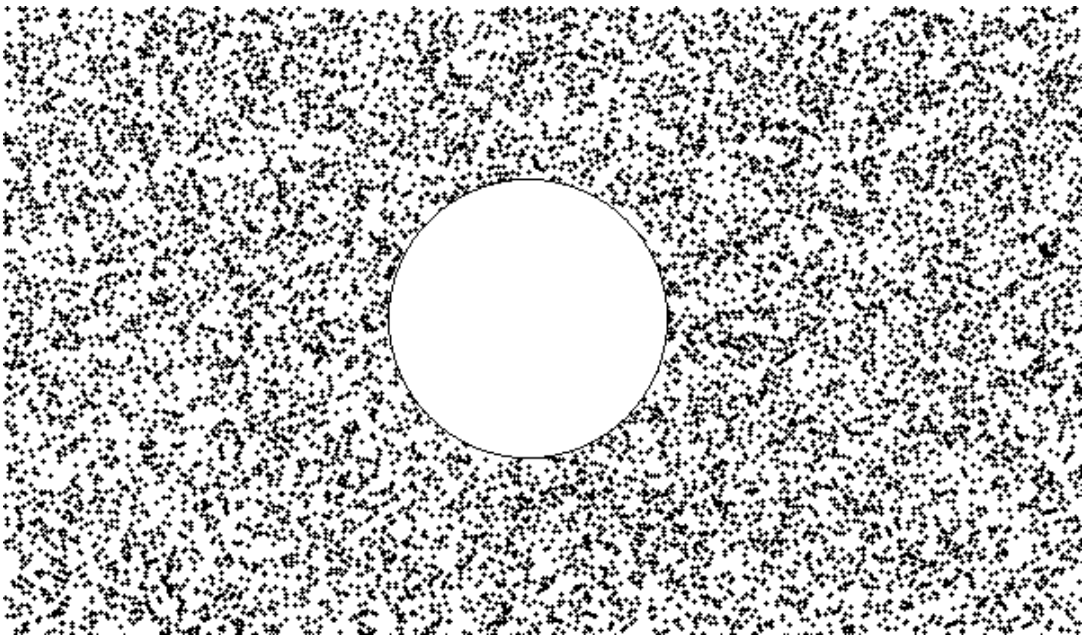


Figura 12.1: Piastra virtuale indeformata.

La piastra è stata, successivamente, deformata in tre modi diversi utilizzando il *software* CorelDraw X2.

12.3.2 Caso A. In questo caso, la piastra ha subito una deformazione del 103% in larghezza e del 97% in altezza. Le dimensioni della piastra deformata, sono di 618x340 `pixel`.

In Figura 12.2 è possibile vedere la piastra indeformata, la piastra deformata ed il campo di deformazione calcolato dal programma.

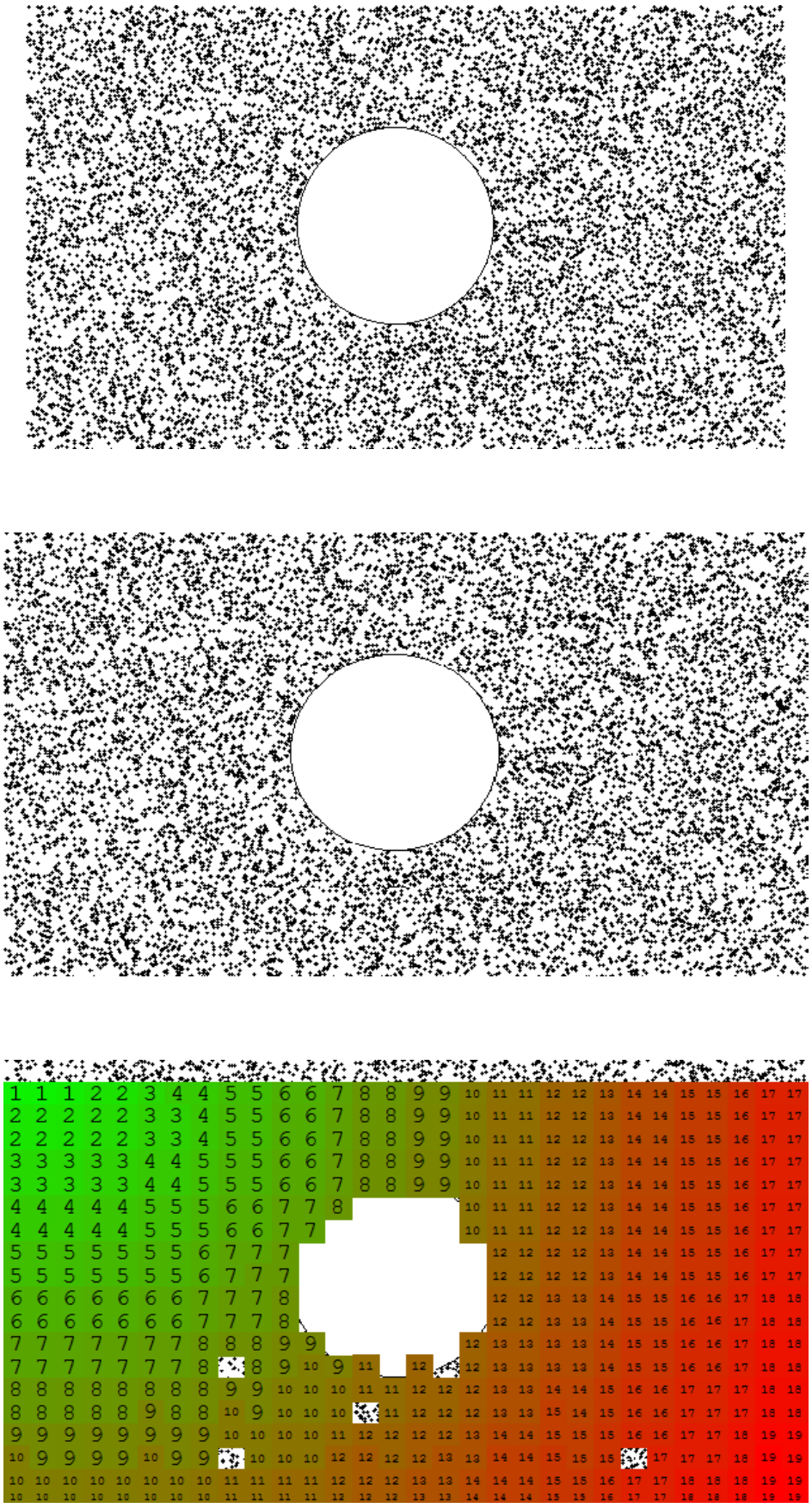


Figura 12.2: Deformazione nel caso A: 103% in larghezza e 97% in altezza. É visibile la deformazione totale calcolata dal programma.

