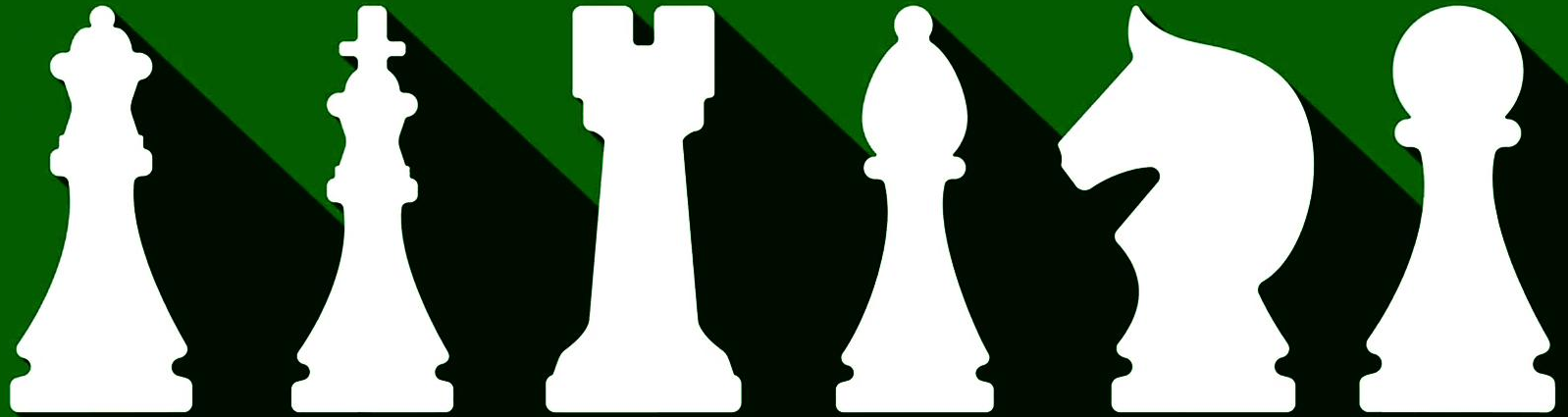
progetto d’esame di elaborazione delle immagini

2017-2018



Micali Giovanni 797654

Parini Pietro 794146

Indice

Introduzione - (1.1)

Dopo aver analizzato il problema abbiamo deciso di dividere il progetto in due macro parti:

1. **Individuazione** ed **estrapolazione** della scacchiera all'interno dell'immagine.
2. **Riconoscimento** dei pezzi della scacchiera e **creazione** della stringa FEN.

Giunti a questa suddivisione ci siamo documentati per avere un'idea di come si potessero implementare le varie parti del progetto.

Divisione dei compiti - (1.2)

Parini Pietro → **Individuazione** ed **estrapolazione** della scacchiera all'interno dell'immagine.

Micali Giovanni→ **Riconoscimento** dei pezzi della scacchiera e **creazione** della stringa FEN.

Insieme → generatore di **matrici di confusione**, **analisi dei risultati** e conclusioni.

Fase implementativa - (1.3)

- Suddivisione delle immagini in:
  1. Per lo sviluppo del codice abbiamo usato circa il 40% delle immagini
  2. Abbiamo effettuato i test sul rimanente 60%
- Creazione di una pipeline completa funzionante sui casi più semplici.
- Analisi delle immagini che non creavano le stringhe FEN corrette, svolte a capire quale fosse il problema.
- Ricerca di una soluzione che portasse al miglioramento dell'algoritmo per avere una maggior precisione.
- Test sulle immagini rimanenti.
- Aggiunta di immagini al dataset (20 immagini) volte ad ampliarlo e avere tutti i pezzi sia sulle celle nere che su quelle bianche (es. la regina nera era assente).
- Generata la matrice di confusione.

Cosa abbiamo usato? - (1.4)

Linguaggio e ambiente di sviluppo

MATLAB

Perché?

E' il linguaggio che abbiamo utilizzato durante il corso.

