Webcam Based Document Annotation

Implementazione di un'applicazione desktop per l'annotazione real-time basata su webcam

Marco Righini
righini.marco88@gmail.com

&

Alessandro Galligari
galligari.alessandro@gmail.com

Versione 1. 0 – 7 settembre 2013

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

Il lavoro precedentemente svolto ha riguardato un certo e ben preciso ambito applicativo, ovvero la "Document Image Retrieval" basata sull'utilizzo di una fotocamera. La natura statica del sistema di acquisizione ha permesso una semplice separazione dei passi di elaborazione e una maggiore varietà di soluzioni implementative, quali un'applicazione desktop a cui fornire in input l'immagine scattata o un'applicazione su dispositivo mobile, con la quale effettuare il recupero delle pagine di un documento direttamente sul proprio smartphone (come realizzato dai nostri diretti predecessori).

Il lavoro affidatoci riguarda nuovamente la "Document Image Retrieval", ma richiede l'annotazione di parti di testo per la creazione di un nuovo documento contenente esclusivamente le parti annotate. La principale differenza rispetto ai precedenti lavori consiste nell'utilizzo di una webcam come strumento di collegamento fra carta e documenti digitali, un dispositivo di acquisizione real-time che necessita la stesura di una progettazione più complessa, introducendo problematiche tecniche e progettuali nuove rispetto ad un sistema di acquisizione statico come la fotocamera.

Lo scopo secondario, ma assolutamente rilevante, consiste quindi nella individuazione delle criticità di un sistema real-time, definendo un preciso flusso di esecuzione e cercando di gestire la maggior parte dei comportanti prevedibili e non di un utente medio.

La nostra attenzione si è concentrata sulla progettazione e sullo sviluppo di una infrastruttura in grado di offrire all'utente la possibilità di identificare la pagina stampata all'interno di un archivio di documenti, precedentemente preparato in una base dati. La fase successiva, che infonde un'ulteriore utilità pratica al tutto, ha visto l'individuazione di annotazioni sulla pagina fisica (in seguito a calibrazione del colore da segmentare) e il conseguente tracciamento sulla pagina digitale.

L'utente può in questo modo, ponendo le sue azioni sotto l'osservazione della webcam, effettuare operazioni di annotazione su pagine stampate e ritrovarsi in maniera automatica documenti digitali con le proprie selezioni.

Nel conseguire l'obiettivo prefissato, gli elementi critici rispetto alle principali soluzioni presenti in letteratura [Nak; NKI06; NKI07; TKI11] sono dettati dalla qualità dell'ottica dell'elemento d'acquisizione delle immagini e dalla necessità di considerare flussi video anzichè immagini statiche. I maggiori risultati riportati in letteratura si riferiscono infatti a fotocamere di qualità nettamente superiore rispetto a quelle delle webcam sul quale è stato impostato il lavoro ed inoltre mancano della componente temporale poichè non effettuano tipicamente l'identificazione e il tracciamento delle annotazioni.

L'utilizzo finale è tipico desktop, con l'uso di una webcam commerciale con modeste qualità di risoluzione ed accuratezza. Verranno presi in esame vari aspetti. Si elencano qui di seguito:

- **LLAH**: una breve analisi del metodo presentato in letteratura per individuare una pagina a partire da una sua immagine scattata;
- **Progettazione ed implementazione**: studio e realizzazione dei moduli per l'identificazione della pagina, calibrazione del colore da segmentare e tracciamento delle annotazioni;

- **Risultati sperimentali**: analisi dei risultati ottenuti per la calibrazione e per il recupero pagina;
- **Manuale utente ed esempio di esecuzione**: guida all'utilizzo dell'applicazione, corredata da un esempio pratico.

2 Locally Likely Arrangement Hashing

Il problema del recupero di pagine di un documento a partire da un'immagine è stato studiato e documentato in [Nak; NKI06; NKI07; TKI11]. Si farà principalmente riferimento alle soluzioni e al lavoro svolto in [NKI06], in particolare verrà descritto il metodo *Locally Likely Arrangement Hashing* (LLAH).

Le difficoltà maggiori sono rappresentate dalla distorsione prospettica, dall'illuminazione irregolare e dalla possibile mancanza di parti della pagina. Per quanto riguarda la prima problematica, LLAH calcola, per ogni insieme di f punti coplanari (opportunatamente definito), un invariante geometrico che ha la proprietà di rimanere inalterato in corrispondenza di trasformazioni geometriche.

Esistono vari tipi di invarianti geometrici, ognuno dei quali richiede un numero differente f di punti coplanari.

- **Cross-Ratio**: invariante rispetto a trasformazioni prospettiche, viene calcolato con l'uso di $f = 5$ punti coplanari A, B, C, D, E come segue

$$\frac{P(A,B,C)P(A,D,E)}{P(A,B,D)P(A,C,E)}$$

  dove $P(X, Y, Z)$ è l'area di un triangolo con vertici X, Y, Z;

- **Invariante affine**: rispetto al caso prospettico, l'invariante affine risulta essere più stringente, in quanto richiede che venga mantenuto il parallelismo delle linee. Per il calcolo vengono usati $f = 4$ punti coplanari A, B, C, D come segue

$$\frac{P(A,C,D)}{P(A,B,C)}$$

La trasformazione prospettica interessa un'area piuttosto limitata di un'immagine ritrante un pagina, quindi può essere approssimata con una trasformazione affine. Poiché il valore dell'invariante è continuo, si rende necessaria una discretizzazione così da poterlo utilizzare come indice nella tabella hash, come mostrato in Figura 1 [1].

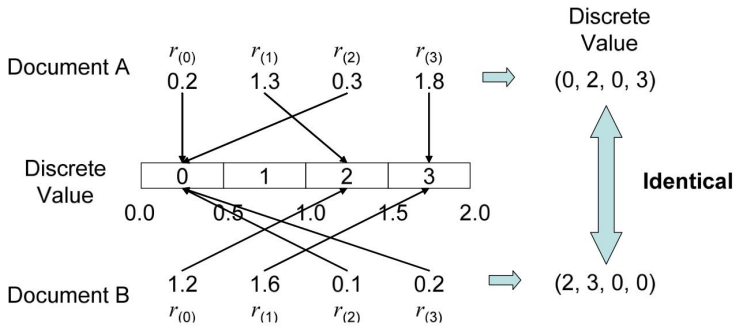


Figura 1: Discretizzazione degli invarianti affini

[1] Le immagini riportate in questa sezione sono riprese da [NKI06].

La discretizzazione avviene considerando k livelli differenti. Le soglie utili per la definizione dei livelli possono essere determinate in maniera tale da distribuire in modo uniforme i valori in essi, così da avere maggiore capacità di separazione dei valori delle feature.

2.1 Schema generale dell'algoritmo

In Figura 2 viene riportato lo schema generale dell'algoritmo.

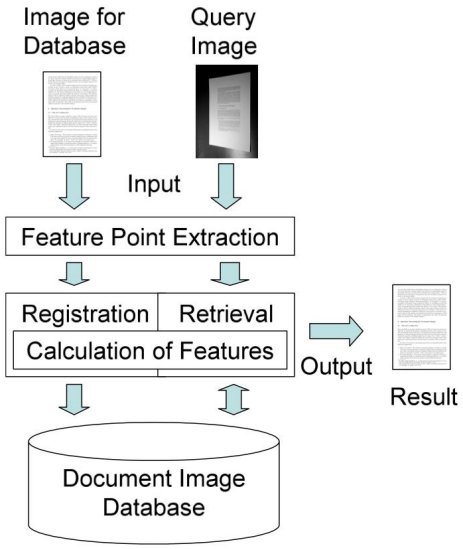


Figura 2: Schema generale dell'algoritmo

Si può notare come alcuni punti dello schema siano condivisi dalle operazioni di registrazione e recupero.

Lo step di estrazione elabora l'immagine in ingresso e rende un insieme di feature point. Questi vengono utilizzati per i calcolo degli invarianti sia in fase di registrazione che in fase di recupero: in fase di registrazione gli invarianti vengono usati per popolare la tabella hash, mentre in fase di recupero vengono utilizzati come indice della tabella e confrontati con il contenuto indicizzato, stabilendo con una votazione a maggioranza quale pagina registrata corrisponda a quella ritratta nella foto scattata.

Si riporta una descrizione dei vari step.

2.1.1 Estrazione dei feature point

L'estrazione dei feature point deve essere tale da fornire lo stesso risultato, ovvero lo stesso insieme di punti, in condizioni di distorsione prospettica, rumore o bassa risoluzione (nei limiti della praticabilità). Per questo motivo l'operazione di estrazione rende i centroidi delle parole, le quali sono considerate sufficientemente semplici da segmentare anche in condizioni critiche come quelle descritte. Generalmente il calcolo dei centroidi a partire da un'immagine (sia essa una pagina digitale o una foto) avviene nel seguente modo:

1. *thresholding adattivo*: si ottiene un'immagine binaria andando ad estrarre le informazioni di interesse anche in zone con irregolare luminosità (tipico delle foto);
2. *stima della dimensione dei caratteri*: ottenuta come la radice quadrata della moda delle aree delle componenti connesse dell'immagine binaria (ovvero i caratteri), serve a determinare un parametro utile per il passo successivo;
3. *blur gaussiano*: la stima della dimensione dei caratteri determina la forma del filtro gaussiano, con il quale idealmente fondere i caratteri tra loro;
4. *thresholding adattivo*: la fusione vera e propria avviene con una seconda operazione di soglia;
5. *calcolo dei centroidi delle parole*: le parole sono rappresentate dalle componenti connesse, dalle quali estrarre i centroidi, ovvero i feature point.

2.1.2 Calcolo delle feature

Per la realizzazione di una robusta operazione di recupero di un documento, il calcolo delle feature deve soddisfare due importanti requisiti:

- **feature stability**: lo stesso valore di feature deve essere ottenuto a partire dallo stesso feature point sottoposto ad una qualsiasi distorsione prospettica;
- **feature discrimination power**: differenti valori di feature dovrebbero essere ottenuti da differenti feature point.

Se non fosse possibile soddisfare tali condizioni si avrebbe, nel primo caso, l'impossibilità di recuperare la pagina corrispondente, mentre nel secondo caso si avrebbe un numero imprecisato di errate corrispondenze con documenti diversi. Per descrivere meglio il requisito di stabilità, si noti come la definizione più semplice del valore di feature di un punto p consista nel considerare gli f punti più vicini (con $f = 5$ per l'invariante cross-ratio e $f = 4$ per l'invariante affine). In corrispondenza di trasformazioni prospettiche, l'intorno dei punti di p potrebbe variare in maniera tale da invalidare irrimediabilmente il valore della feature.

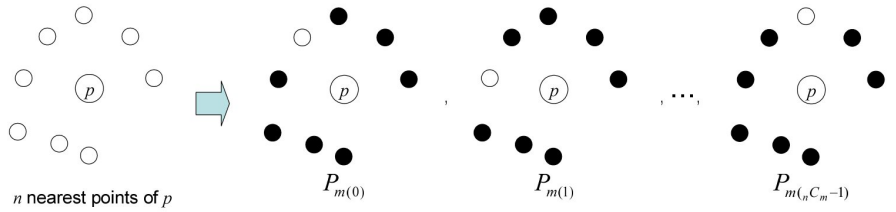


Figura 3: Combinazioni di $m = 7$ punti su $n = 8$ punti vicini

Per risolvere questo inconveniente si può ipotizzare che un numero m di punti su un totale di n punti nell'intorno di p siano comuni rispetto ad una qualsiasi distorsione prospettica. A partire da questa ipotesi, devono essere

esaminate, come mostrato in Figura 3, tutte le $C_m = \binom{m}{n}$ combinazioni $P_{m(0)}, \ldots, P_{m(C_m-1)}$ di m punti sugli n punti più vicini al punto p. Se l'ipotesi vale, almeno una combinazione di m punti è comune e la feature può considerarsi stabile.

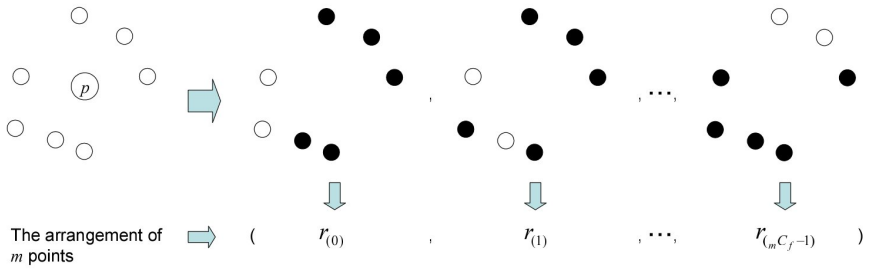


Figura 4: Sistemazione di $m = 7$ punti, descritta come una sequenza di invarianti calcolati da tutte le possibili combinazioni di $f = 5$ punti

Si consideri adesso il secondo requisito. Nuovamente, la via più semplice per definire la feature di m punti è impostare $m = f$ e calcolare l'invariante. Tale soluzione non rende la feature particolarmente discriminativa (può accadere che simili sistemazioni di punti possano essere ottenute da feature point diversi) e pertanto si provvede ad utilizzare feature point di un'area più vasta, incrementando $m(> f)$. All'aumentare di m diminuisce la probabilità che differenti feature point presentino la stessa sistemazione di punti. Come mostrato in Figura 4, una sistemazione di m punti viene descritta come una sequenza di invarianti discretizzati $\left(r_{(0)}, \ldots, r_{(C_f-1)}\right)$ calcolati a partire dalle $C_f = \binom{f}{m}$ combinazioni di f su m feature point.

Riepilogando, per ogni feature point p vengono considerati gli n punti più vicini, dai quali vengono generate tutte le C_m possibili combinazioni di m punti. Le feature vengono definite come sequenze di invarianti discretizzati considerando le C_f combinazioni di f punti su m.

2.1.3 Registrazione

Si riporta lo schema dell'algoritmo di registrazione in Algoritmo 1, nel quale si considerano

- **document ID**: identificatore del documento di appartenenza del punto;
- **point ID**: identificatore del punto;
- H_{index}: indice della tabella hash, calcolato come segue

$$H_{index} = \left(\sum_{i=0}^{C_f-1} r_{(i)} k^i \right) \mod H_{size}$$

dove $r_{(i)}$ è il valore discretizzato dell'invariante, k è il livello di quantizzazione dell'invariante e H_{size} è la dimensione della tabella hash.