Caso	Def. imposta	Def. totale	Def. stimata	Confidenza
A	18px (larghezza) e -10px (altezza)	20px	19px	29%
B	12px (larghezza) e -4px (altezza)	12px	11px	29%
C	48px (larghezza) e -28px (altezza)	55px	54px	29%

Tabella 12.1: Riassunto dei risultati ottenuti dalla simulazione computerizzata.

12.3.3 Caso B. In questo caso, la piastra ha subito una deformazione del 103% in larghezza e del 99% in altezza. Le dimensioni della piastra deformata, sono di 612x346 `pixel`.

In Figura 12.3 è possibile vedere la piastra indeformata, la piastra deformata ed il campo di deformazione calcolato dal programma.

12.3.4 Caso C. In questo caso, la piastra ha subito una deformazione del 108% in larghezza e del 92% in altezza. Le dimensioni della piastra deformata, sono di 648x322 `pixel`.

In Figura 12.4 è poissibile vedere la piastra indeformata, la piastra deformata ed il campo di deformazione calcolato dal programma.

12.3.5 Risultati e confronto qualitativo. I risultati della simulazione, per i tre casi, sono riassunti nella Tabella 12.1. Le deformazioni imposte sono state calcolate correttamente dal programma.

Il campo degli spostamenti, calcolato dal programma, è stato confrontato qualitativamente con un campo di spostamenti realistico.

Per questo è stato realizzato un modello agli elementi finiti di una piastra forata. Due lati (quello in alto e quello laterale sinistro) sono stati vincolati.

Tale piastra è stata, poi, sottoposta a due condizioni di carico. Nella prima, la piastra ha subito una trazione da entrambi i lati. La deformazione così prodotta è stata confrontata con il campo degli spostamenti del caso A. Il confronto qualitativo è presentato in Figura 12.5.

Nella seconda condizione di carico, la piastra ha subito trazione solo sul lato più corto. Il campo degli spostamenti è stato confrontato con il caso B ed i risultati presentati in Figura 12.6.

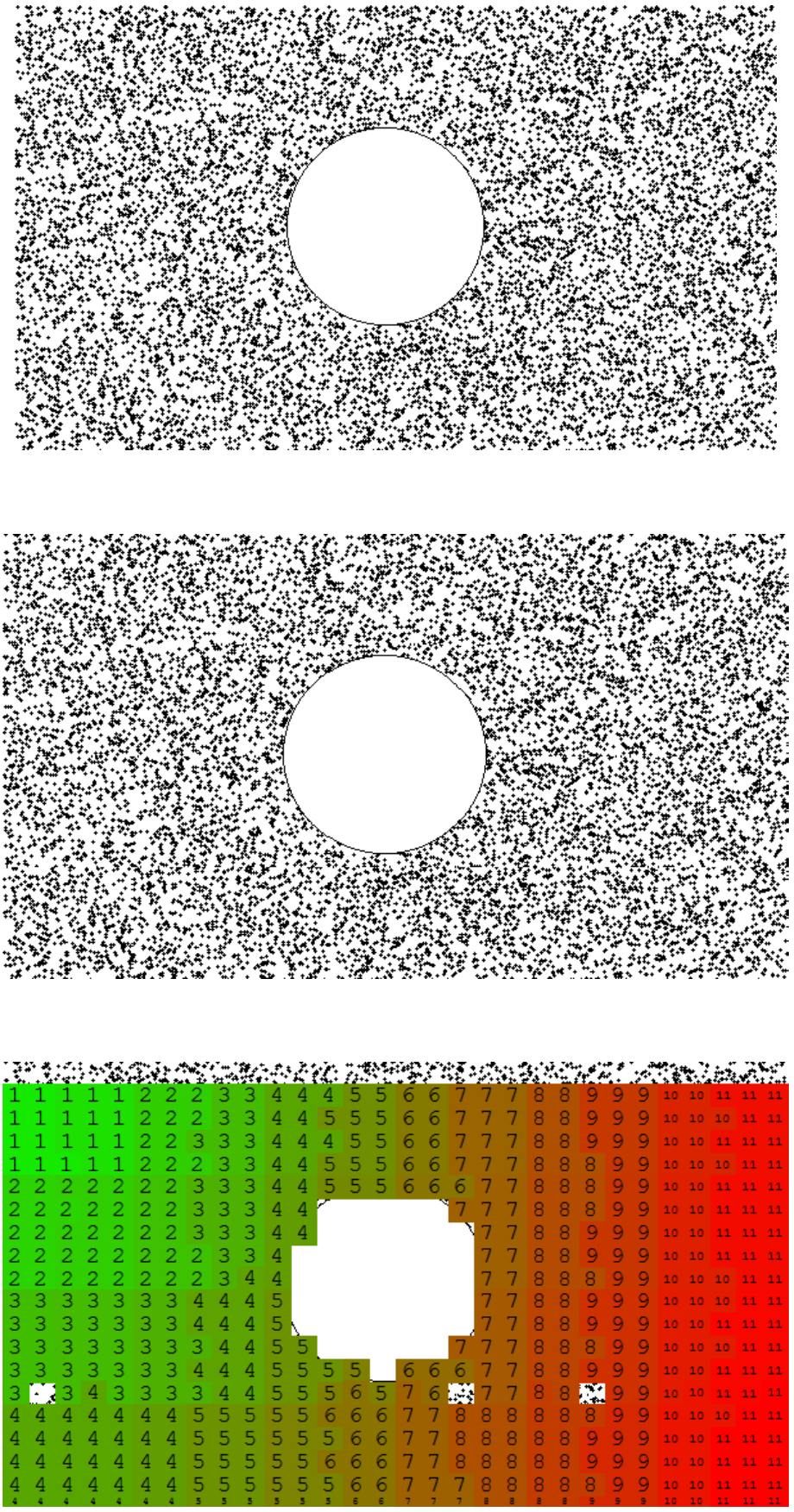


Figura 12.3: Deformazione nel caso B: 102% in larghezza e 99% in altezza. É visibile la deformazione totale calcolata dal programma.