Documentazione progetto

MARKDOWN

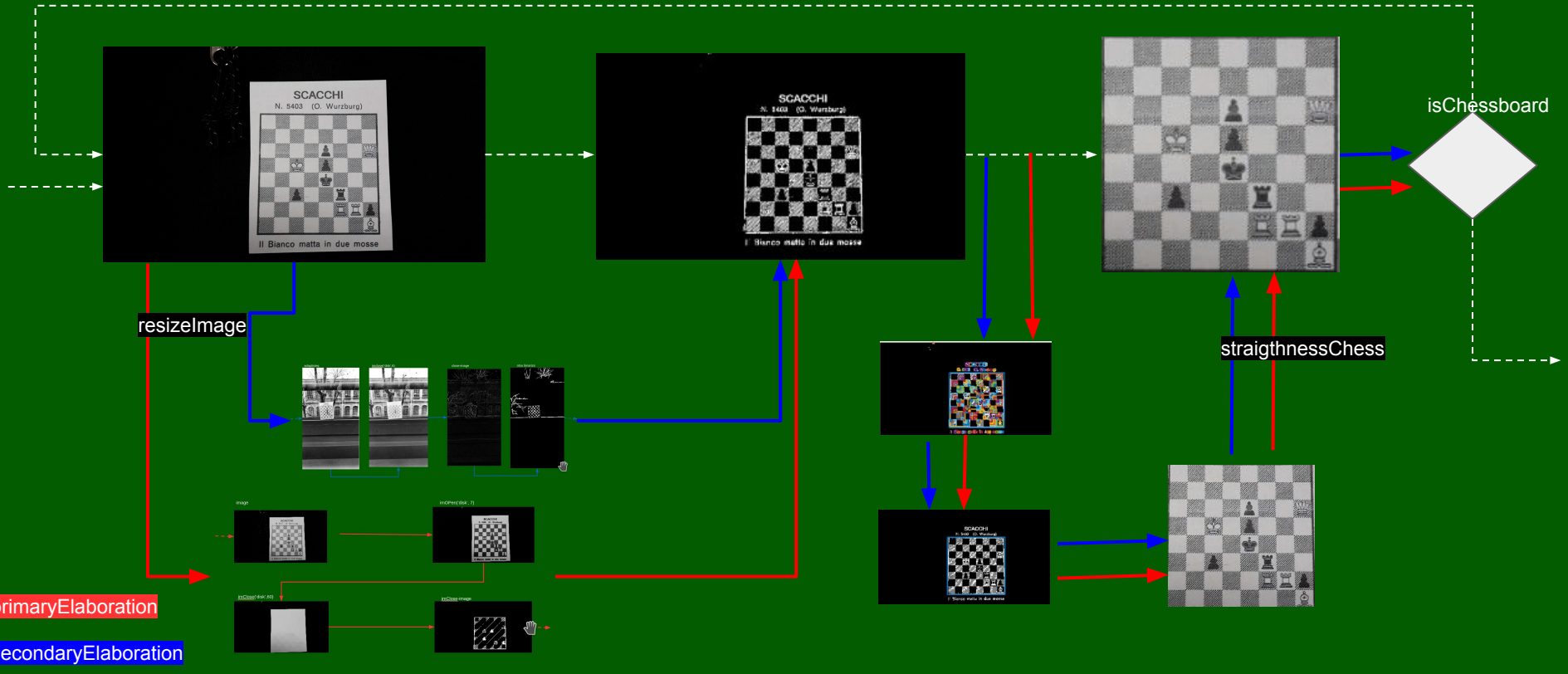
Nel file README.md, presente nella repo del progetto, è disponibile la documentazione specifica di ogni funzione e script di testing.