Algorithm 1 Algoritmo di registrazione

for all $p \in$ All feature points in a database image **do**
  $P_n \leftarrow$ The nearest n points of p in a clockwise order
  for all $P_m \in$ All combinations of m points from P_n **do**
    for all $P_f \in$ All combinations of f points from P_m **do**
      $r_{(i)} \leftarrow$ The invariant calculated with P_f
    end for
    $H_{index} \leftarrow$ The hash index given $\left(r_{(0)}, \ldots, r_{(C_f-1)}\right)$
    Register the item (document ID, point ID, $\left(r_{(0)}, \ldots, r_{(C_f-1)}\right)$) using H_{index}
  end for
end for

L'elemento (document ID, point ID, $\left(r_{(0)}, \ldots, r_{(C_f-1)}\right)$) viene registrato nella tabella hash come mostrato in Figura 5, con l'utilizzo di una lista per la gestione delle collisioni.

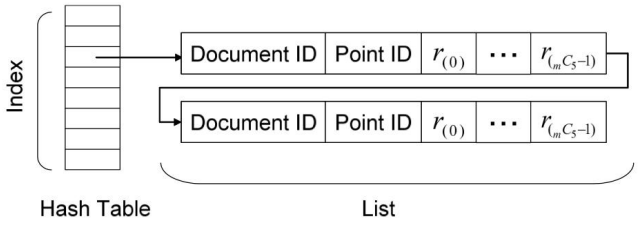


Figura 5: Struttura della tabella hash: le collisioni vengono gestite con una lista

2.1.4 Recupero

In Algoritmo 2 viene mostrato l'algoritmo di recupero di una pagina a partire da un'immagine di ricerca.

Per prima cosa si noti come l'indice della tabella hash venga calcolato in modo analogo rispetto alla registrazione. Con le ipotesi fatte precedentemente, indicizzando la tabella hash con tale valore si ottiene una lista di elementi nella quale potrebbe essere contenuto un elemento riguardante la sistemazione di punti corrispondente a quella analizzata P'_m. Per ogni elemento della lista viene incrementata una misura di votazione nel caso risultino soddisfatte le seguenti condizioni:

- **Condition 1**: tutti i valori di $\left(r_{(0)}, \ldots, r_{(C_f-1)}\right)$ nell'elemento corrispondono a quelli calcolati per la sistemazione P'_m;
- **Condition 2**: è il primo voto per document ID con punto p;
- **Condition 3**: è il primo voto per point ID di document ID;

Nel caso la prima condizione fosse la sola soddisfatta, si va incontro a due tipi di inconsistenze: un punto dell'immagine di ricerca corrisponde a due o più punti di un'immagine registrata e viceversa, ovvero un punto in un'immagine

Algorithm 2 Algoritmo di recupero

```
for all p ∈ All feature points in a query image do
  P_n ← The nearest n points of p in a clockwise order
  for all P_m ∈ All combinations of m points from P_n do
    for all P'_m ∈ Cyclic permutations of P_m do
      for all P_f ∈ All combinations of f points from P'_m do
        r_(i) ← The invariant calculated with P_f
      end for
      H_index ← The hash index given (r_(0), ..., r_(C_f − 1))
      Look up the hash table using H_index and obtain a list
      for all Item of the list do
        if Conditions 1 to 3 are satisfied then
          Vote for the document ID in the voting table
        end if
      end for
    end for
  end for
end for
Return the document image with the maximum votes
```

registrata corrisponde a due o più punti nell'immagine di ricerca. Le condizioni 2 e 3 rimuovono rispettivamente le due tipologie di inconsistenze.

3 Progettazione ed implementazione

Durante la fase di progettazione, l'attenzione è stata rivolta alla specifica di un'architettura modulare che risultasse estendibile e testabile in maniera non invasiva.

Sono stati progettati tre moduli:

- **registrazione**: fornisce la funzionalità di creazione della base dati a partire da documenti in formato PDF;
- **calibrazione**: consente l'individuazione di una o più soglie per la corretta segmentazione dei tratti colorati di annotazione;
- **annotazione**: permette di identificare la pagina inquadrata dalla webcam, rilevarne le annotazioni e generare successivamente i file di annotazioni.

Le sezioni seguenti mettono in luce le scelte progettuali per i singoli moduli ed alcuni particolari dettagli implementativi.

3.1 Registrazione

Il modulo di registrazione ha il compito di creare la tabella hash di feature, così come descritto nel Capitolo 2.

Nel continuo della trattazione si fa riferimento al diagramma di sequenza in Figura 6.

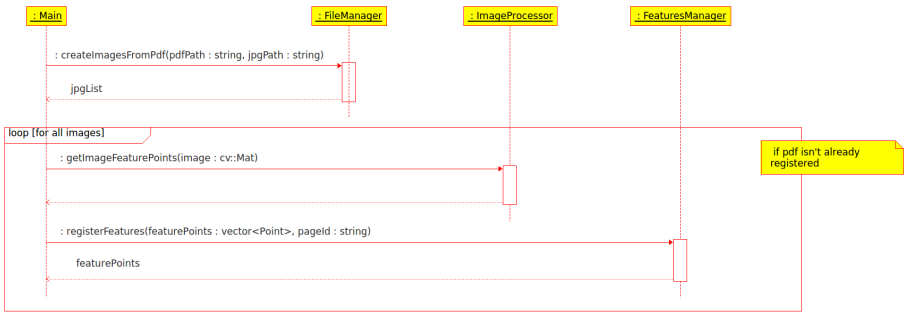


Figura 6: Diagramma di sequenza del modulo di registrazione

La chiamata a *createImagesFromPdf* effettua la conversione di formato del documento, da PDF ad immagini JPEG. Quest'ultime forniscono l'input per il core di registrazione e sono inoltre utilizzate nel modulo di annotazione quando è necessario tracciare le selezioni dell'utente sul file digitale originario.

I metodi *getImageFeaturePoints* e *registerFeatures* forniscono rispettivamente le funzionalità di estrazione dei punti caratterizzanti, i centroidi delle parole, e il calcolo ed inserimento nella tabella hash delle feature da essi derivanti (si faccia riferimento al Capitolo 2 per una descrizione dettagliata del metodo).

La tabella hash considerata è un'astrazione della struttura dati realmente utilizzata in questo contesto. Data la predisposizione di questo modulo alla

registrazione di un numero progressivamente crescente di documenti, si è resa necessaria la creazione di una base dati su DBMS SQLite [Sql]. Questa è caratterizzata da una tabella con i seguenti campi (oltre alla chiave principale):

- *documentId*: identificatore del documento di appartenenza del punto;
- *pointId*: identificatore del punto;
- *invariants*: vettore degli invarianti discretizzati $\left(r_{(0)}, \ldots, r_{(C_f-1)}\right)$ come descritto nel Capitolo 2;
- *hashIndex*: indice hash ottenuto dagli invarianti.

Sul campo *hashIndex* è stato definito un indice secondario per una maggiore efficienza nella fase di fetch dei record.

3.2 Calibrazione

Si descrive in questa sezione la realizzazione del modulo di calibrazione per il calcolo delle soglie utili per la segmentazione dei tratti caratterizzanti le annotazioni.

Questa esigenza nasce dall'impossibilità di determinare un'unica soluzione che porti ad una segmentazione corretta in ogni possibile configurazione di utilizzo e in ogni possibile condizione di luce. Inoltre si permette di disaccoppiare il calcolo delle soglie dall'utilizzo di una particolare webcam, soprattutto se la qualità di questa risulta influenzare in modo determinante l'acquisizione della scena reale.

Il modulo realizzato permette all'utente di calibrare le soglie seguendo un insieme ben definito di istruzioni. Come mostrato in Figura 7, il sistema presenta due finestre:

- la prima (a sinistra) rappresenta la scena reale a colori catturata dalla webcam;
- la seconda (a destra) rappresenta istante per istante la maschera binaria che l'algoritmo di calibrazione dovrà replicare attraverso la segmentazione della scena reale (maschera "truth").

In entrambe si può notare la presenza di alcuni rettangoli che hanno esclusivamente il compito di guidare l'utente al soddisfacimento delle seguenti condizioni prima dell'avvio della calibrazione:

- l'utente deve predisporre sotto la webcam un foglio bianco senza alcun segno distinguibile (testo o altro): questa condizione è resa necessaria dalle modalità di definizione della maschera "truth", ottenuta con thresholding adattivo sulla versione in scala di grigi della scena reale, limitata alla prima metà del rettangolo verde;
- il foglio deve essere posizionato così che la prima metà del rettangolo verde contenga esclusivamente il bianco della pagina: in questo modo nessuna parte dello sfondo sarà presente in tale zona;

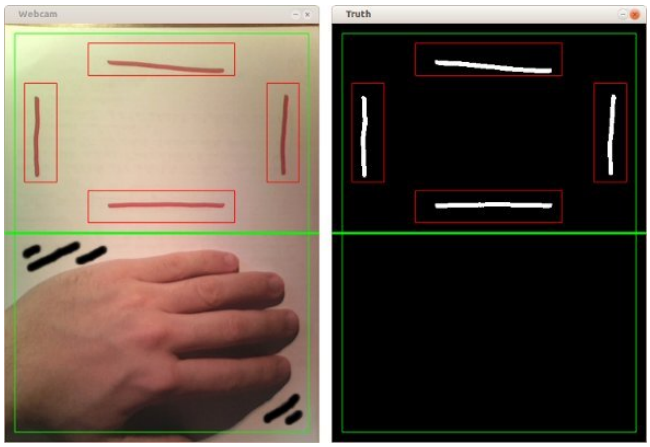


Figura 7: Modulo di calibrazione: la maschera "truth" estrae esclusivamente i tratti di interesse

- l'utente deve disegnare un tratto all'interno di ogni rettangolo di colore rosso: questi tratti rappresentano l'informazione da replicare in fase di ricerca delle soglie (la disposizione dei rettangoli è studiata in maniera tale da rendere la calibrazione indipendente dalla specifica posizione di un tratto);
- l'utente deve posizionare una mano all'interno della seconda metà del rettangolo verde: essendo parte interagente nell'attività di annotazione, la mano deve essere considerata nella fase di ricerca delle soglie ed esclusa dalla maschera "truth", così da segmentare solo i tratti di interesse (come è possibile constatare in Figura 7);
- il testo presente in una pagina di retrieval non deve essere segmentato, pertanto l'utente deve disegnare dei tratti di colore nero nella stessa area in cui è presente la mano (si garantisce maggiore robustezza);
- l'utente può scegliere di rendere visibile parte dello sfondo nella cornice esterna al rettangolo verde: in questo modo la segmentazione potrà risultare più robusta in presenza di considerevoli parti dello sfondo durante l'attività di annotazione.

La calibrazione è stata realizzata considerando i seguenti spazi colore:

- **RGB**: un modello di colore additivo in cui rosso, verde e blu si sommano in vari modi per la riproduzione di una vasta gamma di colori;
- **HSV**: una rappresentazione in coordinate cilindriche di punti di un modello di colore RGB;
- **Lab**: uno spazio color-opponent, con dimensione L per luminosità e a e b per le dimensioni color-opponent, basato sulle coordinate dello spazio colore non lineare CIE XYZ;
- **yCbCr**: una famiglia di spazi colore usata nei sistemi video a componenti e di fotografia digitale.

Nel contesto dell'ottimizzazione sono stati considerati i seguenti canali: i canali R, G e B dello spazio colore RGB, i canali H e S dello spazio colore HSV, i canali a e b dello spazio colore Lab e infine i canali Cb e Cr dello spazio colore yCbCr. In questa lista non sono stati considerati i canali rappresentanti la componente di luminosità perché privi di informazione cromatica.

Ogni canale utilizzato viene preprocessato con l'applicazione dell'operatore morfologico *BOTTOM HAT* per l'estrazione delle componenti di interesse e per la rimozione di illuminazione irregolare. Il risultato viene poi adeguato con l'applicazione di un filtro di smoothing mediano per l'eliminazione del rumore residuo.

L'algoritmo di calibrazione è stato suddiviso in due parti principali: ottimizzazione ristretta ad ogni canale e successiva combinazione delle soluzioni ottenute nel tentativo di caratterizzare con maggiore efficacia la segmentazione dei tratti di interesse. In entrambi i casi, la ricerca è guidata da una funzione obiettivo che verrà descritta nel seguente paragrafo.

3.2.1 Funzione obiettivo e problema di ottimizzazione

Prima di descrivere la funzione obiettivo, si introduce la seguente notazione:

- $\mathbb{T}$: maschera "truth" di riferimento;
- $\mathbb{I}$: canale preprocessato da sottoporre all'operazione di soglia;
- $\Omega = [0, 255] \cap \mathbb{N}$: insieme di numeri naturali da 0 a 255;
- $s = (a, b) \in \Omega^2$: estremo sinistro e destro dell'intervallo di soglia, la coppia rappresenta un punto nello spazio di ricerca.
- $Th(\mathbb{I}, s)$: rappresenta l'operazione di soglia sull'immagine $\mathbb{I}$ con $s = (a, b)$, ottenuta andando a prelevare tutti i pixel di $\mathbb{I}$ con valore nell'intervallo $[a, b]$;

Per la valutazione del punto $s = (a, b)$ vengono messe a confronto le maschere $\mathbb{T}$ e $Th(\mathbb{I}, s)$ e viene definita la seguente coppia di metriche sulla base della Tabella 1.

- $Precision = \frac{TP}{TP+FP}$;
- $Recall = \frac{TP}{TP+FN}$;

	Positivo in $\mathbb{T}$	Negativo in $\mathbb{T}$
Positivo in $Th(\mathbb{I}, s)$	True Positive (TP)	False Positive (FP)
Negativo in $Th(\mathbb{I}, s)$	False Negative (FN)	True Negative (TN)

Tabella 1: Definitioni di TP, FP, TN, FN per la classificazione di un pixel

Indicando con $\Gamma(\mathbb{T}, Th(\mathbb{I}, s))$ il confronto tra le due maschere e con $(p, r) \in [0, 1] \times [0, 1]$ la coppia di indici *precision* e *recall* che ne costituiscono il risultato, si definisce una funzione utile per la sintesi della coppia (p, r) in un unico valore,

nota come *F-measure*, di cui si riporta la forma più generale rispetto a due indici i_1 e i_2:

$$\mathcal{F} : [0,1] \times [0,1] \to [0,1]$$
$$\mathcal{F}(i_1, i_2) = (1+\beta^2)\frac{i_1 \cdot i_2}{\beta^2 \cdot i_1 + i_2}$$

dove $\beta \in \mathbb{R}$ permette una ridistribuzione del peso di i_1 e i_2 nel seguente modo:

- $\beta = 1$, pari peso per i_1 e i_2;
- $\beta = 0,5$, i_1 ha peso doppio rispetto a i_2;
- $\beta = 2$, i_2 ha peso doppio rispetto a i_1;

Tale funzione guiderà il problema di ottimizzazione che verrà in seguito formalizzato (con $\beta = 1$). Gode delle seguenti proprietà:

- $\lim_{(i_1,i_2)\to(0,0)} \mathcal{F}(i_1, i_2) = 0$;
- $\lim_{(i_1,i_2)\to(1,1)} \mathcal{F}(i_1, i_2) = 1$;
- non decrescente in $[0,1] \times [0,1]$, quindi le derivate parziali $\frac{\partial \mathcal{F}}{\partial i_1}$ e $\frac{\partial \mathcal{F}}{\partial i_2}$ risultano non negative in $[0,1]$;
- a sviluppo concavo in $[0,1] \times [0,1]$, quindi la matrice Hessiana risulta definita negativa;
- $\lim_{(i_1,i_2)\to(0,1)} \mathcal{F}(i_1, i_2) = \lim_{(i_1,i_2)\to(1,0)} \mathcal{F}(i_1, i_2) = 0$, così da non premiare soluzioni con prevalente fattore di i_1 o i_2.