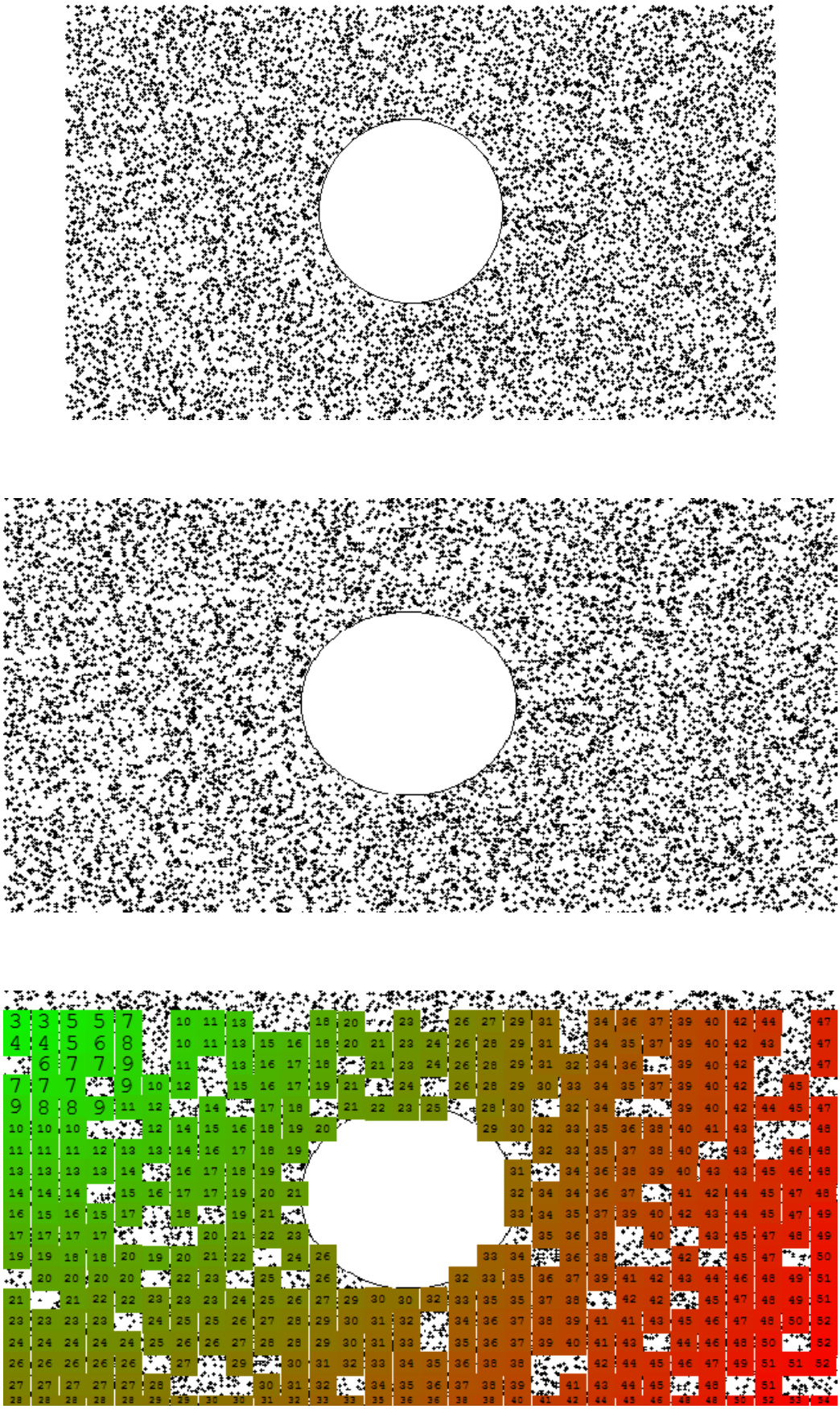


Figura 12.4: Deformazione nel caso C: 108% in larghezza e 92% in altezza. É visibile la deformazione totale calcolata dal programma.

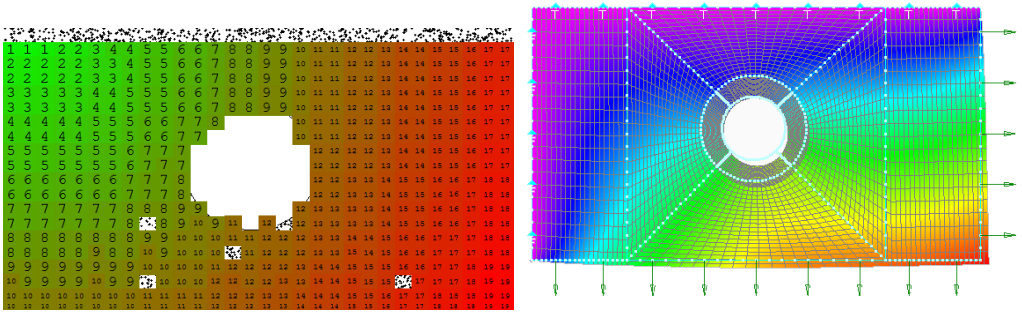


Figura 12.5: Confronto qualitativo della deformazione stimata dal programma e quella calcolata da un *software* FEM. La piastra è in trazione da entrambi i lati.

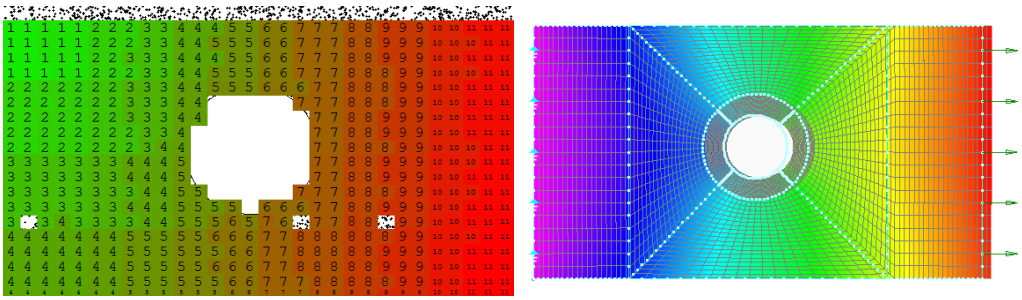


Figura 12.6: Confronto qualitativo della deformazione stimata dal programma e quella calcolata da un *software* FEM. La piastra è in trazione dal lato corto.

Come si nota, in entrambi i casi il campo di deformazione è qualitativamente ben predetto.

12.4

Validazione per la traslazione

La validazione per traslazione è avvenuta riproducendo i risultati descritti in bibliografia.

Un provino, con motivo casuale, ha subito delle traslazioni di entità controllata. Da una posizione ferma, il provino è stato spostato, lungo una sola direzione, di alcuni millimetri.

Ad ogni piccolo spostamento la sua posizione è stata registrata con un apparato fotografico, scattando 3 fotografie per ogni posizione del provino.

É stata condotta l'analisi di confidenza (per stabilire la soglia di confidenza) e l'analisi multipla. Ogni fotografia è stata confrontata con la posizione iniziale, per ottenerne lo spostamento del provino.

Come in bibliografia l'entità della traslazione è stato riportato su un grafico di confronto.

12.4.1 Apparato sperimentale. L'apparato sperimentale (Figura 12.7), utilizzato per la prova, è composto da:

- Fotocamera da 5 `MPixel` dotata di autoscatto;
- Cavalletto fotografico livellabile;
- Piano di riferimento disegnato a computer;
- Lampada alogena.

Figura 12.7: Apparato sperimentale utilizzato per la validazione della traslazione.

12.4.2 Piano di riferimento. Per applicare la traslazione al provino è stato utilizzato un piano di riferimento. Tale piano presenta una posizione iniziale, e delle barre colorate di dimensione nota. Utilizzando tali barre è possibile spostare il provino applicandogli con precisione la traslazione voluta.

Il piano di riferimento è stato disegnato al calcolatore con il *software* CorelDraw X2.