1. Introduzione
2. **Parte prima**
   - Individuazione scacchiera
   - Raddrizzamento scacchiera
3. Parte seconda
   - individuazione pezzi
   - generazione stringa FEN
4. Analisi dei dati
5. Conclusioni

Dettaglio prima parte - (2.1)

- Ridimensionamento immagini
- Pipeline di pre-elaborazione principale
- Pipeline di pre-elaborazione secondaria
- Ricerca delle bounding-box ~ quadrate
- Scelta della bounding-box della scacchiera
- Maschera della presunta scacchiera
- Ritaglio della presunta scacchiera
- Raddrizzamento scacchiera nella sua bounding-box
- Confronto per verificare che sia una scacchiera e nel caso chiamare la seconda procedura di pre-elaborazione

Diagramma generale parte prima - (2.2)



Ridimensionamento immagini - (2.3)

resizeImage

funzione che si occupa di fare una resize dell'immagine in modo da rendere più veloce e più scalabile tutta la computazione, e di lavorare in un formato standardizzato. Diminuisce tutte le immagini a un massimo di dimensioni dei 2 lati di 1042 px tenendo le proporzioni originali.

- input: immagine del dataset di immagini da analizzare
- output: array di due elementi , immagine di nuove dimensione, scala usata per le immagini
- parametri: misura massima degli assi dell'immagine = 1000+42
- matlab functions: size, imresize
- test: testResize

view official doc

Pipeline di pre-elaborazione principale - (2.4)

primaryElaboration

funzione, prima possibilità di elaborazione, pensata per poter individuare le scacchiere là dove c'è la presenza di un background a texture.

- input: immagine già nella dimensione stabilita per l'elaborazione
- output: immagine in bianco e nero pronta per il riconoscimento dei componenti
- parametri: misure dischi : 7,60
- matlab functions: size, rgb2gray, im2double, imopen, imclose, strel, imbinarize
- test: testPrimaryElaboration

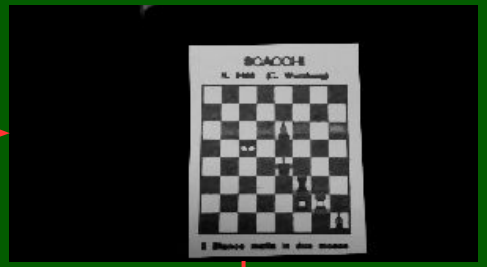
view official doc

Pipeline di pre-elaborazione principale - (2.5)

image

rgb2gray

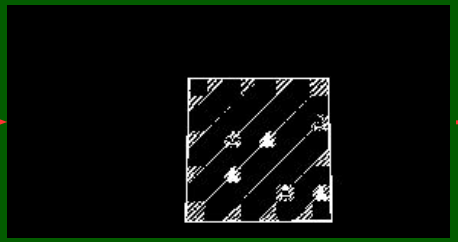
imOpen(‘disk’, 7)



imClose(‘disk’,60)

imbinarize(imClose-image)

ottimizzazione per futura computazione



Pipeline di pre-elaborazione secondaria - (2.6)

secondaryElaboration

funzione, seconda possibilità di elaborazione, elabora tramite equalizzazione dell'istogramma e sogliatura immagine con soglia individuata tramite metodo Otsu.

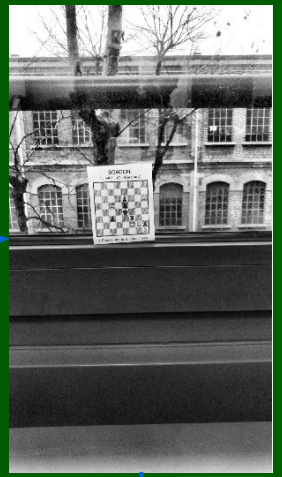
- input: immagine già nella dimensione stabilita per l'elaborazione, boolean per il testing
- output: immagine in bianco e nero pronta per il riconoscimento delle componenti
- parametri: misura disco: 6
- matlab functions: size, rgb2gray, adapthisteq, imclose, strel, graythresh, imbinarize
- test: testSecondaryElaboration

view official doc

Pipeline di pre-elaborazione secondaria - (2.7)