Si definisce infine il problema di ottimizzazione

$$\begin{aligned} \max\ & \mathcal{F}(p, r) \\ & (p, r) = \Gamma\left(\mathbb{T}, Th(\mathbb{I}, s)\right) \\ & s = (a, b) \in \Omega^2 \\ & a < b \end{aligned}$$

Lo spazio di ricerca è Ω^2 e il vincolo $a < b$ è necessario per la definizione di un intervallo $[a, b]$.

3.2.2 Ottimizzazione di un singolo canale

Analizzando il problema di ottimizzazione appena formulato è possibile constatare come lo spazio di ricerca sia di dimensioni sufficientemente modeste: a partire da Ω^2, lo spazio viene dimezzato con la condizione $a < b$ per un totale di circa 32000 soluzioni ammissibili.

Pertanto sono stati sviluppati due algoritmi differenti: il primo effettua una ricerca esaustiva (*Brute Force Search*) su tutto lo spazio ammissibile mentre il secondo raffina una soluzione iniziale ottenuta con una scelta *greedy* (*Greedy Search*).

Algorithm 3 Algoritmo Brute Force Search

Data: $\mathbb{T}, \mathbb{I}$
$s^* \leftarrow (0, 255)$
$\mathcal{F}^* \leftarrow 0$
for all $a \in [0, 255]$ **do**
  for all $b \in [a, 255]$ **do**
    $s = (a, b)$
    $(p, r) = \Gamma(\mathbb{T}, Th(\mathbb{I}, s))$
    if $\mathcal{F}(p, r) > \mathcal{F}^*$ **then**
      $s^* \leftarrow s$
      $\mathcal{F}^* \leftarrow \mathcal{F}(p, r)$
    end if
  end for
end for

3.2.2.1 Brute Force Search

Si riporta lo schema nell'Algoritmo 3.

Trattandosi di una ricerca esaustiva, l'algoritmo determina l'ottimo globale del problema di ottimizzazione definito nel paragrafo precedente, ovvero la coppia di numeri naturali che permettono la segmentazione migliore rispetto ai criteri descritti (*precision* e *recall*).

Sebbene lo spazio di ricerca sia limitato, la valutazione di tutti i punti richiede un tempo di esecuzione eccessivo per l'attività di calibrazione. Pertanto è stato necessario studiare un'alternativa.

3.2.2.2 Greedy Search

L'algoritmo *Greedy Search* (Algoritmo 4) si basa su una semplice scelta "golosa", ovvero il punto iniziale della ricerca viene definito nel seguente modo:

- si collezionano nell'insieme $\mathcal{P}$ i pixel dell'immagine $\mathbb{I}$ corrispondenti ai tratti di interesse, estraendoli con l'ausilio della maschera $\mathbb{T}$;
- denotato con $gray(p)$ il valore grayscale del pixel $p \in \mathcal{P}$, si definisce il punto $s^* = (a^*, b^*)$ nel seguente modo

$$a^* = \min_{p \in \mathcal{P}} gray(p)$$
$$b^* = \max_{p \in \mathcal{P}} gray(p)$$

La scelta *greedy* permette di definire un punto tale da estrarre perfettamente i tratti di interesse, ma non garantisce l'esclusione di pixel che risultano essere di background nella maschera $\mathbb{T}$: rispetto ai criteri definiti, il punto iniziale s^* avrà *recall* massima.

Il passo successivo è l'applicazione di un metodo di ottimizzazione per il raffinamento della soluzione. Data l'assenza di informazioni sul gradiente della funzione obiettivo, è stato scelto un metodo di ricerca locale basato su una variante del metodo delle direzioni coordinate. Essendo Ω^2 uno spazio discreto e a due dimensioni, si hanno le seguenti caratteristiche:

Algorithm 4 Algoritmo Greedy Search

Data: $\mathbb{T}$, $\mathbb{I}$
$s^* \leftarrow ClosestInterval(\mathbb{T}, \mathbb{I})$
while stop criterion not satisfied **do**
  $s \leftarrow CoordinateDirectionsMethod(\mathbb{T}, \mathbb{I}, s^*)$
  if $isAcceptable(s, s^*)$ **then**
    $s^* = s$
  end if
end while

- quattro direzioni, due per ogni dimensione, ovvero le direzioni di incremento e decremento degli estremi dell'intervallo a e b;
- passo α lungo una direzione in $\mathbb{N}$.

Si descrivono nei seguenti punti le peculiarità dell'algoritmo e le principali operazioni effettuate dalla funzione $CoordinateDirectionsMethod(\mathbb{T}, \mathbb{I}, s^*)$:

- il criterio di arresto consiste nella verifica della disponibilità di direzioni di ricerca: qualora nessuna di queste fosse percorribile, l'algoritmo termina;
- ad ogni iterazione vengono valutate le direzioni di ricerca disponibili. Se una direzione porta ad un peggioramento della funzione obiettivo viene scartata;
- ad ogni iterazione la soluzione migliore s tra quelle ottenute viene confrontata con la soluzione attuale s^* e viene accettata se si verifica una delle due seguenti condizioni
  - $\mathcal{F}(s) > \mathcal{F}(s^*)$: la ricerca ha portato ad un miglioramento, in questo caso vengono riabilitate tutte le direzioni;
  - $\mathcal{F}(s) = \mathcal{F}(s^*) \wedge b - a > b^* - a^*$: la ricerca ha prodotto una soluzione di pari costo, ma ha permesso di definire un intervallo più ampio, così da permettere una maggiore capacità di generalizzazione (è possibile definire un numero massimo di passi di ricerca consecutivi a pari costo).

L'algoritmo di ricerca risulta essere efficiente ed efficace, capace di ripetere i risultati di *Brute Force Search* in tempi decisamente minori, rendendo l'attività di calibrazione adeguata rispetto a requisiti temporali più o meno stringenti.

3.2.3 Combinazione delle soluzioni su singolo canale

Una volta eseguite le varie ottimizzazioni sui singoli canali, è stato previsto un meccanismo di valutazione di tutte le possibili combinazioni tra k soluzioni, $k \in \{1, \ldots, k_{MAX}\}$, così da caratterizzare ulteriormente la segmentazione.

A seconda del colore scelto e delle condizioni di luce, è emerso che in alcune circostanze la combinazione di due o più soluzioni ha portato ad un notevole beneficio in termini di funzione obiettivo $\mathcal{F}$.

Per evitare una caratterizzazione troppo specifica, il numero massimo di soluzioni che possono comporre una combinazione è stato limitato ad un valore massimo k_{MAX}.

3.3 Annotazione

Il modulo di annotazione presenta, come obiettivo finale, la generazione di un file PDF con le annotazioni effettuate sulle pagine del documento stampato. Per perseguire questo scopo si è deciso di implementare una macchina a stati, i cui stati sono descritti in dettaglio nelle sezioni successive.

3.3.1 Macchina a stati

In sede di introduzione è stata presentata la problematica della gestione di una modalità di acquisizione completamente differente rispetto a quella utilizzata in precedenza. La natura statica di una fotocamera definisce implicitamente la sequenzialità delle operazioni di recupero di una pagina digitale: l'utente inquadra il documento cartaceo, scatta una foto, fornisce (in modo automatico o non) l'input ad un programma di elaborazione, il quale rende il risultato del processo di recupero ed eventualmente individua annotazioni di vario genere (coadiuvato da una buona risoluzione del sistema di acquisizione). Ogni deviazione da questo flusso sequenziale comporta un evidente malfunzionamento, facilmente individuabile dal sistema (ad esempio l'assenza di un documento).

Nel caso dell'utilizzo di un sistema di acquisizione real-time come una webcam, il flusso di operazioni non è banale e ben definito come nel caso precedente. Senza un'opportuno studio preliminare, l'utente sarebbe chiamato ad un'interazione particolarmente gravosa, utile a notificare al sistema di elaborazione cosa stia accadendo sotto l'obiettivo della webcam.

Per questo motivo si è resa necessaria la definizione di una macchina a stati che delineasse i principali passi di esecuzione, dall'acquisizione di un frame al recupero di un documento fino alla "detection" delle annotazioni. Individuati gli stati maggiormente rappresentativi del flusso delle operazioni, è stata definita una serie di eventi che caratterizzano le transizioni da stato a stato: tali eventi sono stati pensati per rendere l'interazione dell'utente intuitiva e ridotta all'essenziale e per catturare e gestire, senza generare malfunzionamenti o condizioni di ambiguità, tutte le eventuali deviazioni dell'utente dal comportamento "ideale".

In Figura 8 viene presentata la macchina a stati ideata e realizzata per questo progetto, della quale verranno descritti in modo esaustivo gli stati e gli eventi che la caratterizzano.

3.3.2 PageDiscovering

Lo stato *PageDiscovering* è stato progettato per assolvere la funzione di riconoscimento della presenza di una pagina stampata. Tramite questo passo iniziale si è voluto assicurare che allo stato seguente di recupero della pagina sia fornita in input una effettiva immagine di pagina fisica, di modo da rendere l'intero processo maggiormente stabile.

Il diagramma in Figura 9 visualizza i principali blocchi di elaborazione, di seguito descritti in dettaglio.

Threshold adattivo : questa tecnica di preprocessing è utilizzata per segmentare oggetti con marcata differenza di intensità all'interno delle immagini, andando a rimuovere l'illuminazione non costante.

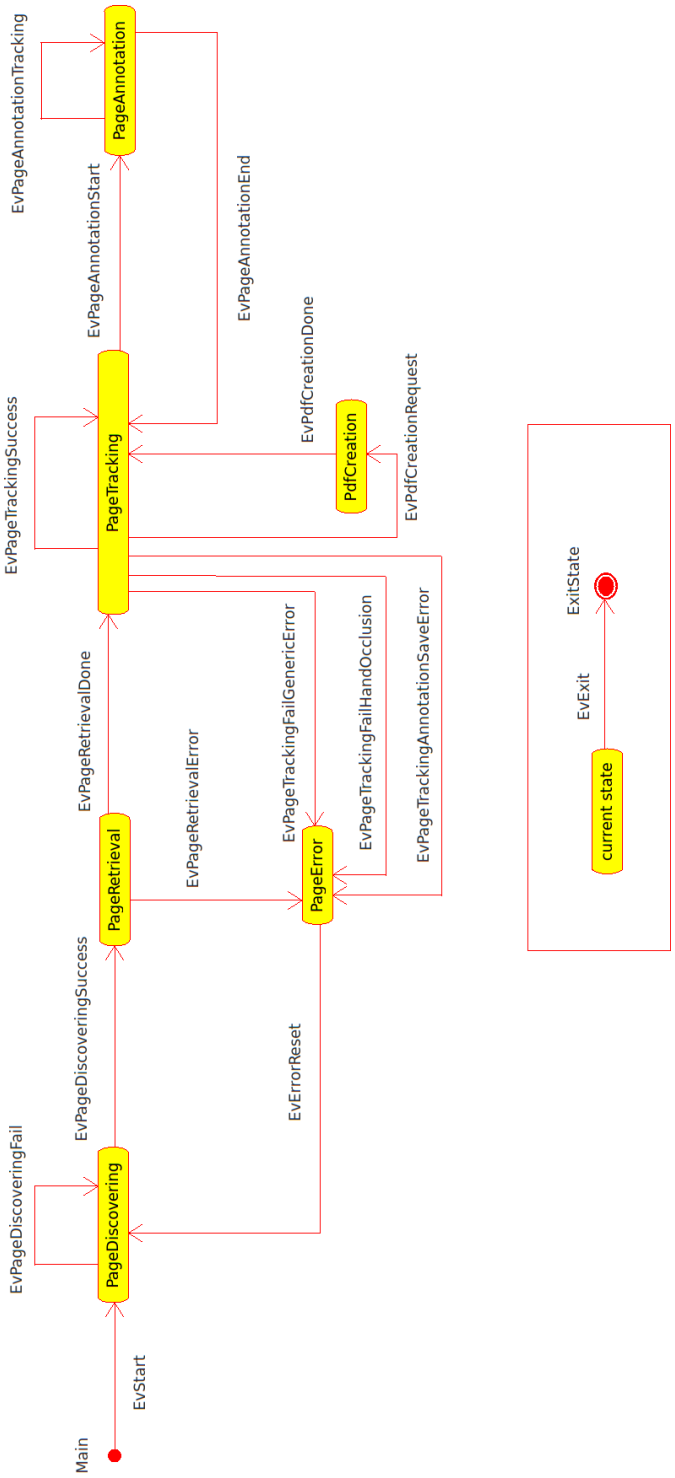


Figura 8: Macchina a stati

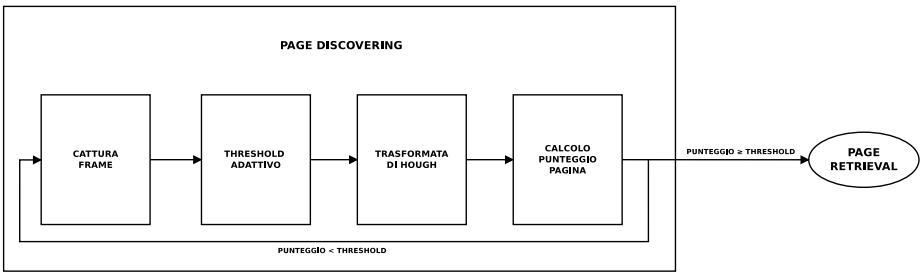


Figura 9: Diagramma a blocchi dello stato PageDiscovering

A differenza dei metodi di thresholding di tipo globale, che effettuano la binarizzazione considerando un unico valore soglia prefissato per l'intera immagine, quelli di tipo adattivo si fondano sulla computazione di un valore di threshold per ogni pixel.

In dettaglio, viene calcolata la media dei pixel che si trovano in un intorno, di dimensione prefissata, del pixel in esame e questa è considerata come soglia per il punto considerato.

Il thresholding adattivo è risultato particolarmente utile per segmentare i caratteri dal resto dell'immagine.