12.4.3 Registrazione delle immagini. Ad ogni incremento di traslazione, la posizione del provino è stata fotografata utilizzando l'autoscatto. L'uso dell'autoscatto ha permesso di eliminare la possibilità di spostamenti involontari da parte umana.

	Stima della traslazione			
Spostamento applicato	**Foto 1**	**Foto 2**	**Foto 3**	**Media**
3	3.026	3.012	3.004	3.014
5	5.222	5.245	5.247	5.238
10	11.067	11.043	11.053	11.054
13	14.173	14.099	14.115	14.129
18	20.027	19.995	20.031	20.017
20	22.414	22.293	22.335	22.347
30	30.714	30.723	30.750	30.729
40	42.086	42.080	42.110	42.092

Tabella 12.2: Traslazione del provino: riassunto delle prove effettuate e della stima dello spostamento.

In Figura 12.8, a titolo d'esempio, sono presenti tre fotografie. La traslazione del provino rispetto alla posizione iniziale, come si vede, è stata lentamente incrementata.

12.4.4 Analisi dei dati. L'analisi dei dati ha preso il via dall'analisi di confidenza. Successivamente, stabilità la soglia di probabilità, la dimensione delle finestre di campionamento, è stata condotta l'analisi multipla tra le varie fotografie del provino e l'istantanea alla posizione iniziale.

La dimensione delle finestre di campionamento è stata di 24x24 `pixel`.

Per ogni traslazione sono state registrate ed analizzate tre fotografie. I risultati delle tre fotografie sono stati mediati.

12.4.5 Risultati. I risultati della prova, come calcolati dal *Data Analyzer*, sono riportati nella Tabella 12.2. In Figura 12.9 sono rappresentate due rette: la prima indica lo spostamento *reale* e l'altra lo spostamento come calcolato dal programma. Come si vede, c'è buon accordo tra le due, confermando i risultati di bibliografia.

12.5 Validazione per la rotazione

La validazione per rotazione ha ricalcato i metodi descritti per la prova di traslazione.

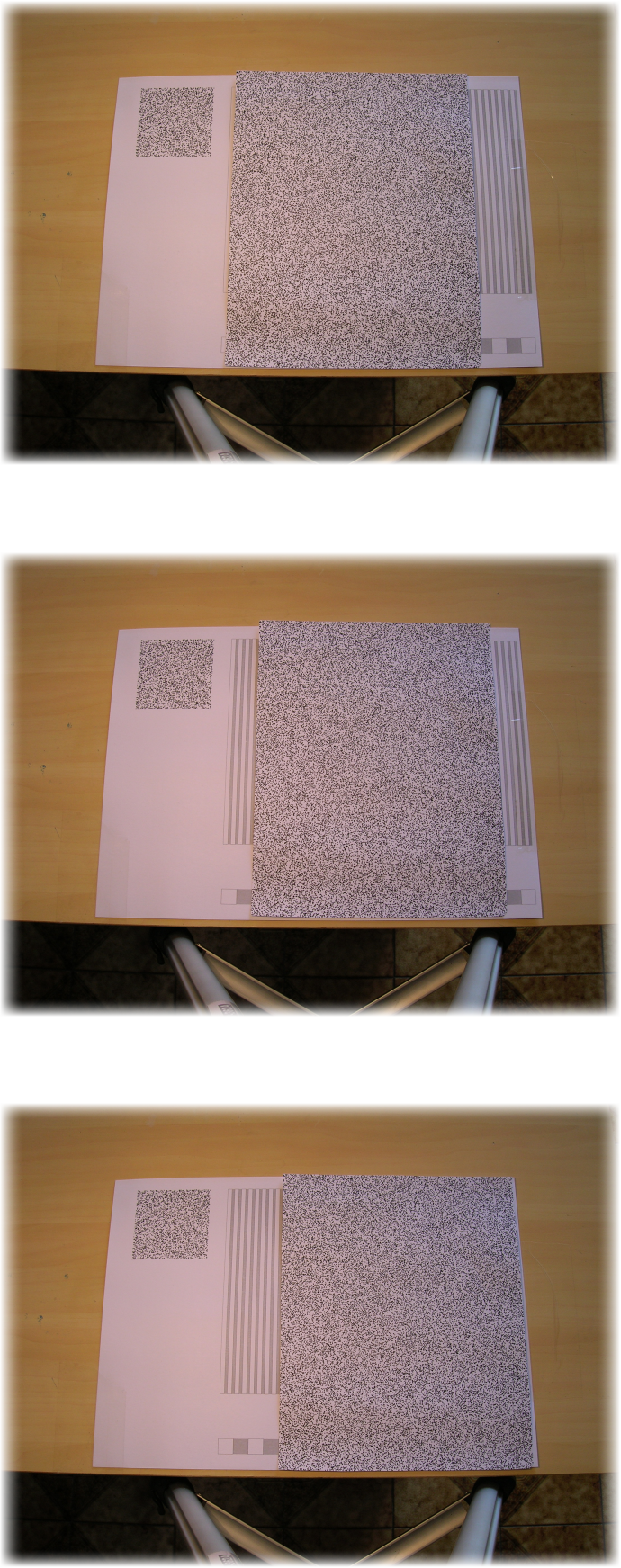


Figura 12.8: Validazione della traslazione: tre fotografie del provino con traslazioni rispettivamente di 3, 20 e 40 millimetri.