rgb2gray

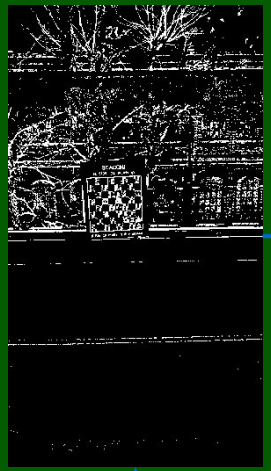
adaptisteq



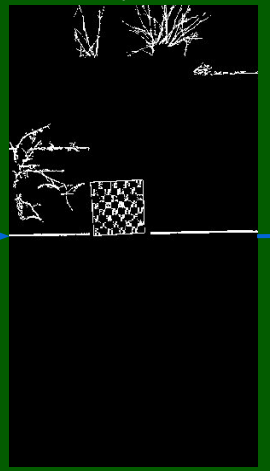
imclose('disk',6)

close-adaptisteq

otsu binarize



ottimizzazione per futura computazione



Individuazione scacchiera - (2.8)

chessDiscover

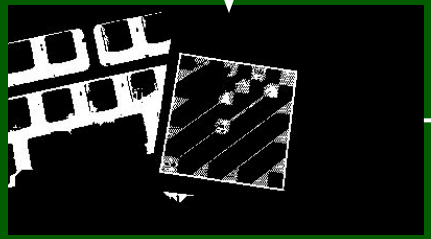
funzione che si occupa di individuare la scacchiera. Sfruttando le bounding-box va a cercare le bounding-box quadrate, con un errore del 18%, per poi selezionare quella più grande.

- input: immagine elaborata, la scala dell'immagine elaborata (output di resizeImage), immagine originale.
- output: la probabile chessboard sotto forma di struct contenente boundingbox, convexarea, convexImage ed scacchiera ritagliata dall'immagine originale
- parametri: errore di approssimazione dei lati = 0.18
- matlab functions: regionprops, sort, fliplr
- test: testChessDiscover

view official doc

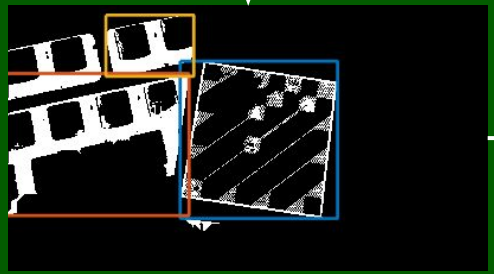
Individuazione scacchiera - (2.9)

immagine pre-elaborata

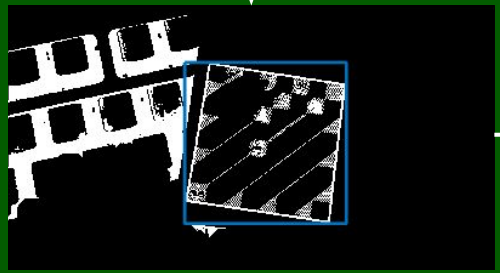


errorAprox=0.18;
if abs((width - height) / (height + width)) < errorAprox

quadrati trovati



probabile scacchiera



Raddrizzamento scacchiera - (2.10)

cornersMask(aux)

funzione che si occupa di individuare i 4 corner di una maschera binaria che riceve in input.

- input: maschera binaria figura bianco su sfondo nero
- output: matrice con i 4 corner
- parametri: 0
- matlab functions: find
- invocata da: straightnessChess

straightensChess

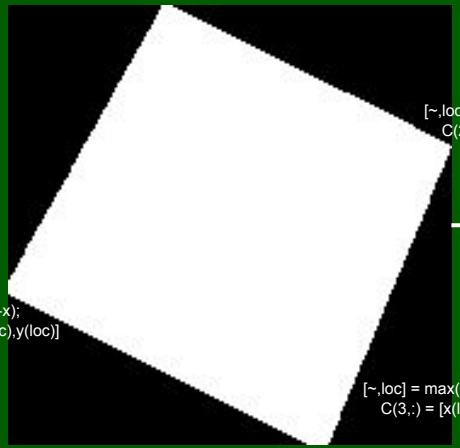
funzione che si occupa di raddrizzare la scacchiera all'interno della bounding-box che la contiene.