Trasformata di Hough : la trasformata di Hough è una tecnica che permette di riconoscere particolari configurazioni di punti presenti nell'immagine come segmenti, curve e forme prestabilite. È basata sulla "validazione delle ipotesi" per la quale, dopo aver definito la curva che si intende cercare nella scena, si calcolano, per ogni pixel dell'immagine, i parametri di tutte le possibili curve che potrebbero passare per quel punto e si incrementano le celle dello spazio n-dimensionale (con n numero dei parametri) che corrispondono alle varie curve. In questo modo si ottiene una funzione di accumulazione nello spazio dei parametri e infine saranno i massimi di questa funzione, i.e. i punti nello spazio dei parametri con il maggior numero di voti, a rappresentare le curve che hanno probabilità elevata di essere presenti nell'immagine.

Nel lavoro esposto questa tecnica è stata utilizzata per ricercare segmenti di linee, basandosi su una ipotesi fondamentale: le parole del testo formano segmenti di linee ben definiti e pressochè paralleli. Assumendo infatti che la pagina sia poggiata su un piano e la telecamera sia parallela a quest'ultimo, la pagina stampata non subisce rilevanti deformazioni prospettiche.

Effettuando un'azione di filtraggio sui risultati della trasformata (i.e. le differenti orientazioni) è possibile stimare la presenza o assenza di un foglio stampato nel cono visivo della webcam, così come descritto nel seguente paragrafo.

Calcolo punteggio pagina : nel caso generico di scena inquadrata arbitraria, il risultato della trasformata di Hough è un insieme disomogeneo di segmenti, caratterizzato da un ampia fascia di orientazioni differenti. Al fine di stimare la presenza di una pagina si è così sfruttata la differenza

fra la configurazione generale precedentemente introdotta e la configurazione con pagina presente: in quest'ultima le orientazioni si discostano in maniera molto meno significativa ed inoltre, sfruttando l'ipotesi di assenza di deformazioni prospettiche, è possibile considerarle identiche a meno di pochi gradi (nel lavoro si è visto che in inquadrature standard, i coefficienti angolari delle linee di pagina si discostano al massimo di circa 3 gradi in difetto o eccesso).

Seguendo le considerazioni sopra esposte, sono state effettuate le seguenti operazioni:

- *filtraggio dei coefficienti angolari*: a partire dalla distribuzione dei coefficienti angolari ottenuti dalla trasformata di Hough, si prende l'elemento mediano come riferimento. Successivamente si computa la differenza fra ciascun angolo e quello di riferimento: se la variazione è inferiore ad una soglia angolare (i.e. 3 gradi in difetto o eccesso) questo segmento è considerato parallelo ed inserito nell'insieme dei segmenti positivi. Quest'ultimo insieme è passato all'operazione successiva di valutazione della funzione punteggio;
- *valutazione della funzione composta di punteggio*: la funzione di punteggio è il prodotto di due componenti. La prima è la percentuale di segmenti positivi rispetto al totale di segmenti trovati; questa nella sua individualità non è sufficiente a valutare la presenza di una pagina poichè si hanno valori alti sia nel caso in cui siano stati trovati molti o pochi segmenti. É risultato così necessario introdurre la seconda componente (vedi Figura 10) che effettua un'azione di pesatura rispetto alla precedente.

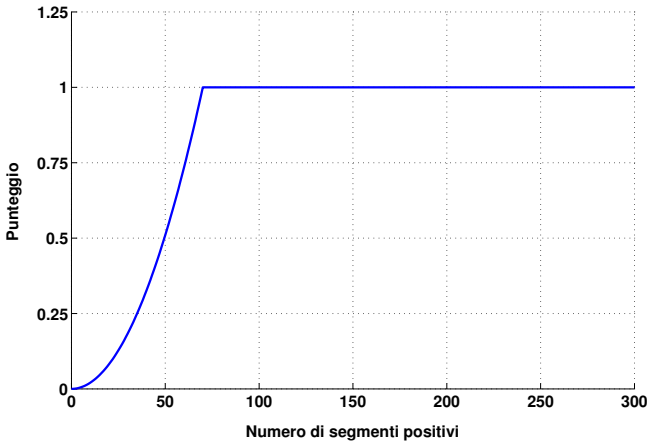


Figura 10: Seconda componente della funzione di punteggio

Si attribuisce peso unitario quando nell'immagine i segmenti positivi sono numericamente superiori ad un valore soglia preliminarmente analizzato (i.e 70). La funzione valutata è quadraticamente descrescente al decrescere del numero di segmenti positivi.

Se il prodotto delle componenti di punteggio è superiore ad un valore minimo (i.e. 0.8) si può affermare che nell'immagine è presente un foglio stampato o una sua rilevante porzione e si determina il passaggio allo stato *PageRetrieval* tramite l'evento *EvPageDiscoveringSuccess*.

3.3.3 PageRetrieval

Lo stato di recupero della pagina corrispondente a quella ritratta nel frame considerato è puramente uno stato di transizione. Come mostrato in Figura 11 non è previsto nessun self-loop in quanto lo scopo principale è l'esecuzione dell'operazione di "retrieval" della pagina. Di seguito vengono descritti i dettagli.

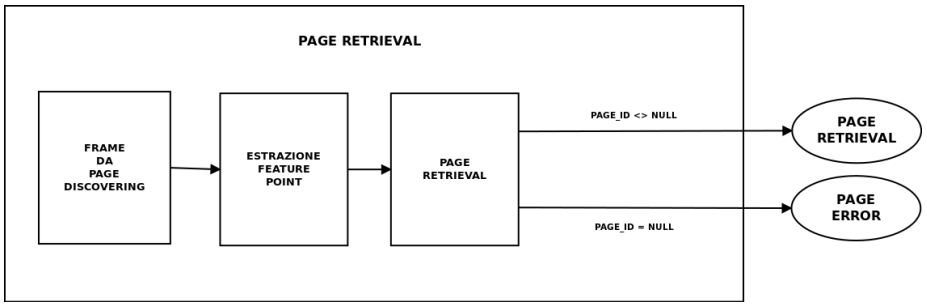


Figura 11: Diagramma a blocchi dello stato *PageRetrieval*

Frame da *PageDiscovering* : dallo stato *PageDiscovering* viene considerato il frame che ha determinato la transizione di stato che, con buona approssimazione, rappresenta una scena contenente una pagina.

Estrazione feature : il frame viene processato per l'estrazione dei feature point come descritto nel Capitolo 2.

Page Retrieval : dati i feature point viene eseguito l'Algoritmo 2 di recupero di Pagina 12, il quale rende un identificatore di pagina. Si distinguono due casi:

- *Pagina trovata*: l'algoritmo di recupero fornisce un identificativo di una pagina, in particolare il nome del file immagine; in tal caso viene generato l'evento *EvPageRetrievalDone* per la transizione allo stato *PageTracking*;
- *Pagina non trovata*: l'algoritmo di recupero non è stato in grado di ricostruire nessuna corrispondenza e pertanto viene generato l'evento *EvPageRetrievalError* per la transizione allo stato *PageError*. Le cause possono essere molteplici: documento non presente nel database, database vuoto, frame raffigurante una scena senza pagina o con una pagina disposta in modo critico (forte distorsione prospettica, indistinguibilità di caratteri o parole, lontananza della pagina o assenza rilevante di parti della stessa, motion blur, ecc.) oppure inadeguata impostazione dei parametri della webcam (ad esempio la messa a fuoco).

Sebbene le operazioni si riducano alla sola esecuzione dell'algoritmo di recupero, è stato scelto di definire uno stato apposito poiché tale operazione può richiedere un tempo sufficientemente apprezzabile da parte dell'utente. Il sistema di visualizzazione permetterà quindi di notificare all'utente le operazioni in corso per una agevole interpretazione delle stesse.

3.3.4 PageTracking

Lo stato *PageTracking* è il fulcro dell'intero processo di annotazione. In Figura 12 se ne riporta il diagramma a blocchi.

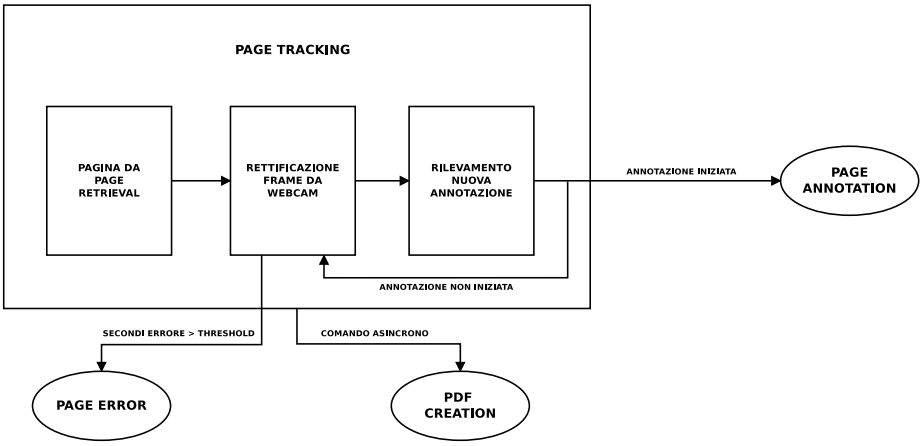


Figura 12: Diagramma a blocchi dello stato *PageTracking*

Di seguito sono descritti i principali costituenti, con gli eventi annessi.

Pagina da *PageRetrieval* : a partire dal path della pagina recuperata nello stato di *PageRetrieval*, è caricato in memoria il corrispondente file immagine che svolge la funzione di riferimento per la successiva rettificazione;

Rettificazione del frame da webcam : l'obiettivo preposto è quello di determinare la relazione fra un frame da webcam e l'immagine di riferimento; tramite la computazione dell'omografia planare che lega i due piani immagine è possibile effettuare un'operazione di rettificazione e determinare la corrispondenza univoca fra un pixel dell'immagine di riferimento e un pixel scelto del frame da webcam.

Si distinguono i seguenti passaggi:

- *Determinazione keypoint e descrittori SIFT* : la prima fase di elaborazione include il rilevamento di punti notevoli per le due singole immagini e la computazione dei corrispondenti descrittori[2].

  I keypoint SIFT sono definiti come massimi locali del risultato derivante dall'applicazione di un filtro DOG (*Difference of Gaussians*)

[2]Si noti che, rimanendo l'immagine di riferimento immutata durante l'intera esecuzione dello stato, non è necessario, per quest'ultima, effettuare il rilevamento dei keypoint e la loro descrizione ad ogni iterazione ciclica di *PageTracking* ma bensì una sola volta e mantenere i descrittori in memoria.

ad una serie di immagini multiscala, ottenute tramite smoothing e resampling dell'immagine originale. Punti candidati con basso contrasto locale o disposti su edge sono scartati; ai rimanenti sono assegnati quattro parametri (x, y, scala e orientazione) che definiscono il keypoint. L'utilizzo dello spazio multiscala assicura invarianza a traslazione, rotazione e scala che sono trasformazioni tipiche dell'ambiente di questo lavoro.

Successivamente, ad ogni keypoint è associato un descrittore SIFT, ottenuto come istogramma (a 128 dimensioni), pesato e interpolato, delle orientazioni dei gradienti in un blocco di pixel circostante la posizione del punto notevole considerato.

- *2-NearestNeighbour matching e filtraggio* : il processo di matching prevede, per ogni descrittore del frame da webcam, la ricerca dei due descrittori dell'immagine di riferimento più similari secondo una metrica prestabilita. Nel lavoro è stata utilizzata la norma L2 che in fase di esecuzione ha riportato i risultati più soddisfacenti.

  In dettaglio, si è scelto di ricercare i due descrittori più vicini invece di uno unico poichè ciò permette di utilizzare un accurato metodo di filtraggio dei match inesatti (come descritto in [Low03]): una corrispondenza è accettata se e solo se la distanza dal primo descrittore più vicino è significativamente più piccola di quella dal secondo descrittore più vicino, i.e. il rapporto fra la prima distanza e la seconda è inferiore ad una certa soglia (0.8). Un ulteriore filtraggio scarta i match per i quali la distanza dal primo elemento più vicino è maggiore di un threshold determinato empiricamente.

  Attraverso i criteri di selezione precedentemente esposti, si assicura che al calcolo dell'omografia siano fornite il maggior numero di match esatti.

- *Computazione omografia e rilevamento errori* : a partire dalle corrispondenze ottenute al passo precedente, si vuole stimare la trasformazione planare fra i due piani immagine, quello di riferimento e quello inquadrato dalla webcam.

  Sebbene gli algoritmi di stima dell'omografia risultino robusti a errori di misura sulle posizioni dei match, anche dopo l'operazione di filtraggio può accadere che siano presenti corrispondenze errate. Per ovviare a questo problema, è stato utilizzato il metodo di stima robusta RANSAC (*Random Sample Consensus*), un algoritmo di tipo iterativo che presenta come obiettivo generale la stima dei parametri di un modello matematico a partire da un insieme di dati contenente outlier[3].

  Durante il processo di stima sono state gestite due differenti situazioni di errore. La prima scaturisce quando il passo precedente di matching fornisce un insieme di corrispondenze filtrate numericamente insufficiente al processo di calcolo della trasformazione planare[4], a causa ad esempio di un'errata pagina recuperata nello stato *PageRetrieval*.

[3] Il termine si riferisce a dati in ingresso erroneamente considerati appartenti al modello da stimare.

[4] Un'omografia presenta otto gradi di libertà e può essere calcolata a partire da almeno quattro corrispondenze poichè ciascuna coppia fissa due gradi di libertà.

La seconda situazione si verifica invece quando il metodo RANSAC e i precedenti filtraggi dei match non riescono a sopperire alla presenza di match errati: controllando il determinante della matrice di omografia, se risulta prossimo a zero (matrice mal condizionata) possiamo affermare, con buona approssimazione, di essere in presenza di una errata stima di trasformazione. Alla generazione di una situazione di errore, un processo di *skin detection* permette di discernere la causa dell'anomalia fra errore generico e errore dovuto ad una eccessiva occlusione della scena da parte della mano: se lo stato di anomalia si potrae per un prefissato numero di secondi, sono generati rispettivamente gli eventi *EvPageTrackingFailGenericError* e *EvPageTrackingFailHandOcclusion* per la transizione allo stato *PageError*.

Rilevamento nuova annotazione : nel flusso esecutivo, il rilevamento di una nuova annotazione è eseguito dopo aver effettuato con successo il calcolo della matrice di rettificazione.

Questa fase prevede la ricerca delle componenti connesse di colore nell'immagine da webcam rettificata, secondo le soglie determinate in fase di calibrazione. Dopo averle individuate si procede al confronto con una eventuale maschera di annotazioni precedentemente salvata su filesystem per la pagina recuperata: se le componenti trovate intersecano la maschera preesistente siamo in presenta di annotazioni già trattate altrimenti l'annotazione è considerata come nuova e deve essere gestita.

La presenza di un flag di salvataggio della maschera di annotazione permette di effettuare due azioni differenti: se è settato a false viene generato l'evento *EvPageAnnotationStart* che porta alla transizione verso lo stato *PageAnnotation* mentre se è settato a true si procede all'aggiornamento della maschera di annotazioni (se in questa fase si ha un errore reiterato del tracking è generato l'evento *EvPageTrackingAnnotationSaveError* che porta nello stato di errore *PageError*).