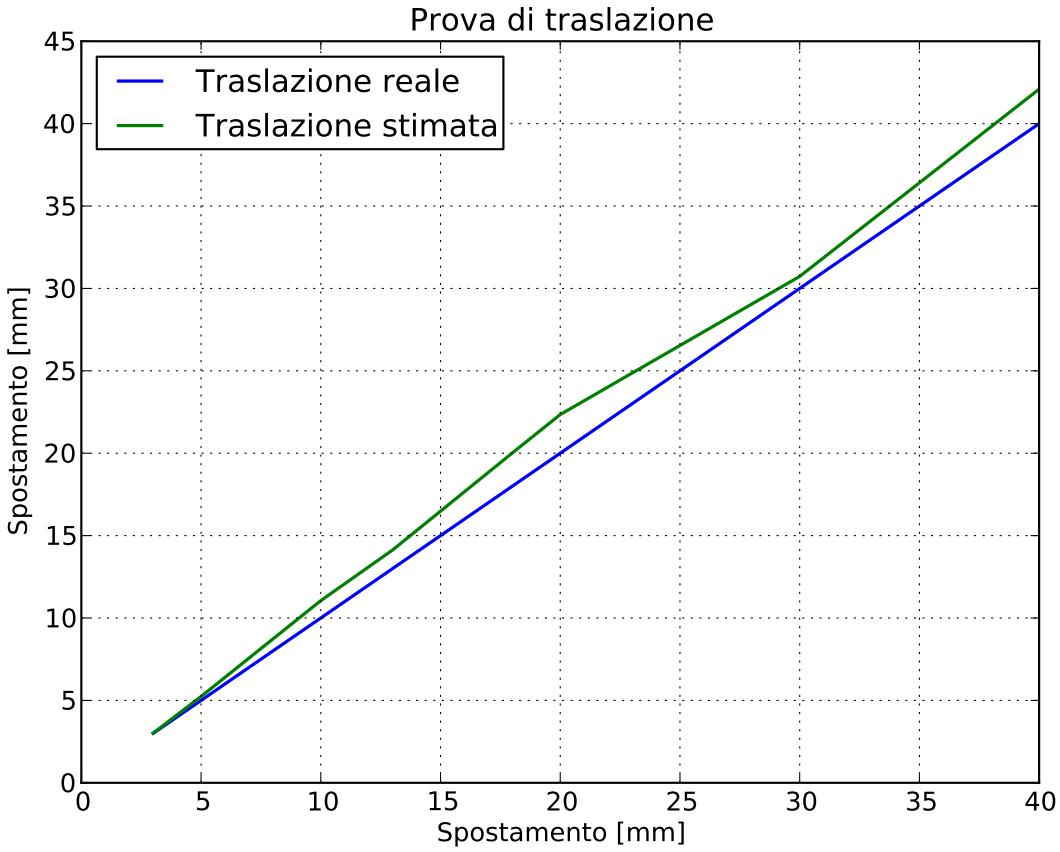


Figura 12.9: Traslazione del provino: confronto tra la traslazione reale e quella misurata.

Il provino, con motivo casuale, ha subito delle rotazioni di entità controllata, sempre maggiore.

Ad ogni rotazione, sono state scattate tre fotografie (mediante lo stesso apparato fotografico della prova di traslazione).

É stata condotta, come sempre, prima l'analisi di confidenza (per stabilire la soglia di confidenza) e successivamente l'analisi multipla. Ogni fotografia è stata confrontata con la posizione iniziale.

I *file* dei dati sono stati elaborati per calcolare, scelto un polo, l'entità della rotazione.

12.5.1 Apparato sperimentale. L'apparato sperimentale utilizzato per la prova di rotazione è stato il medesimo della prova di traslazione (Figura 12.7). L'unica differenza, consiste nel piano di riferimento.

12.5.2 Piano di riferimento. Per applicare la rotazione al provino è stato realizzato un piano di riferimento. con un polo di rotazione ed una linea base. A partire da tale linea sono state tracciati dei segmenti con pendenza controllata.

Ad ogni rotazione il provino era allineato ad un segmento, in modo da essere sicuri dell'angolo di rotazione rispetto alla posizione iniziale.

Il piano di riferimento è stato disegnato al calcolatore con il *software* CorelDraw X2.

12.5.3 Registrazione delle immagini. Ad ogni incremento di rotazione, la posizione del provino è stata fotografata utilizzando, come in precedenza, l'autoscatto eliminando la possibilità di errori involontari da parte umana.

In Figura 12.10, a titolo d'esempio, sono presenti tre fotografie. La rotazione del provino rispetto alla linea base, come si vede, è stata lentamente incrementata.

12.5.4 Analisi dei dati. Stabilita la soglia di probabilità e la dimensione delle finestre di campionamento, è stata condotta l'analisi multipla tra le varie fotografie del provino e l'istantanea alla posizione iniziale.

La dimensione delle finestre di campionamento è stata di 18x18 `pixel`, maggiore della dimensione utilizzata per la prova di traslazione. Infatti, la stima della rotazione è risultata più difficoltosa.

Per ogni angolo di rotazione sono state registrate ed analizzate tre fotografie. Gli angoli sono stati ricavati utilizzando il metodo descritto nella Parte II.

12.5.5 Risultati. I risultati della prova sono riportati presentati in Tabella 12.3. In Figura 12.11 sono rappresentate due rette: la prima indica gli angoli di rotazione *reale* e l'altra gli angoli di rotazione calcolati dal programma.

Come ci si aspettava, la stima delle rotazioni oltre 5-6 gradi risulta non attendibile. Tale problema è stato evidenziato anche in [1] ed è considerata una limitazione dell'*Image Correlation*.

Figura 12.10: Validazione della rotazione: tre fotografie del provino con rotazione rispettivamente di 0, 6 e 20 gradi.

	Stima dell'angolo di rotazione			
Rotazione reale	**Foto 1**	**Foto 2**	**Foto 3**	**Media**
1	1.268	1.255	1.250	1.258
2	1.948	2.024	1.959	1.977
3	2.822	2.836	2.843	2.834
4	3.862	3.866	3.871	3.866
5	4.879	4.869	4.907	4.885
6	4.755	4.782	4.754	4.763
7	6.510	6.619	6.670	6.600
9	8.327	16.785	14.049	13.054

Tabella 12.3: Rotazione del provino: riassunto delle prove effettuate e della stima dell'angolo di rotazione.

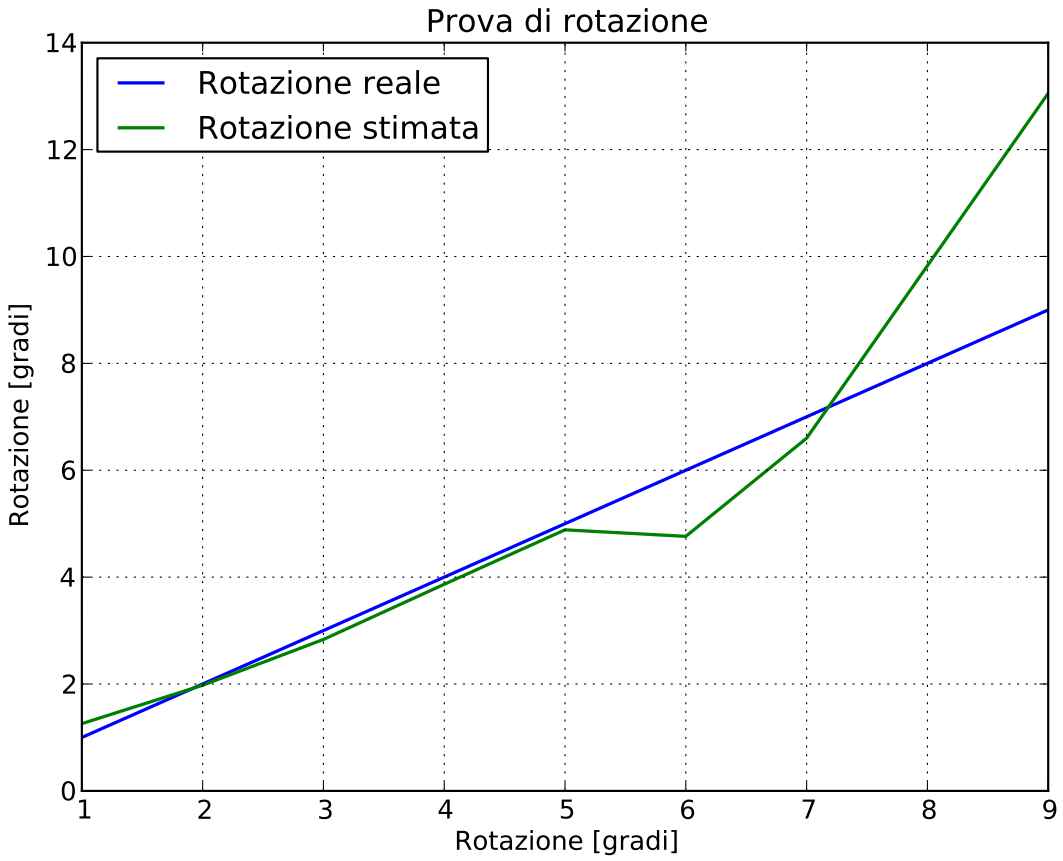


Figura 12.11: Rotazione del provino: confronto tra la rotazione reale e quella misurata.