- input: immagine boundingBox da raddrizzare , maschera immagine boundingbox da raddrizzare
- output: immagine raddrizzata
- parametri: 0
- matlab functions: size, imresize, fitgeotrans , imwarp, regionprops, imcrop
- test: testStraightensChess

view official doc

Raddrizzamento scacchiera - (2.11)

[~,loc] = min(y+x);
C = [x(loc),y(loc)];



[~,loc] = max(y-x);
C(4,:) = [x(loc),y(loc)]

[~,loc] = min(y-x);
C(2,:) = [x(loc),y(loc)];

[~,loc] = max(y+x);
C(3,:) = [x(loc),y(loc)];

left = mean(Corners([1 4],1));
right = mean(Corners([2 3],1));
top = mean(Corners([1 2],2));
bottom = mean(Corners([3 4],2));

fitgeotrans(Corners ,newCorners,'projective');

Algoritmo di scelta pre-elaborazione - (2.12)

chooseElaboration

funzione che si occupa di stabilire se la presunta scacchiera trovata con primaryElaboration, dopo ritaglio e raddrizzamento, è effettivamente una scacchiera. In caso contrario chiama secondaryElaboration e stabilisce il risultato migliore.

- input: immagine ridimensionata, scala del ridimensionamento, immagine originale
- output: immagine scacchiera
- parametri: stima scacchiera =0.50= stimato sulle prime 20 scacchiere