In dettaglio, durante l'aggiornamento della maschera si ricercano eventuali corrispondenze fra annotazioni, analizzando lunghezza e posizione delle nuove componenti trovate: se due componenti risultano accoppiate definiscono una zona di annotazione rettangolare racchiusa fra esse mentre se una componente è singola l'annotazione corrispondente è una striscia che attraversa tutta l'immagine in larghezza. La maschera aggiornata è così composta dall'unione della nuova maschera e di quella preesistente: questa è utilizzata per aggiornare anche l'immagine che rappresenta il filtraggio della pagina originale rispetto alle selezioni.

In conclusione, si noti la presenza del comando asincrono per la creazione del PDF delle annotazioni, che genera l'evento *EvPdfCreationRequest*;

3.3.5 PageAnnotation

Dallo stato *PageTracking* si ha una transizione allo stato *PageAnnotation* in concomitanza con l'individuazione di un'annotazione diversa rispetto a quelle salvate sull'apposita maschera.

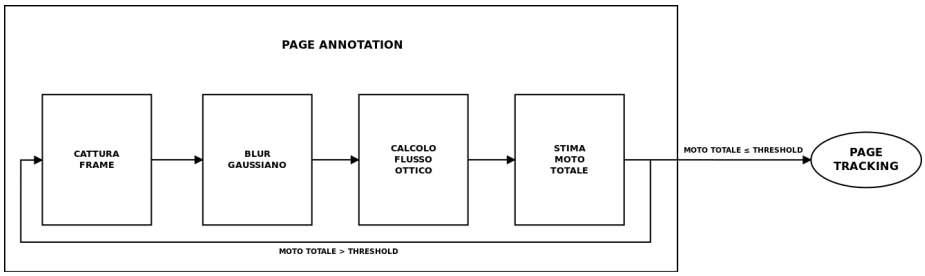


Figura 13: Diagramma a blocchi dello stato *PageAnnotation*

L'obiettivo di questo stato consiste nel concedere all'utente il tempo necessario a terminare l'annotazione. Il sistema sarà chiamato ad interpretare la fine dell'annotazione con l'assenza totale di moto nell'immagine, condizione che determinerà la transizione indietro allo stato *PageTracking* con l'evento *EvPageAnnotationEnd*. Fintanto che del moto viene individuato, il sistema permane nello stato *PageAnnotation* con l'evento *EvPageAnnotationTracking*.

L'idea alla base della progettazione di questa parte della macchina a stati consiste nell'individuare e separare i punti principali dell'operazione di annotazione, ovvero inizio, prosieguo e fine dell'annotazione. Infatti l'evento di "inizio" viene catturato nello stato *PageTracking* determinando il passaggio a *PageAnnotation*, il prosieguo viene consentito nello stato *PageAnnotation* e l'evento di "fine" annotazione viene catturato nello stato *PageAnnotation*, determinando nuovamente la transizione a *PageTracking* che, come descritto, provvederà al salvataggio e alla notifica dello stesso.

In Figura 13 viene mostrato il flusso di operazioni dello stato *PageAnnotation*. In seguito vengono descritti i dettagli.

Blur gaussiano : anche detto "smoothing gaussiano", è una tra le più utilizzate e note tecniche di preprocessing per la riduzione di rumore o di dettaglio. L'applicazione di tale filtro corrisponde, da un punto di vista matematico, alla convoluzione dell'immagine con una funzione gaussiana bivariata

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

Nel contesto dello stato *PageAnnotation*, il blur gaussiano è stato utilizzato per attenuare gli effetti del rumore introdotto dalla webcam, così da stabilizzare il comportamento del calcolo del flusso ottico. In questo modo si tende ad abbassare il numero di outliers in fase di matching.

Calcolo flusso ottico : il flusso ottico rappresenta il modello di moto apparente di oggetti in una scena visiva, causato dal movimento relativo tra un osservatore e la scena. In OpenCV esistono varie implementazioni per il calcolo del flusso ottico, fra le quali il metodo con piramidi di Lucas-Kanade, l'algoritmo di Gunnar Farneback e l'algoritmo SimpleFlow.

Per l'implementazione è stato scelto il metodo di Lucas-Kanade: si assume che il flusso sia costante in un intorno locale del pixel sotto esame e si risolvono le equazioni del flusso ottico per tutti i pixel in tale intorno, con l'utilizzo del criterio dei minimi quadrati.

In questo ambito non c'è alcun particolare dettaglio da seguire da un frame all'altro: il calcolo del flusso ottico è utile esclusivamente per l'individuazione di moto e pertanto, ad ogni iterazione, viene definito un reticolo di punti del frame precedente da individuare nel frame catturato.

Stima moto totale : in questa fase vengono filtrate le varie corrispondenze trovate dall'algoritmo di Lucas-Kanade, calcolando per ognuna di esse lo spostamento compiuto. Tale quantità viene considerata se maggiore di una soglia minima (per filtrar via gli effetti di eventuale rumore residuo) e minore di una soglia massima (utile per scartare spostamenti inverosimili, ovvero outliers). Gli spostamenti filtrati vengono sommati e il moto totale confrontato con una soglia per stabilire la presenza o meno di moto apprezzabile nell'immagine. Nel caso il moto totale sia minore di tale soglia, il sistema interpreta tale situazione come fine dell'annotazione e si ha la transizione allo stato *PageTracking* tramite l'evento *EvPageAnnotationEnd*, dopo un prefissato intervallo di tempo.

3.3.6 PdfCreation

Lo stato *PdfCreation* può essere raggiunto esclusivamente dallo stato *PageTracking* attraverso l'interazione dell'utente con un comando asincrono. In questo stato viene creato un documento PDF a partire dalle pagine del documento originale che risultano annotate. In Figura 14 viene mostrato il flusso di esecuzione.

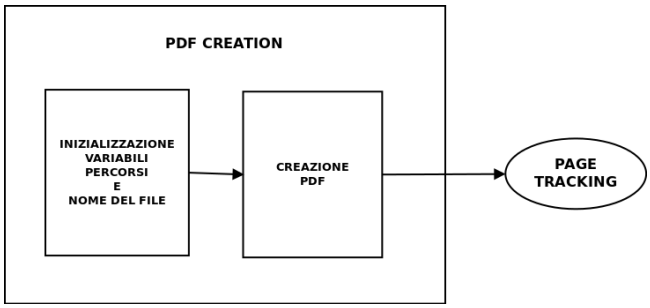


Figura 14: Diagramma a blocchi dello stato *PdfCreation*

Inizializzazione variabili percorsi e nome del file : si ha la definizione del percorso di salvataggio del documento e del nome dello stesso a partire dall'identificatore della pagina (PAGE_ID) recuperata nello stato *PageRetrieval* e inseguita nello stato *PageTracking*.

Creazione PDF : vengono considerati tutti i file immagine ottenuti in fase di salvataggio dell'annotazione con prefisso il nome del documento da creare (un'immagine presenterà del testo laddove questo risulti essere sovrapposto da una qualche annotazione che interessa tale pagina). Si noti come questo documento risulterà composto dalle sole pagine caratterizzate da annotazioni.

Per la conversione da un insieme di immagini ad un PDF è stato utilizzato il comando di sistema *convert*.

Al termine della creazione si ha il passaggio allo stato *PageTracking* tramite l'evento *EvPageCreationDone*.

3.3.7 PageError

Lo stato *PageError*, come è possibile constatare dalla Figura 8, rappresenta lo stato di destinazione della macchina qualora si presenti un errore durante il normale funzionamento del sistema.

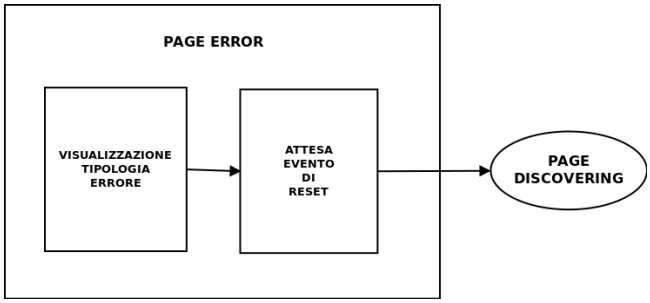


Figura 15: Diagramma a blocchi dello stato *PageError*

Tale stato ha il semplice compito di interporsi tra l'errore e il reset del sistema per la notifica all'utente dell'errore stesso. L'interazione dell'utente determina il reset del sistema con la transizione allo stato iniziale *PageDiscovering* data dall'evento *EvErrorReset*.

3.3.8 ExitState

Per completezza, *ExitState* è lo stato finale della macchina a stati, raggiungibile con l'evento *EvExit* a partire da ogni stato (ad eccezione di *PageRetrieval*, in quanto in esso non è prevista nessuna interazione da parte dell'utente).

4 Risultati sperimentali

In questo capitolo si analizzano i risultati ottenuti in riferimento alle attività di calibrazione e di recupero pagina.

4.1 Calibrazione

Si riportano alcune prove di calibrazione con pennarelli di vario colore. I test sono stati effettuati in tre differenti condizioni di luce:

1. Fonte unica non isotropa: luce artificiale con portata limitata (luce da scrivania);
2. Due fonti non isotrope: combinazione di luci artificiali con portata limitata disposte in modo tale da ridurre al minimo la presenza di ombre (due luci da scrivania);
3. Fonte unica isotropa: luce diurna (test effettuati all'aperto).

In Tabella 2 sono stati riportati i valori di *precision*, *recall* e *F-measure* relativi all'utilizzo di un insieme di colori in condizioni di luce differenti. A prescindere da quale colore esibisca il comportamento migliore rispetto alla segmentazione, i risultati ottenuti in termini di *precision* e *recall* risultano essere soddisfacenti per la maggior parte dei colori testati, così da permettere all'utente di svincolarsi dall'utilizzo di uno specifico colore.

Passando ad un veloce confronto tra i colori testati, l'arancione risulta essere segmentabile con successo in tutte le condizioni di luce previste. Sebbene alla luce diurna le differenze risultino meno marcate, in condizione di luce non isotropa colori come rosso, viola e rosa forniscono una risposta particolarmente inadeguata, tale da portare con probabilità ad un'errata interpretazione in fase di annotazione.

Colori	Condizione 1			Condizione 2			Condizione 3		
	p	r	$\mathcal{F}(p,r)$	p	r	$\mathcal{F}(p,r)$	p	r	$\mathcal{F}(p,r)$
Rosso	0.75	0.90	0.82	0.92	0.68	0.78	0.95	0.89	0.92
Verde	0.77	0.92	0.83	0.88	0.86	0.87	0.93	0.87	0.90
Marrone	0.72	0.94	0.82	0.91	0.82	0.86	0.89	0.89	0.87
Viola	0.78	0.91	0.84	0.76	0.67	0.71	0.87	0.88	0.88
Rosa	0.88	0.72	0.79	1.00	0.74	0.85	0.85	0.86	0.86
Arancione	0.92	0.85	0.89	0.99	0.82	0.90	0.91	0.93	0.92
Giallo	0.81	0.87	0.84	0.86	0.83	0.84	0.89	0.90	0.90
Celeste	0.89	0.89	0.89	0.89	0.84	0.86	0.84	0.87	0.86
Blu	0.86	0.76	0.80	0.80	0.91	0.85	0.87	0.88	0.88

Tabella 2: Tabella dei test con *precision*, *recall* e *F-measure*.

4.2 Recupero pagina

Allo scopo di analizzare quantitativamente il lavoro svolto, sono stati effettuati test che evidenziano l'accuratezza ed il tempo di esecuzione del metodo di recupero della pagina.

Gli esperimenti sono stati condotti utilizzando la webcam *Trust Cuby Webcam Pro - Titanium* come dispositivo di acquisizione. Quest'ultima è caratterizzata da una risoluzione di 1.3 megapixel ed ha permesso di delineare il comportamento dell'algoritmo in condizioni di acquisizione di bassa qualità.

Il dataset di documenti è composto dai 307 articoli della conferenza ICDAR11 (*2011 International Conference on Document Analysis and Recognition*). Questi presentano una struttura a doppia colonna ed una lunghezza media di 5 pagine, per un totale di 1540 pagine complessive.

4.2.1 Accuratezza

L'obiettivo principale dei test di accuratezza è stato quello di definire l'andamento dell'algoritmo di recupero pagina al variare dei due seguenti fattori critici:

numero di pagine registrate : l'aumentare dello spazio di ricerca, i.e. della dimensione del dataset, genera una degradazione del meccanismo di votazione poichè possono essere inseriti elementi similari alla pagina ricercata. Per effettuare test a dimensioni differenti, si è suddiviso il dataset in 10 set in modo tale che l'i-esimo ($i = 1, \ldots, 10$) fosse una collezione aleatoria di $i * (1540/10)$ pagine. Per ogni pagina da ricercare sono state così considerate 10 differenti configurazioni;

percentuale di righe scoperte : l'esclusione di porzioni della pagine genera una diminuzione del numero di feature estratte e perciò del numero di voti conseguibili. In particolare i test sono stati effettuati andando ad incrementare l'occlusione del 20% di righe ad ogni prova, per un totale di 5 differenti configurazioni per pagina da ricercare.

Per ognuna delle possibili configurazioni (50 in totale) derivanti dalla variazione dei fattori sopra esposti, sono stati valutati i risultati prendendo come test set un insieme casuale di 10 pagine[5].

Come indice di valutazione è stata scelta la **posizione media** che le pagine di test ricercate occupano nella classifica che l'algoritmo restituisce. Nel caso in cui nella classifica siano stati generati punteggi ex aequo, è stato attribuito un indice pari alla posizione dell'ultimo elemento a pari punteggio. Nella situazione in cui invece la pagina di test non sia presente all'interno della classifica, si è attribuito un punteggio uguale al peggior punteggio ottenuto a parità di dimensione del dataset.

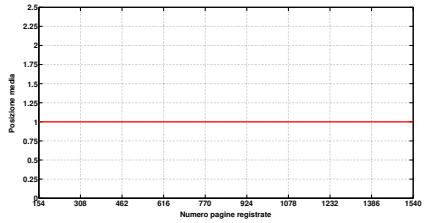
In figura 16 e nella tabella 3 sono riportate le posizioni medie ottenute per differenti percentuali di righe scoperte, al variare della dimensione del dataset.

I risultati denotano un ottimo comportamento generale del metodo che scala in maniera efficace per differenti dimensioni del dataset. L'elemento critico più influente è rappresentato dal numero di righe scoperte, che nel caso di percentuali ridotte (20 %), porta ad una diminuzione del numero di feature estratte tale che non vi è più una forte caratterizzazione di una pagina rispetto ad altre simili.