13 Spaghetti Structure

Con questa prova si sono esplorati i limiti del modulo Geometry From Image*. Una struttura a traliccio, simulata con degli stecchini, è stata fotografata ed analizzata. Il programma ha ricavato correttamente, dopo una fase di regolazione dei parametri, le linee base della geometria.*

13.1 Obiettivi

Lo scopo di questa prova, è stato mettere alla prova il modulo *Geometry From Image*.

Come anticipato nella Parte II la tecnologia alla base (l'*Edge Detector*) non risulta ancora matura per un uso affidabile e quotidiano. Nonostante questo, impostandone opportunamente i parametri e utilizzando l'algoritmo di semplificazione si sono ottenuti dei buoni risultati.

13.2 Procedura

La prova è stata effettuata utilizzando una struttura reticolare (denominata *Spaghetti Structure*) costruita utilizzando una serie di stecchini.

13.2.1 La struttura. Utilizzando 7 stecchini (incollati tra loro) è stata costruita una tipica struttura reticolare, come mostrato in Figura 13.1.

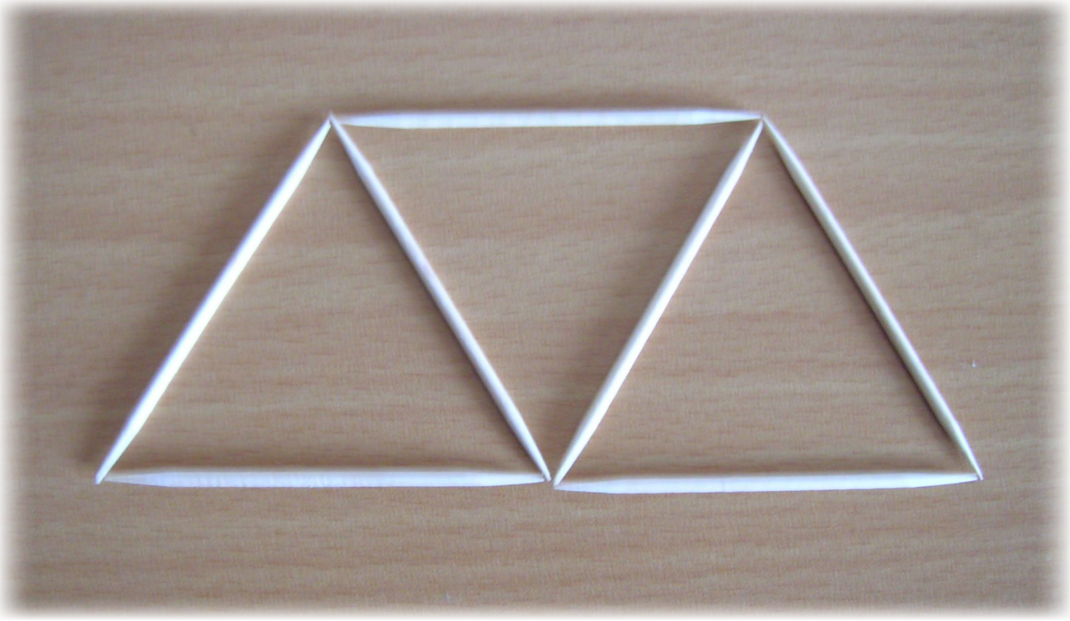


Figura 13.1: Struttura reticolare utilizzata per la prova.

Combinazione	Soglia	Lunghezza minima delle linee	Figura
A	50%	2 px	Figura 13.2
B	45%	0 px	Figura 13.3

Tabella 13.1: Struttura reticolare: parametri ottimali.

13.2.2 Apparato fotografico. L'apparato fotografico è costituito da una fotocamera da 5 `MPixel`. L'immagine è stata ridimensionata dopo l'acquisizione, per ottenere tempi di calcoli ridotti.

La dimensione dell'immagine è quella di 640x480 `pixel`, così come la dimensione dell'immagine risultante.

13.3

Analisi con *Geometry From Image*

Per la ricerca dei parametri, sono state effettuate varie prove. Tra le varie combinazioni due sono risultate promettenti.

13.3.1 Parametri ottimali. In Tabella 13.1 sono riassunte due combinazioni di parametri (denominate A e B). Con tali parametri, il risultato è nettamente migliore rispetto ad altre combinazioni.

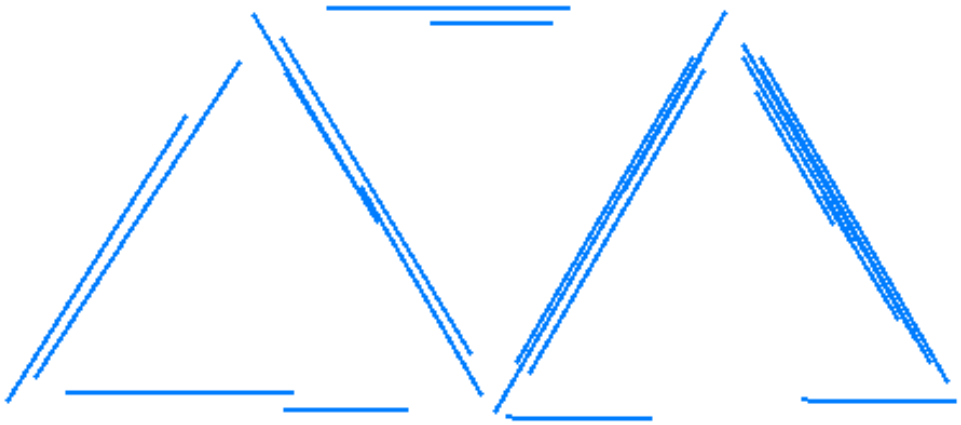


Figura 13.2: Geometria risultante con i parametri della combinazione A.