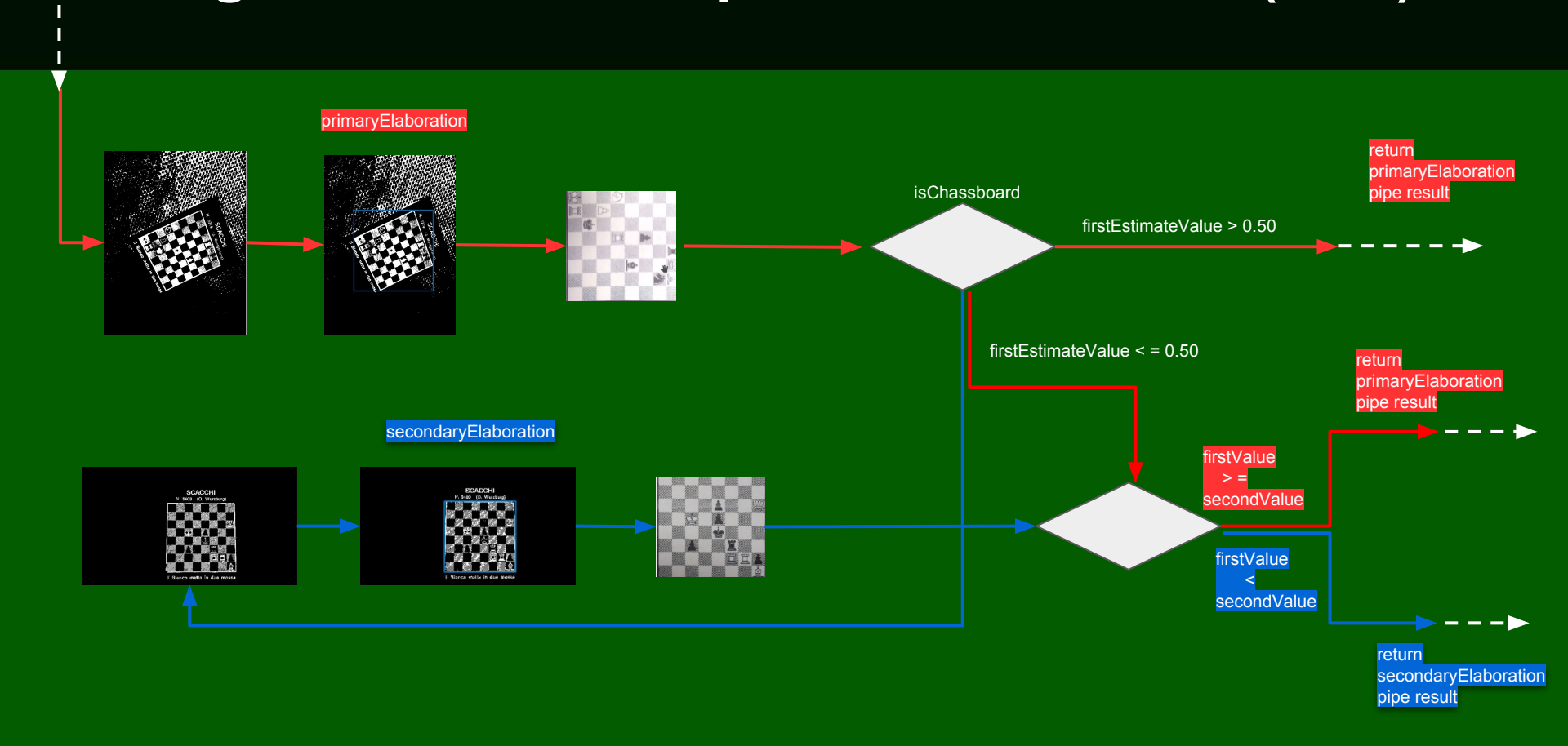
isChessBoard

funzione che si occupa di stimare una percentuale che indica la probabilità che l'immagine passata sia effettivamente una scacchiera.

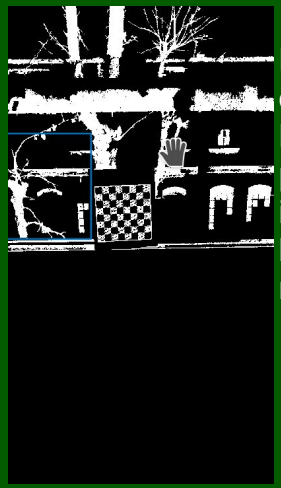
- input: immagine (presunta scacchiera)
- output: valore numerico 0<x<1
- parametri: disco di dimensione 3
- matlab functions: rgb2gray, size, imread, rgb2gray, imbinarize, imopen, imresize, corr2
- test: testIsChessboard

view official doc

Algoritmo di scelta pre-elaborazione - (2.13)

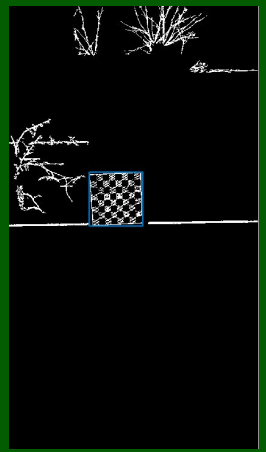


Perchè due diverse pre-elaborazioni? - (2.15)

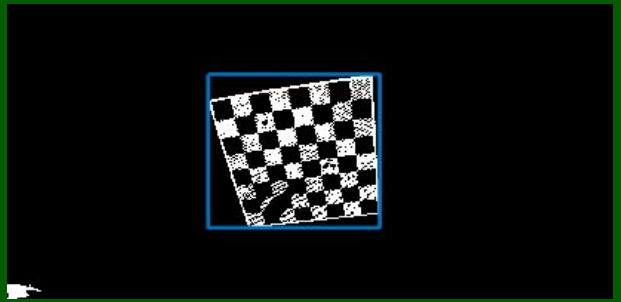


primary

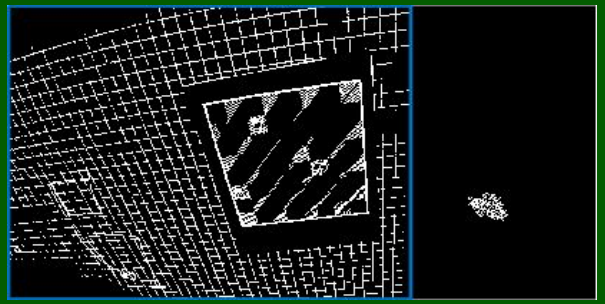
Presenza di un background con oggetti di varie dimensioni.
La prima procedura di pre-elaborazione fallisce, individuando come candidata bounding-box un albero presente nel background.



secondary



primary



secondary

Presenza di un background con una texture.
La seconda procedura di pre elaborazione non è adeguata a ripulire il background , la bounding-box candidata è quella contenente il foglio.