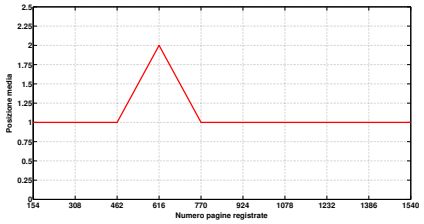
[5] le pagine da ricercare sono state scelte a priori rispetto alla creazione dei dataset di differenti cardinalità, di modo che queste fossero presenti in ogni singolo set.

Righe scoperte	Numero pagine registrate									
	154	308	462	616	770	924	1078	1232	1386	1540
100 %	1	1	1	1	1	1	1	1	1	1
80 %	1	1	1	2	1	1	1	1	1	1
60 %	1	1	1	1	1	1	1	1	1	1
40 %	1	1	1	1	1	1.2	1	1	1	1
20 %	1.2	1.7	1	1	1	1	1.9	2.2	1	1

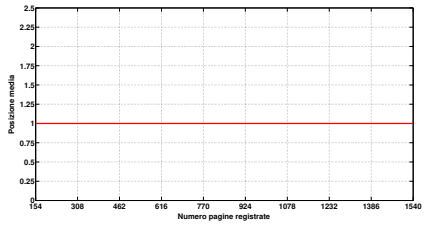
Tabella 3: Tabella dei test di accuratezza



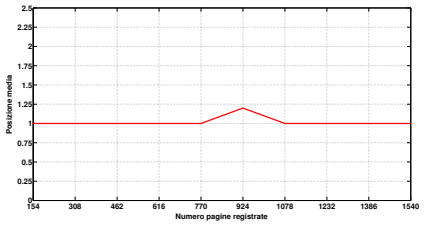
(a) 100 % righe scoperte



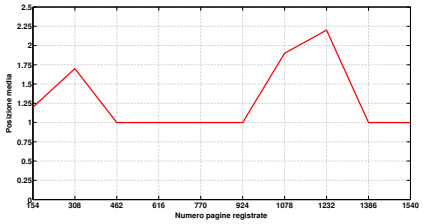
(b) 80 % righe scoperte



(c) 60 % righe scoperte



(d) 40 % righe scoperte



(e) 20 % righe scoperte

Figura 16: Grafici dei test di accuratezza

4.2.2 Tempo di esecuzione

L'analisi temporale dell'algoritmo di recupero pagina è stata condotta su un PC con CPU Intel Core2 Duo a 2.53GHz per singolo core, 4GB di memoria e un hard disk a stato solido.

In figura 17 e nella tabella 4 sono riportati i tempi di esecuzione ottenuti a pagina intera.

É possibile notare che il tempo di esecuzione scala linearmente con la dimensione del dataset, rimanendo tuttavia maggiore rispetto a quello riportato

	Numero pagine registrate									
	154	308	462	616	770	924	1078	1232	1386	1540
Tempo (s)	8.5	12	15.6	18.9	23	24.5	30.6	32.3	35.8	42.4

Tabella 4: Tabella dei tempi di recupero pagina (secondi)

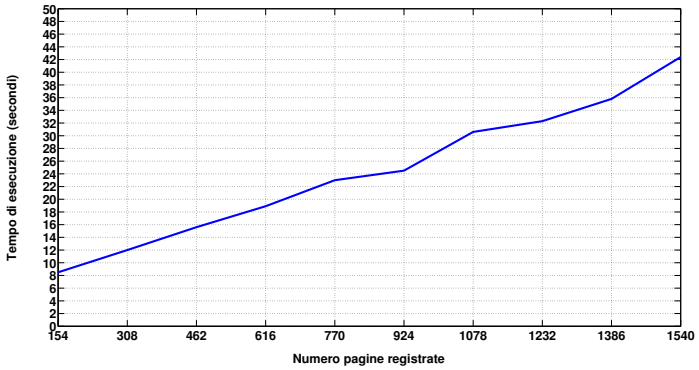


Figura 17: Grafico dei tempi di recupero pagina (secondi)

in [NKI06]. Le motivazione sono da ricercare nell'assenza di ottimizzazione nell'implementazione e nell'utilizzo di un DBMS dalle prestazioni non sufficienti per un recupero efficiente.

5 Esempio di esecuzione

Dopo una breve descrizione dell'ambiente di lavoro ideale per l'utilizzo di questa applicazione, in questo capitolo verrà riportato un semplice esempio di esecuzione illustrato. La struttura di supporto per la webcam (Figura 18) è stata realizzata in modo tale da facilitare l'interazione da parte dell'utente sulla pagina fisica.

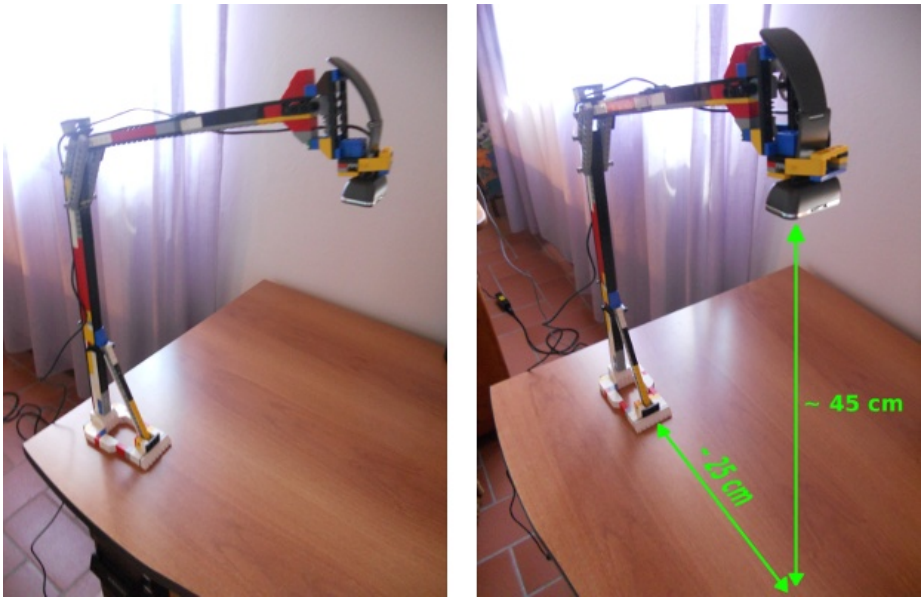


Figura 18: Supporto per la webcam

La struttura è stata realizzata con elementi del *LEGO*® in modo tale da essere regolabile a seconda delle condizioni di utilizzo: come riportato in figura, è stata stimata un'altezza di circa 45 cm per inquadrare una pagina intera e per una buona operabilità è stata stimata una distanza di circa 25 cm tra la base del supporto e il piede della perpendicolare dell'obiettivo della webcam.

Passiamo all'illustrazione dell'esempio di esecuzione. In Figura 19 viene riportata la schermata relativa allo stato iniziale della macchina descritta nel paragrafo 3.3.1. La prima finestra rappresenta (per questa e le prossime immagini) l'uscita della webcam senza alcuna elaborazione mentre la seconda finestra riporterà le elaborazioni proprie dello stato corrente. Nel caso di Figura 19, in *PageDiscovering* non si avrà alcuna transizione in quanto, in assenza di un documento, lo score associato al numero di linee risulta essere molto basso (8% rispetto alla soglia 80%).

Viceversa, in Figura 20, il sistema individua un numero di linee tale da asserire la presenza di un documento nella scena (87% rispetto alla soglia 80%). Come si può notare, in corrispondenza del superamento della soglia viene attivato un timer per il passaggio automatico allo stato di retrieval: questa scelta progettuale permetterà all'utente di limitare al minimo l'interazione e di comprendere l'evoluzione del sistema. Questo accorgimento è stato adottato nella maggior parte degli stati, in corrispondenza di eventi che determinano transizioni di stato.

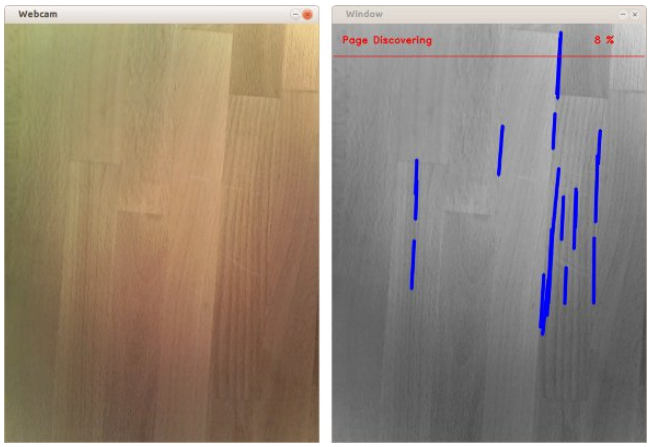


Figura 19: *PageDiscovering*: punto di ingresso della macchina a stati

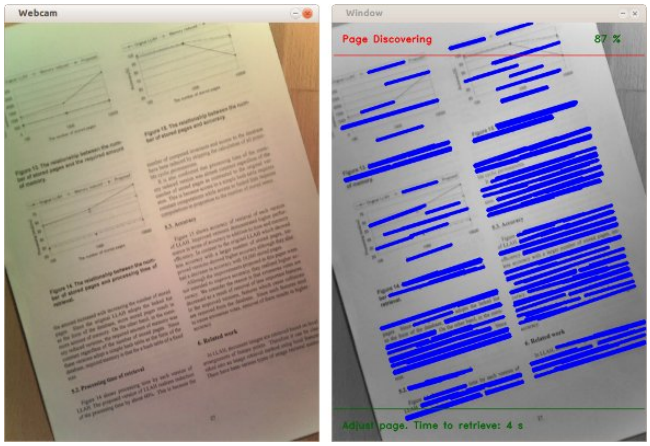


Figura 20: *PageDiscovering*: score alto in presenza di un documento

Tuttavia l'utente può dirigere il flusso di esecuzione in modo arbitrario grazie alla predisposizione dei comandi asincroni.

Al termine del conto alla rovescia o in seguito all'interazione dell'utente, si ha il passaggio allo stato *PageRetrieval*, nel quale il sistema esegue l'algoritmo di recupero, come mostrato in Figura 21.

L'operazione di retrieval, se andata a buon fine, si conclude con la transizione allo stato *PageTracking* (Figura 22). Si noti la comparsa di una terza finestra che rappresenta l'immagine JPEG della pagina recuperata.

In questo frame l'operazione di rettificazione riesce con successo a sovrapporre l'immagine da webcam sull'immagine originale. In generale tale algoritmo risulta essere particolarmente robusto, infatti i possibili disturbi, quali occlusione da parte di una mano o distorsione della pagina, devono essere consistenti per generare errore. Si veda a proposito le Figure 23 e 24 che riportano avvisi di malfunzionamento in condizioni al limite tra l'errore e la corretta rettificazione.

In entrambi i casi si noti il conto alla rovescia (che si concluderà con la transizione allo stato di errore *PageError*).

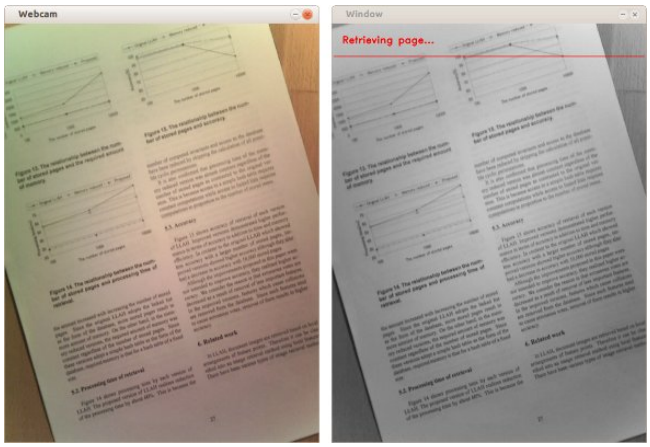


Figura 21: *PageRetrieval*: esecuzione dell'algoritmo di retrieval

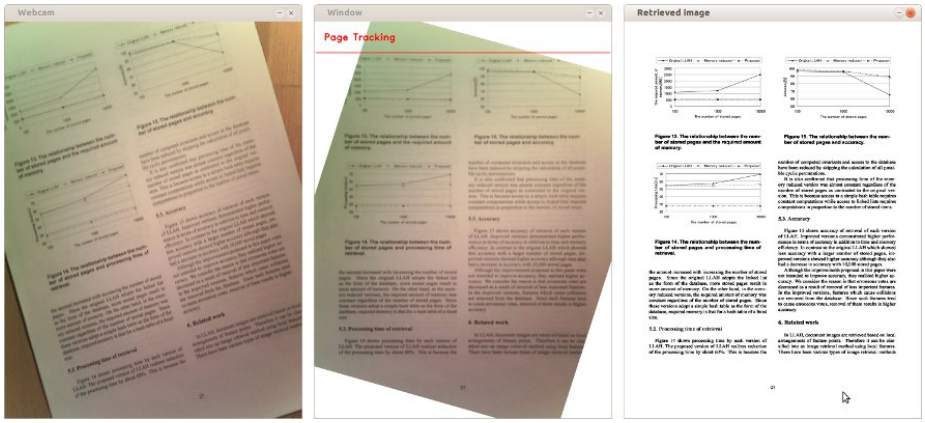


Figura 22: *PageTracking*: inseguimento della pagina avvenuto con successo

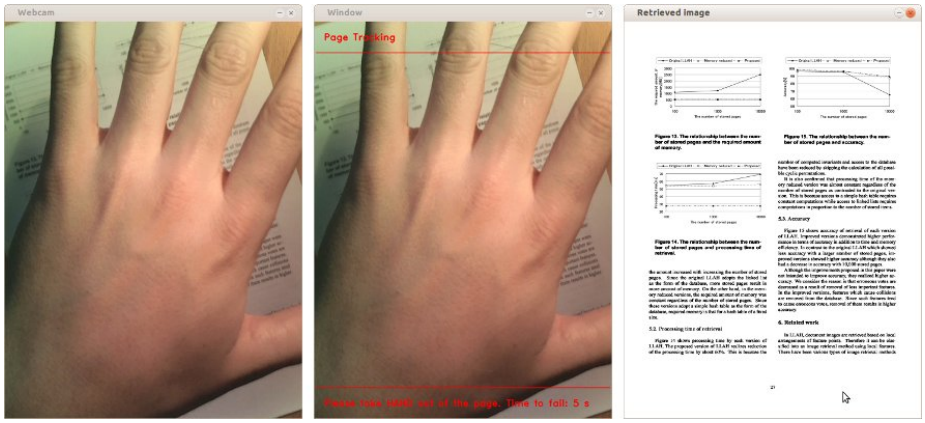


Figura 23: *PageTracking*: errore dovuto all'occlusione da parte di una mano

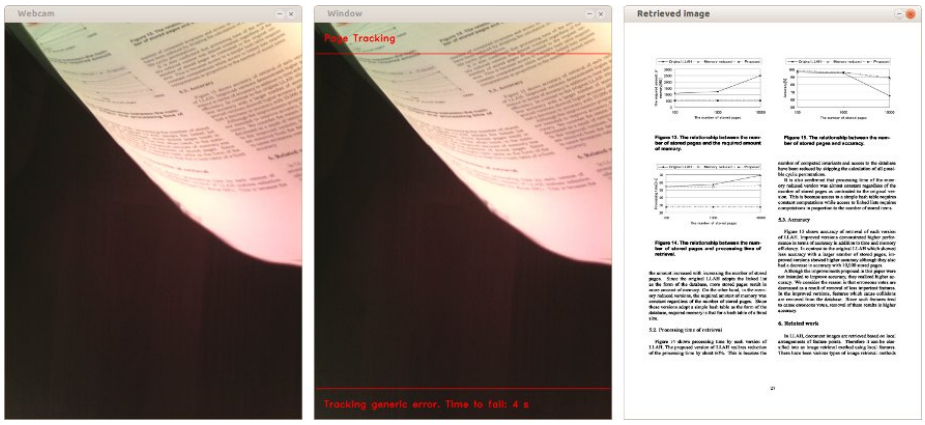


Figura 24: *PageTracking*: errore dovuto ad una generica distorsione della pagina

Tornando in condizioni ideali, ovvero senza alcun tipo di distorsione o di occlusione, si mostra il comportamento del sistema relativamente all'operazione di annotazione. In Figura 25 viene individuata una nuova annotazione (in accordo con quanto detto nel paragrafo 3.3.4) e viene attivato un timer che determina la transizione allo stato *PageAnnotation*.