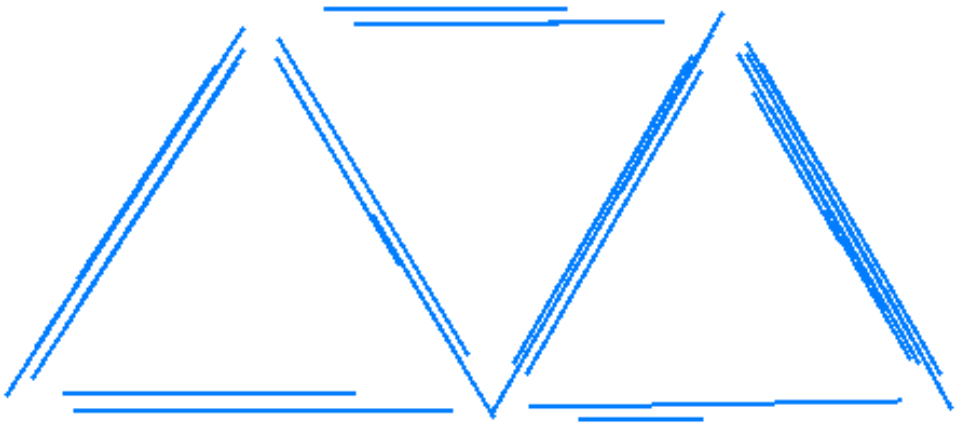


Figura 13.3: Geometria risultante con i parametri della combinazione B.

13.3.2 Algoritmo di semplificazione. Come è possibile notare nelle immagini 13.2 e 13.3 le linee non sono ancora ben definite. Questo è il risultato dell'*Edge Detector*.

Applicando l'algoritmo di semplificazione, sviluppato per questa tesi, e descritto nella Parte II, il risultato cambia notevolmente.

In Figura 13.4 e 13.5 possiamo vedere come vengono trasformati i risultati prima descritti. Entrambe le combinazioni di parametri portano a ottenere 7 linee e 5 nodi, cioè il risultato atteso.

13.3.3 Errore percentuale. I risultati mostrati in Figura 13.5 sono stati analizzati statisticamente.

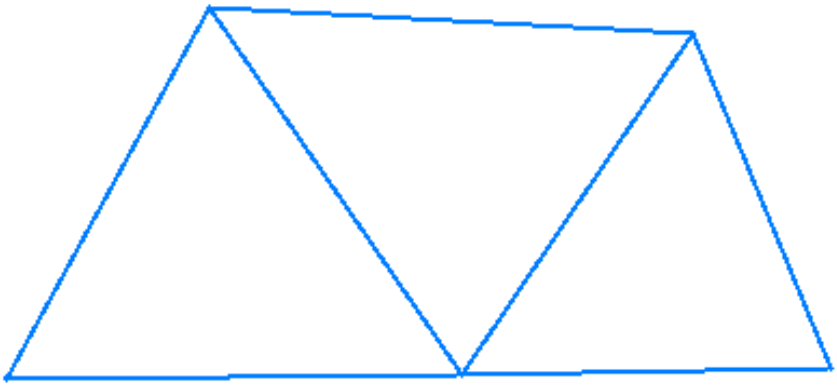


Figura 13.4: Risultato dell'algoritmo di semplificazione per i parametri della combinazione A.

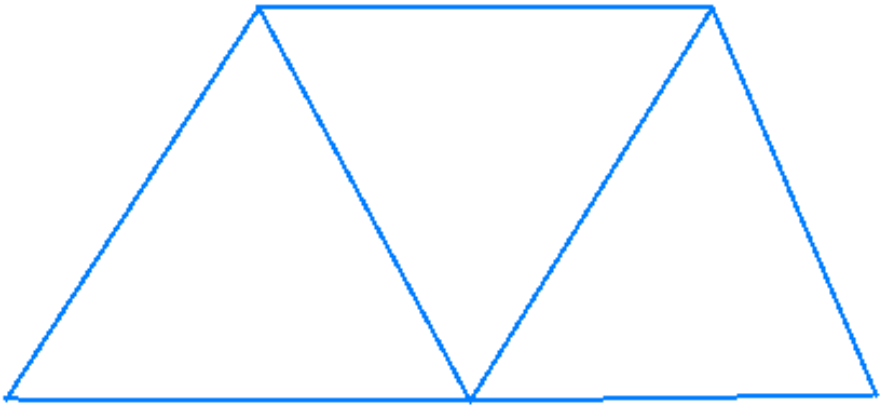


Figura 13.5: Risultato dell'algoritmo di semplificazione per i parametri della combinazione B.

Nella realtà, la struttura è costituita da tre triangoli equilateri, per cui il programma dovrebbe, in linea teorica, ottenere tre triangoli equilateri.

Misurando la lunghezza dei lati ricavati dal programma, se n'è calcolato il valore medio e la deviazione standard pari, rispettivamente, a 206.2 e 10.029 `pixel`.

Ogni stecchino, nella realtà, misura 6.5 centimetri. Ritenendo che questo valore corrisponda alla lunghezza media ricaviamo il fattore di conversione `pixel` in centimetri.

Infine, la deviazione standard espressa in `pixel` è stata convertita in millimetri, risultando pari a 3.16 millimetri.

In conclusione, quindi, un errore di 3.16 millimetri su 6.5 centimetri rappresenta un errore potenziale del 4.90%.

13.3.4 Esportazione DXF. La geometria è stata esportata in formato DXF. Il *file* così ottenuto, è stato aperto e visualizzato in *AutoCAD LT 2012*.

In Figura 13.6 è possibile vedere la schermata del programma AutoCAD mentre modifica la geometria appena ricavata dal modulo *Geometry From Image*.