Scelta tra la prima e la seconda elaborazione - (2.16)

	primaryElaboration	secondaryElaboration	caso computazionale
1	0,78	0,84	
2	0,7	0,74	
3	0,7	0,07	
4	0,6	0,6	
5	0,67	0,67	
6	0,57	0,66	
7	0,58	0,55	
8	0,66	0,68	
9	-0,61	-0,6	
10	0,47	0,5	
11	0,43	0,42	
12	0,26	0	
13	0,62	0,5	
14	0,68	0,66	
15	0,56	0,63	
16	0,09	0,41	
17	0,58	0,41	
18	0,6	0,59	
19	0,65	0,04	
20	0,55	-0,06	

Valore negativo?

Il valore negativo si ha quando la scacchiera è ruotata, nel codice il problema è risolto con un valore assoluto, cioè se il valore è fortemente negativo possiamo essere sicuri che sia una scacchiera.

Perchè due diverse pre-elaborazioni? - (2.14)

Le due procedure di pre-elaborazione sono state pensate per agire in modo complementare per coprire il maggior numero di casi possibili.

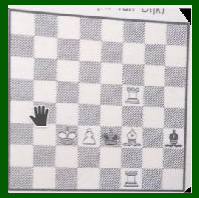
Questo approccio è stato utilizzato perchè alcune foto hanno un background con texture molto sottili (paragonate alle dimensioni della scacchiera), mentre altre mostrano background con sfondi contenenti vari oggetti, di varie dimensioni.

Una procedura pensata per “pulire” uno sfondo con una texture non è sufficiente per pulire lo sfondo da oggetti dimensionalmente simili alla scacchiera, viceversa una procedura pensata per risolvere questo secondo tipo di problema fallisce in caso di background con trame fini.

Vediamo qualche esempio per fare maggior chiarezza.

Immagini non correttamente processate - (2.17)

fail 1



Esempio d'immagine che a causa dello sfondo non appropriato e della eccessiva luminosità non viene correttamente rilevata.

fail 2

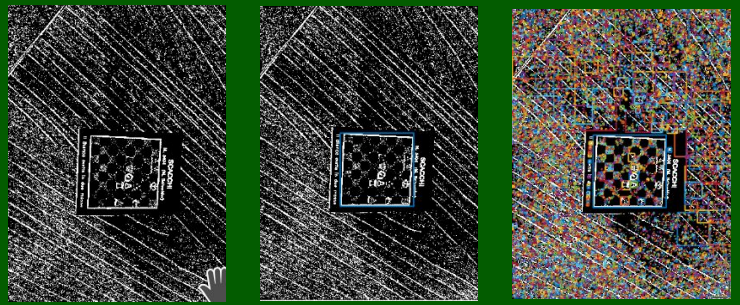
Altro esempio di immagine mal illuminata (troppo luminosa nella parte superiore) che viene per questo motivo rilevata erroneamente.

fail 3

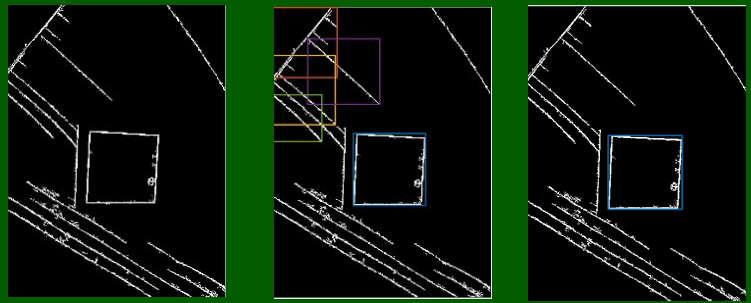
In questo esempio non fallisce la procedura di pre-elaborazione ma l'algoritmo di raddrizzamento, che non è adatto all'eccessiva rotazione della scacchiera.(~45°)