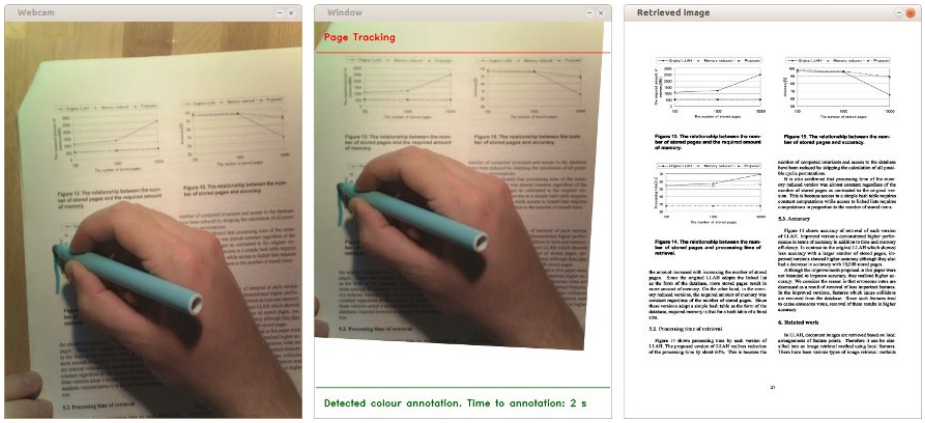


Figura 25: *PageTracking*: individuazione di una nuova annotazione

Il passaggio allo stato *PageAnnotation* permette all'utente di completare l'annotazione, come mostrato in Figura 26. I vettori disegnati in blu individuano punti dell'immagine in cui il flusso ottico è non nullo e pertanto il sistema non esegue nessuna transizione, rimanendo in attesa del completamento dell'annotazione.

L'utente, interagendo in modo asincrono o sincrono (rimuovendo il moto nell'immagine), pone fine all'operazione di annotazione (Figura 27).

Il completamento dell'annotazione determina la transizione indietro allo stato *PageTracking* e il salvataggio dell'annotazione. L'utente viene notificato dell'avvenuto salvataggio grazie all'evidenziazione dell'area interessata sull'immagine originale, come mostrato in Figura 28.

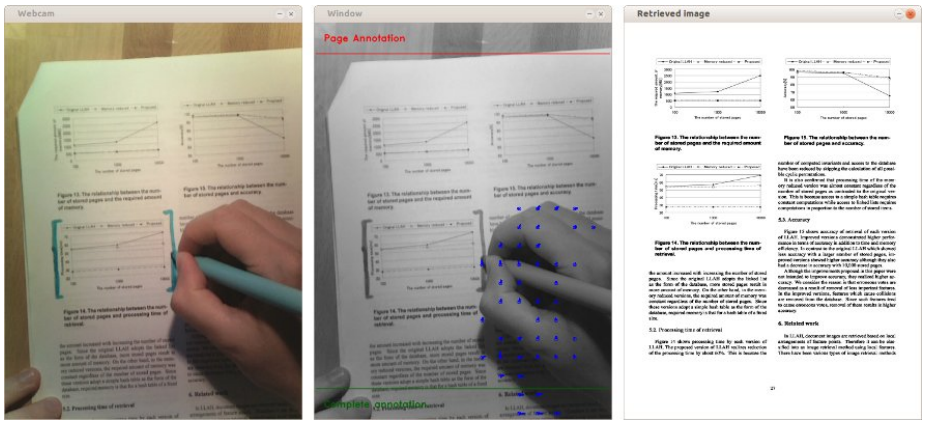


Figura 26: *PageAnnotation*: il sistema interpreta l'annotazione in via di completamento grazie alla presenza di moto

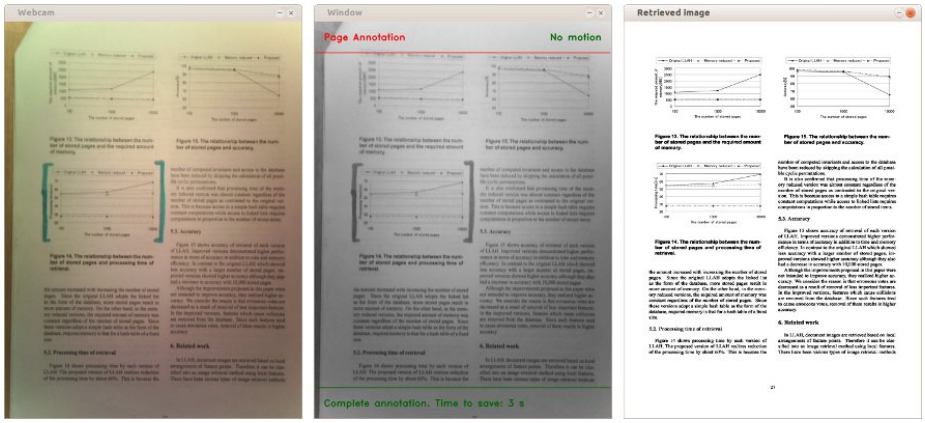


Figura 27: *PageAnnotation*: flusso ottico nullo, nessun moto nell'immagine

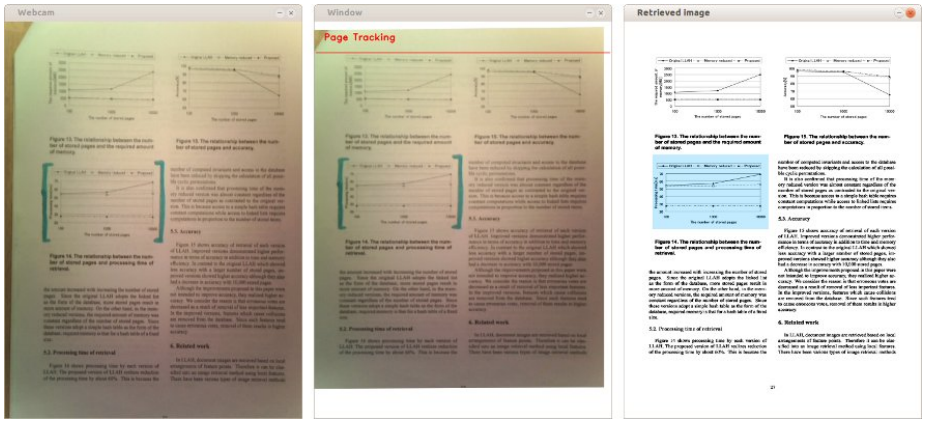


Figura 28: *PageTracking*: salvataggio dell'annotazione e feedback all'utente

Una volta effettuata una serie di annotazioni riguardanti un insieme di pagine di uno stesso documento, l'utente può comporre un nuovo documento (riportante il testo annotato) utilizzando in *PageTracking* l'apposito comando asincrono per la transizione allo stato *PdfCreation*. In esso l'utente attenderà il completamento dell'operazione prima di poter tornare allo stato *PageTracking*, come riporta la Figura 29.

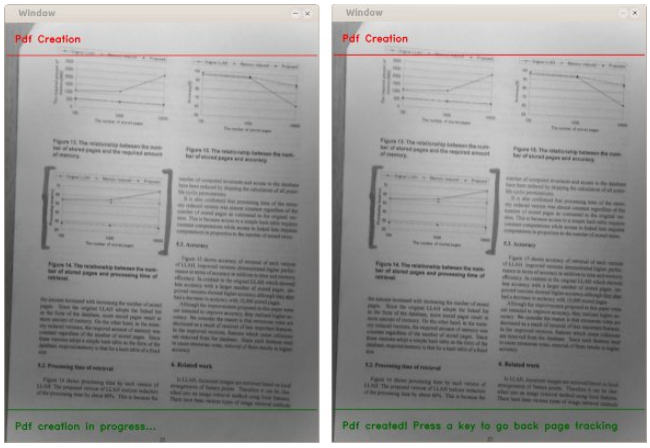


Figura 29: *PdfCreation*: creazione in corso e completamento

Infine, per cambiare pagina da annotare, l'utente può decidere di utilizzare il comando predisposto o rimuovere il documento per generare errore in *PageTracking*: in ogni caso si ha la transizione allo stato *PageError*, dal quale è possibile operare il reset della macchina a stati con la pressione di un tasto (Figura 30).

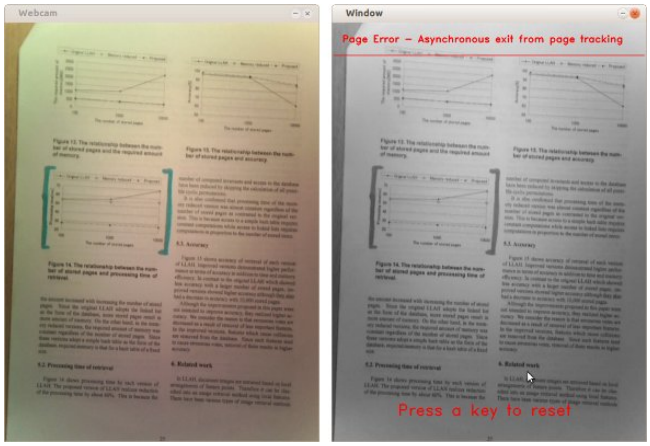


Figura 30: *PageError*: stato di errore

6 Conclusioni e sviluppi futuri

Il progetto ha riguardato lo sviluppo di un'applicazione desktop per il retrieval e l'annotazione di pagine, in modo tale da creare nuovi documenti riportanti solo le parti interessate da annotazioni. Il requisito principale è stato l'utilizzo di un sistema di acquisizione real-time come la webcam. Dapprima sono stati descritti i fondamenti teorici e pratici dell'algoritmo *LLAH* per la registrazione e il recupero dei documenti e nel seguito è stata riportata in modo dettagliato la progettazione e l'implementazione dell'applicazione, suddivisa nei suoi tre aspetti primari, ovvero la registrazione, la calibrazione e l'annotazione.

La prima è risultata particolarmente agevole, grazie anche al buon lavoro svolto dai nostri predecessori Lapini e Bandini.

La seconda ha permesso l'applicazione di tecniche e conoscenze riguardanti i metodi di ottimizzazione, in modo tale da definire un algoritmo elegante di ricerca di soglie per la corretta segmentazione dei tratti di annotazione.

La terza è risultata molto interessante dal punto di vista tecnico: dallo stato *PageDiscovering* fino allo stato *PageAnnotation*, passando per il nodo cruciale *PageTracking*, è stato possibile avere a che fare con un nutrito insieme di tecniche derivanti dalle prevalenti discipline riguardanti l'elaborazione delle immagini, come la rettificazione di piani per la sovrapposizione della pagina inquadrata dalla webcam e quella originale, il calcolo del flusso ottico per la stima del moto totale in *PageAnnotation* o la tecnica di Hough per la ricerca di linee in una maschera binaria.

La bontà della scelta degli strumenti di elaborazione è stata confermata ampiamente in sede di implementazione, dove le principali difficoltà hanno riguardato solo piccoli dettagli tecnici, risolti grazie all'ausilio delle documentazioni [Tea13; Boo; Sql].

Da un punto di vista del funzionamento, abbiamo riscontrato una buona efficacia in tutti gli aspetti più rilevanti: l'operazione di retrieval si è rivelata precisa e accurata nonostante la bassa risoluzione della webcam (*Trust Cuby Webcam Pro - Titanium* da 1.3 MegaPixels) fornendo il risultato giusto anche in presenza di scarsa luminosità o in assenza di parti della pagina; la rettificazione ha mostrato una buona robustezza e precisione; l'annotazione e la creazione dei documenti si è rivelata corretta e sufficientemente resistente al rumore o a comportamenti al limite della praticabilità. In particolare l'operazione di calibrazione ha permesso una segmentazione corretta dei tratti di annotazione, anche in variegate condizioni di luce e in presenza di corpi estranei quali la mano.

Possibili miglioramenti futuri possono riguardare l'estensione di questa applicazione all'utilizzo di webcam più performanti e capaci di segmentare tratti più fini. Con una maggiore risoluzione è possibile quindi riportare del testo e non solo box di annotazione per il filtraggio di parti di testo.

In fase di test, relativamente ad una specifica pagina, è stato utilizzato un frame differente per ogni test (coppia dataset e percentuale di righe scoperte): un miglioramento consiste nell'utilizzare un unico frame per ogni test relativo alla stessa pagina, così da rendere i risultati indipendenti dalle componenti di rumore presenti nella fase di acquisizione da webcam.

L'interfaccia utente può essere estesa con l'adozione di *Qt*, una libreria multipiattaforma per lo sviluppo di programmi con interfaccia grafica tramite l'uso di

widget, che permetta una presentazione maggiormente compatta delle finestre di lavoro.

Un altro possibile miglioramento potrebbe consistere nella realizzazione di un modulo di correzione nel quale interpretare segni sulle parole come modifiche o cancellazioni della stessa e riflettere tale variazione su un documento in LaTeX opportunatamente predisposto.

Infine potrebbe essere progettata un'integrazione tra la precedente proposta e moduli di riconoscimento di testo scritto a mano (di cui abbiamo avuto modo di apprenderne lo sviluppo da parte di alcuni nostri colleghi studenti) per costruire una macchina di correzione real-time di documenti collegati ai rispettivi sorgenti in LaTeX.