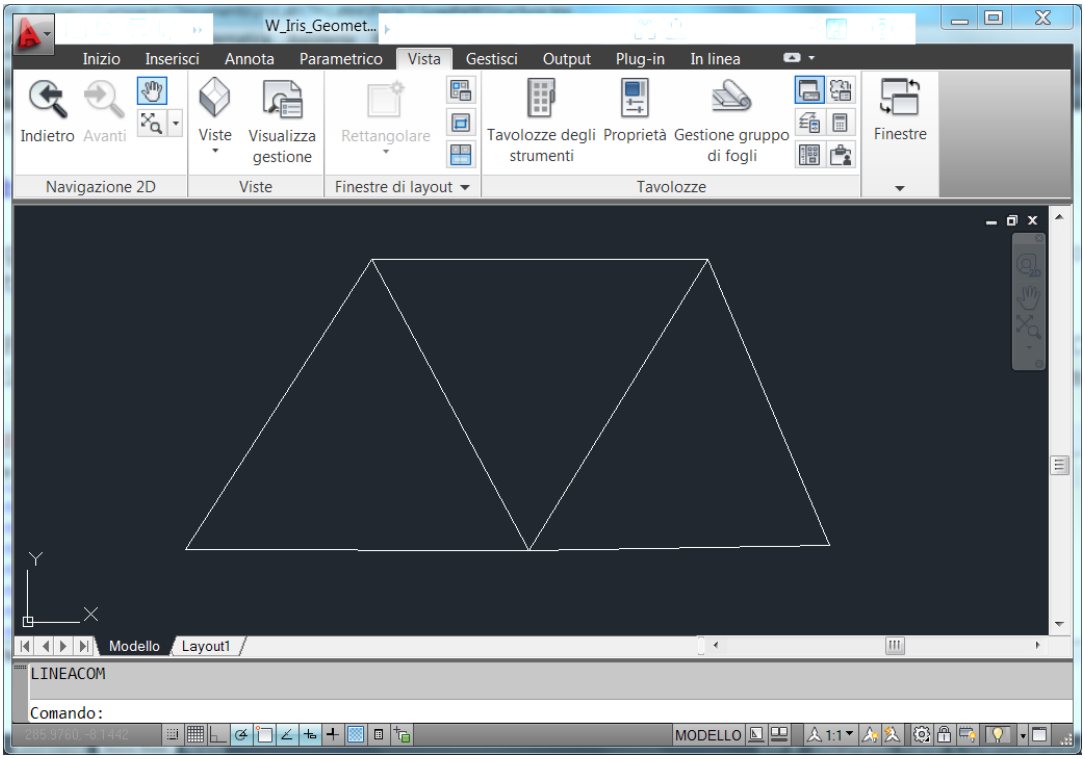


Figura 13.6: *Software* AutoCAD mentre modifica la geometria esportata dal modulo *Geometry From Image*.

14 Conclusioni

Il lavoro di tesi ha riguardato la realizzazione di un *software* basato sulle tecniche della *Digital Image Processing* per applicazioni strutturali.

Il *software* è stato diviso in tre moduli interconnessi tra di loro. Il primo si occupa della gestione video, ed in particolare dell'analisi del moto di punti ben definiti utilizzando la tecnica dell'*Object Detector*. Il secondo modulo, denominato *Deformation From Image*, ha il compito di determinare il campo di spostamenti e deformazioni in base a due fotografie di un provino, una dello stato indeformato ed una dello stato deformato. Infine, l'ultimo modulo ha applicato la tecnologia sperimentale dell'*Edge Detector* per il riconoscimento delle linee basilari delle strutture reticolari.

Alla realizzazione del software in linguaggio Python, è seguita una fase di *testing*. Per ogni modulo sono stati ideati e realizzati uno o più esperimenti, seguendo, talvolta, le indicazioni trovate in bibliografia.

Per il primo modulo, l'*Object Detector* è stato utilizzato per lo studio completo del moto di un pendolo. L'analisi, automatica, ha determinato tra l'altro la frequenza di oscillazione e il coefficiente di smorzamento.

Per il secondo modulo, le prove effettuate sono state tre. La prima, virtuale, ha permesso di verificare che il campo di deformazione è ottenuto correttamente. Le altre due prove, eseguite con criteri in bibliografia, sono servite per validare il metodo.

Per il terzo modulo, è stata effettuata una prova con una struttura reticolare. Mediante un'attenta regolazione dei parametri, è stata ricavata la geometria con sufficiente accuratezza da permetterne l'uso.

Il risultato delle prove ha messo in luce che il *software* sviluppato durante questa tesi è pronto per essere sottoposto a prove più realistiche. La non invasività delle tecniche ha permesso di stimare spostamenti e deformazioni da immagini o video acquisiti con

un apparato *hardware* economico e alla portata di ogni potenziale utilizzatore. Inoltre, parametri quali frequenze di oscillazione e coefficienti di smorzamento sono stati stimati con ottima precisione.

Tutto questo, con un'attenzione particolare all'usabilità: il *software*, lungi dall'essere complicato, è stato realizzato con una grafica moderna e, per quanto possibile, semplice.

15 Sviluppi futuri

Gli sviluppi futuri del progetto riguarderanno soprattutto l'utilizzo del *software* per casi di studio realistici.

Alcuni di queste possibili applicazioni saranno affrontate in un prossimo futuro:

- Stima di spostamenti, deformazioni e sollecitazioni in un provino metallico (con il modulo *Deformation From Image*);
- Stima delle deformazioni in una struttura reale, come ad esempio un'ala o una fusoliera sottoposte a prove statiche (con il modulo *Deformation From Image*);
- Misura delle frequenze di oscillazioni e modi di vibrare per una trave (con la tecnica dell'*Object Detector*);
- Misura di distanze e velocità in campo aeronautico, quali:
  - Distanza e velocità di decollo o atterraggio;
  - Quota di sorvolo;
  - Analisi di traiettoria;

  (con la tecnica dell'*Object Detector*).

Alcune di questi obbiettivi si dimostreranno di difficile approccio, altre impossibili, altre facili. Tuttavia il linguaggio di programmazione giovane, la modularità con cui è stato progettato il *software*, permetteranno, se necessario, di implementare con semplicità nuovi moduli e nuove caratteristiche.

In tal modo, lo sviluppo del programma potrà seguire l'evoluzione della *Computer Vision* e della *Digital Image Processing*, portando all'analisi strutturale i benefici di queste nuove tecniche informatiche.

Ringraziamenti

Durante la stesura della tesi ho pensato varie volte a cosa scrivere in questa pagina.

Questo è il lavoro conclusivo di cinque anni di studi universitari e, più in generale, di una meravigliosa esperienza. Ogni epilogo, si sà, porta con sé speranze e nostalgia pensando alle persone che ho incontrato, a quelle che ho conosciuto, a chi è diventato mio amico.

L'elenco dei ringraziamenti sarebbe, senza dubbio, lungo. Il mio ringraziamento particolare va, però, ai miei genitori che mi hanno dato l'opportunità di vivere questi anni univesitari con tranquillità e potendo sempre contare sul loro supporto.

Ringrazio il Prof. Marulo per avermi consigliato e sostenuto nell'affrontare questo argomento così interdisciplinare ed innovativo, stimolando la mia curiosità ed intraprendenza.

Non posso non nominare i miei amici, nonché colleghi universitari, Eva e Gabriele che sono stati al mio fianco in questi anni e con cui ho condiviso molte esperienze di vita, momenti belli e momenti difficili del percorso universitario.

Infine, per ultimo ma non meno importante, un abbraccio affettuoso va a Gilda che mi ha sopportato, aiutato e consigliato fin dalla scelta della facoltà, dai primi giorni ad oggi e, spero, continuerà a farlo in futuro.

Bibliografia

[1] P. Hung - A. S. Voloshin, In-plane Strain Measurement by Digital Image Correlation, 2003

[2] C. P. Papageorgiu - M. Oren - T. Poggio, A General Framework for Object Detection, 1998

[3] P. Viola - M. Jones, Robust Real-time Object Detection, 2001

[4] J. Canny, A Computational Approach to Edge Detection

[5] Natoshi Seo, Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features

[6] OpenCV Documentation, Cascade Classifier Training

[7] T. H. G. Megson, Aircraft Structures for Engineering

Elenco delle figure

Elenco delle tabelle

Indice