Riferimenti bibliografici

[Boo] *Boost Documentation*. 2013. URL: `http://www.boost.org/doc/`.

[Low03] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints*. 2003.

[Nak] Iwamura Nakai Kise. "Hashing with Local Combinations of Feature Points and Its Application to Camera-Based Document Image Retrieval". In: ().

[NKI06] Tomohiro Nakai, Koichi Kise e Masakazu Iwamura. "Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval". In: *In Lecture Notes in Computer Science (7th International Workshop DAS2006*. Springer, 2006, pp. 541–552.

[NKI07] Tomohiro Nakai, Koichi Kise e Masakazu Iwamura. "Camera based document image retrieval with more time and memory efficient LLAH". In: *Proc. CBDAR07*. 2007.

[Sql] *SQLite Documentation*. 2013. URL: `http://www.sqlite.org/docs.html`.

[Tea13] OpenCV Dev Team. *OpenCV Documentation*. 2013. URL: `http://http://docs.opencv.org/index.html`.

[TKI11] Kazutaka Takeda, Koichi Kise e Masakazu Iwamura. "Real-Time Document Image Retrieval for a 10 Million Pages Database with a Memory Efficient and Stability Improved LLAH". In: *Proceedings of the 2011 International Conference on Document Analysis and Recognition*. IEEE Computer Society, 2011.

A Installazione del sistema

Nelle seguenti sezioni sono riportate le librerie e i flag necessari per poter compilare il codice sorgente C++ del lavoro. Si evidenzia inoltre la necessità dei pacchetti *convert* e *pdfinfo* per la manipolazione di documenti PDF.

A.1 Libreria OpenCV

OpenCV[6] è una libreria open source scritta in C/C++, per i sistemi operativi Linux, Windows e MacOS, con l'obiettivo principale di fornire strumenti per la risoluzione di problemi di computer vision.

La struttura prevede uno strato di basso livello dove sono implementati i tipi di dato fondamentali e, a partire da questo, sono implementate funzionalità di alto livello tramite le quali si possono tipicamente risolvere problemi di visione computazionale complessi.

Il Listato 1 riporta le istruzioni necessarie per l'installazione della versione attuale di OpenCV (al momento della scrittura v2.4.5) e delle sue dipendenze, in ambiente Linux.

Listato 1: Script per l'installazione di OpenCV 2.4.5 in ambiente Linux

```
arch=$(uname -m)
if [ "$arch" == "i686" -o "$arch" == "i386" -o "$arch" == "i486" -o "
    $arch" == "i586" ]; then
flag=1
else
flag=0
fi
echo "Installing OpenCV 2.4.5"
mkdir OpenCV
cd OpenCV
echo "Removing any pre-installed ffmpeg and x264"
sudo apt-get remove ffmpeg x264 libx264-dev
echo "Installing Dependencies"
sudo apt-get install libopencv-dev
sudo apt-get install build-essential checkinstall cmake pkg-config yasm
sudo apt-get install libtiff4-dev libjpeg-dev libjasper-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev
    libdc1394-22-dev libxine-dev libgstreamer0.10-dev libgstreamer-
    plugins-base0.10-dev libv4l-dev
sudo apt-get install python-dev python-numpy
sudo apt-get install libtbb-dev
sudo apt-get install libqt4-dev libgtk2.0-dev
sudo apt-get install libfaac-dev libmp3lame-dev libopencore-amrnb-dev
    libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev
echo "Downloading x264"
wget ftp://ftp.videolan.org/pub/videolan/x264/snapshots/x264-snapshot
    -20121114-2245-stable.tar.bz2
tar -xvf x264-snapshot-20121114-2245-stable.tar.bz2
cd x264-snapshot-20121114-2245-stable/
echo "Installing x264"
if [ $flag -eq 1 ]; then
./configure --enable-static
else
./configure --enable-shared --enable-pic
fi
make
sudo make install
cd ..
echo "Downloading ffmpeg"
wget http://ffmpeg.org/releases/ffmpeg-0.11.2.tar.bz2
echo "Installing ffmpeg"
tar -xvf ffmpeg-0.11.2.tar.bz2
```

[6] Disponibile all'indirizzo `http://opencv.org`.

```
38 cd ffmpeg-0.11.2/
39 if [ $flag -eq 1 ]; then
40 ./configure --enable-gpl --enable-libfaac --enable-libmp3lame --enable-
       libopencore-amrnb --enable-libopencore-amrwb --enable-libtheora --
       enable-libvorbis --enable-libx264 --enable-libxvid --enable-nonfree
        --enable-postproc --enable-version3 --enable-x11grab
41 else
42 ./configure --enable-gpl --enable-libfaac --enable-libmp3lame --enable-
       libopencore-amrnb --enable-libopencore-amrwb --enable-libtheora --
       enable-libvorbis --enable-libx264 --enable-libxvid --enable-nonfree
        --enable-postproc --enable-version3 --enable-x11grab --enable-
       shared
43 fi
44 make
45 sudo make install
46 cd ..
47 echo "Downloading v4l"
48 wget http://www.linuxtv.org/downloads/v4l-utils/v4l-utils-0.8.9.tar.bz2
49 echo "Installing v4l"
50 tar -xvf v4l-utils-0.8.9.tar.bz2
51 cd v4l-utils-0.8.9/
52 make
53 sudo make install
54 cd ..
55 echo "Downloading OpenCV 2.4.5"
56 wget -O OpenCV-2.4.5.tar.gz http://downloads.sourceforge.net/project/
       opencvlibrary/opencv-unix/2.4.5/opencv-2.4.5.tar.gz
57 echo "Installing OpenCV 2.4.5"
58 tar -xvf OpenCV-2.4.5.tar.gz
59 cd opencv-2.4.5
60 mkdir build
61 cd build
62 cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D
       WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D
       INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D
       BUILD_EXAMPLES=ON -D WITH_QT=ON -D WITH_OPENGL=ON ..
63 make
64 sudo make install
65 sudo sh -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'
66 sudo ldconfig
67 echo "OpenCV 2.4.5 ready to be used"
```

A.2 Librerie Boost

Le Boost[7] sono una collezione di librerie C++ open source che vanno ad estendere le funzionalità della libreria standard. In dettaglio, sono riportate le librerie compilate utilizzate nel progetto[8]:

- *filesystem*: facilita la richiesta e manipolazione di path, file e directory di sistema;
- *serialization*: implementazione delle funzionalità di serializzazione e deserializzazione di strutture dati;
- *program options*: permette di ottenere, in modo semplificato, le opzioni del programma;
- *thread*: fornisce un'interfaccia C++ portabile per il multi-threading;
- *system*: supporto al sistema operativo, include ad esempio condizioni di errore generate dal sistema operativo e API di basso livello.

[7] Disponibili all'indirizzo `http://www.boost.org/`.

[8] Sono state utilizzate anche librerie boost non compilate (.hpp) quali iterator, archive, lexical cast, algorithm.

A.3 Libreria SQLite3

Una volta installato il paccetto *sqlite3*, è possibile prelevare all'indirizzo `http://www.sqlite.org/download.html` la libreria per la manipolazione di database SQLite con i linguaggi di programmazione C/C++.

A.4 Flag di compilazione

Si riportano di seguito alcuni flag necessari alla compilazione:

- **-std=c++11**: flag da impostare nel compilatore GCC C++, consente di compilare rispetto allo standard ISO C++ 2011 che introduce una serie di nuove caratteristiche nello standard C++;
- **-pthread**: fornisce supporto per il multithreading con la libreria pthreads.

B Manuale utente

In questa sezione verranno descritte le modalità d'uso dell'applicazione, con la specifica dei comandi asincroni messi a disposizione dell'utente.

Si presentano le possibili modalità d'uso dell'applicazione *TBDAnnotation* nel Listato 2:

Listato 2: Usage dell'applicazione TBDAnnotation

```
1  Usage:
2  TBDAnnotation calibrate [-h] [--output CALIBRATION_OUTPUT_PATH]
3  TBDAnnotation register [-h] [--documents DOCUMENTS_PATH]
4      [--output DATASET_OUTPUT_PATH] [--nSets N_SETS]
5      [--nRetained N_RETAINED]
6  TBDAnnotation annotate [-h] [--dataset DATASET_PATH]
7      [--output PDF_OUTPUT_PATH] [--calibration CALIBRATION_FILE]
8  TBDAnnotation test [-h] [--dataset DATASET_PATH]
9      [--pageName PAGE_NAME]

11 -- Option Descriptions --

13 Subcommands:
14    subcommand              subcommand to execute: calibrate, register,
15                            annotate, test

17 Option Arguments:
18    -h [ --help ]           print usage messages

20    --documents             path to pdf documents
21    --output                output files directory
22    --nSets                 number of datasets.
23                            The i-th dataset has size i*nPages/nSets.
24                            Optional, default=1
25    --nRetained             number of same pages that will be present in
26                            every dataset. Required if nSets is setted
27    --dataset               dataset files directory
28    --pageName              name of page to retrieve. Required if test
29                            mode enabled
30    --calibration           calibration file path. Optional
```

B.1 Calibrazione

Il sottocomando di calibrazione *calibrate* determina le soglie per la segmentazione del colore di annotazione che si intende utilizzare. *CALIBRATION_OUTPUT_PATH* è l'unico argomento da specificare e rappresenta la cartella dove verrà salvato il file di calibrazione.

B.2 Registrazione

Il sottocomando di registrazione *register* presenta come parametri:

- *DOCUMENTS_PATH*: il percorso di input dei documenti, può rappresentare sia un singolo documento sia una cartella contenente un insieme di documenti;
- *DATASET_OUTPUT_PATH*: il percorso di output dei dataset;
- *N_SETS*: il numero di dataset di dimensioni differenti da creare, opzionale, 1 di default;
- *N_RETAINED*: il numero di pagine che devono essere comuni a tutti i dataset, richiesto se N_SETS diverso da 1.

Nel caso in cui *N_SETS* sia differente da 1, è richiesta la specifica di *N_RETAINED*. In questo caso la suddivisione dei dataset e la scelta seguono quanto descritto nel paragrafo 4.2.1.

Il flusso di esecuzione della registrazione è sequenziale e non richiede nessuna interazione da parte dell'utente: ogni pagina viene convertita in formato JPEG e registrata come descritto nella sezione 3.1.

B.3 Annotazione

Il sottocomando di annotazione *annotate* viene eseguito con la specifica del dataset di riferimento *DATASET_PATH* (opportunatamente creato), della cartella di destinazione *PDF_OUTPUT_PATH* dei documenti da comporre a partire dalle annotazioni e del file di calibrazione *CALIBRATION_FILE* (se presente).

L'esecuzione di questo sottoprogramma è in accordo con il funzionamento della macchina a stati in Figura 8. In questo paragrafo si esplicitano i comandi necessari per un'interazione asincrona rispetto all'evoluzione del sistema.

- **PageDiscovering**: sono disponibili i seguenti comandi

  s - esegue la transizione immediata allo stato di retrieval *PageRetrieval*;

  ESC - esce dall'applicazione.

- **PageRetrieval**: non è disponibile nessun comando.

- **PageTracking**: sono disponibili i seguenti comandi

  f - termina l'inseguimento della pagina determinando la transizione allo stato *PageError* (dal quale è possibile reinizializzare il sistema);

  s - esegue la transizione allo stato *PdfCreation* per la creazione del documento a partire dalle annotazioni;

  ESC - esce dall'applicazione.

- **PageAnnotation**: sono disponibili i seguenti comandi

  s - termina l'annotazione ed esegue la transizione immediata allo stato *PageTracking*, consentendone il salvataggio;

  ESC - esce dall'applicazione.

- **PdfCreation**: al termine della creazione del documento saranno disponibili i seguenti comandi

  key - torna allo stato *PageTracking* con la pressione di un tasto;

  ESC - esce dall'applicazione.

- **PageError**: sono disponibili i seguenti comandi

  key - reset della macchina a stati (transizione allo stato iniziale *PageDiscovering*) con la pressione di un tasto;

  ESC - esce dall'applicazione.

B.4 Test

Il sottocomando *test* richiede il dataset di riferimento *DATASET_PATH* e la specifica del nome della pagina che il sistema deve essere in grado di ottenere nella fase di recupero.

Il test evolve secondo una versione ridotta della macchina a stati in Figura 8, in quanto si ha il passaggio diretto allo stato finale *ExitState* una volta conclusa l'operazione di retrieval in *PageRetrieval*.

Il sistema produce un log utile per la valutazione del test, in cui viene riportata la posizione assunta dalla pagina indicata in input.

C Organizzazione file e cartelle

La cartella di un dataset al percorso *DATASET_PATH* ha la seguente struttura:

- *DATASET_PATH/images/*: cartella contenente le pagine dei documenti in formato JPEG che sono risultato dell'operazione di conversione in fase di registrazione;
- *DATASET_PATH/masks/*: cartella contenente le maschere binarie delle immagini in */images* nelle quali il foreground rappresenta l'area annotata e di interesse;
- *DATASET_PATH/annotations/*: cartella contenente le immagini delle pagine filtrate con le maschere delle annotazioni. L'esistenza di questa sottocartella e del suo contenuto è giustificata dalla possibilità di comporre il nuovo documento avendo già a disposizione le pagine di interesse, senza doverle, ad ogni creazione del PDF, filtrare con le maschere binarie.
- *DATASET_PATH/pageHashTable.db*: database SQLite utile per l'operazione di retrieval;

In fase di registrazione è necessario distinguere rispetto ai due seguenti casi.

- $\boldsymbol{N\_SETS = 1}$: viene creato o aggiornato il dataset al percorso *DATASET_OUTPUT_PATH*. Se la cartella *DATASET_OUTPUT_PATH* e/o la sottocartella *DATASET_OUTPUT_PATH/images/* non sono presenti, il sistema provvede alla creazione. Lo stesso criterio viene adottato per il database *DATASET_OUTPUT_PATH/pageHashTable.db*.
- $\boldsymbol{N\_SETS > 1}$: al percorso *DATASET_OUTPUT_PATH* viene definita una cartella *images* con tutte le pagine convertite in JPEG dei documenti presenti in *DOCUMENTS_PATH* e una cartella *retained* contenente le pagine comuni a tutti i dataset. Infine vengono create *N_SETS* cartelle, con prefisso *dataset_* e suffisso il numero di pagine di cui si comporrà il dataset (paragrafo 4.2.1), ognuna delle quali ha una struttura pari a quella descritta in precedenza in riferimento al percorso *DATASET_PATH*.

In fase di annotazione è assolutamente richiesta l'esistenza delle cartelle e dei file

DATASET_PATH

DATASET_PATH/images/

DATASET_PATH/pageHashTable.db

indispensabili per l'operazione di retrieval. Invece le cartelle

DATASET_PATH/masks/

DATASET_PATH/annotations/

vengono predisposte nel caso non esistessero.