Analisi dei tempi prima parte - (2.18)

Ci siamo resi conto che l'eccessiva durata dell'analisi era dovuta al fatto che venivano generate centinaia o migliaia di bounding-box. In questi casi l'elaborazione poteva durare oltre i 20 secondi, abbiamo quindi aggiunto un algoritmo che eliminasse le componenti connesse di dimensioni inferiori ad un certo numero di pixel, riducendo così drasticamente i tempi elaborazione.



Prima dell'ottimizzazione,
Tempo ~19 s.
Tempo medio parte prima ~4.5 s.



Dopo l'ottimizzazione,
Tempo 1.12 s
Nuovo tempo medio : ~1.10 s.

Miglioramento parte prima - (2.19)

Com’è possibile migliorare la fase di riconoscimento della scacchiera?

È possibile aggiungere altre procedure di pre-elaborazione e utilizzare sempre il paragone dei valori forniti dalla funzione isChessboard per stabilire quale sia il risultato più accurato.

É poi anche possibile migliorare l’algoritmo di raddrizzamento, in modo da riuscire a raddrizzare anche le scacchiere fotografate eccessivamente ruotate.

1. Introduzione
2. Parte prima
   - Individuazione scacchiera
   - Raddrizzamento scacchiera
3. **Parte seconda**
   - **Individuazione pezzi**
   - generazione stringa FEN
4. Analisi dei dati
5. Conclusioni

Riconoscimento dei pezzi: premessa - (3.1)

Per riconoscere i pezzi abbiamo provato varie combinazioni tra differenti set di pezzi da usare come templates e varie funzioni per il confronto fino ad ottenere un risultato soddisfacente.

Il risultato finale è stato ottenuto tramite l'utilizzo di un set di pezzi ritagliati da alcune scacchiere (usati come templates) e la **Correlazione incrociata normalizzata** (in MATLAB viene calcolata mediante la funzione 'normxcorr2').

Lo scopo di questa parte di programma è quello di creare una matrice 8x8 che rappresenti la scacchiera scansionata e che contenga gli indici dei pezzi rilevati.

Attraverso questa matrice nella fase successiva genereremo la stringa FEN.

Riconoscimento dei pezzi: dettaglio - (3.2)

Per riconoscere i pezzi abbiamo per prima cosa ritagliato dalla scacchiera le singole celle. Per farlo abbiamo fatto scorrere un quadrato leggermente più grande di una singola casella per essere sicuri che questa fosse interamente contenuta.

Siccome la dimensione della cella è stata sovradimensionata abbiamo creato un dataset con delle dimensioni minori per agevolare il confronto. Siccome i pezzi ritagliati dalla scacchiera risultano più nitidi abbiamo effettuato un leggerissimo blur su di essi per migliorare i risultati del confronto.

Abbiamo quindi effettuato il confronto tra tutte le celle ed il re nero per 4 volte, ogni volta ruotando quest'ultimo di 90°. Questo per capire in che verso fosse girata la scacchiera.

Riconoscimento dei pezzi: dettaglio - (3.3)

Tramite i risultati ottenuti abbiamo cercato quale fosse la coppia cella/re-nero che generasse il valore di correlazione maggiore in modulo. In questo modo siamo in grado di capire dove si trovi il re e in che verso sia girata la scacchiera.

Una volta capito il verso della scacchiera abbiamo girato nel verso corretto tutte le immagini dei templates e abbiamo completato il confronto rilevando così i pezzi rimanenti.

Contemporaneamente al confronto viene costruita una matrice 8x8 in cui viene salvato l'indice del pezzo riconosciuto nella posizione da cui è stata estratta la casella con cui è stato effettuato il confronto.

Questo tipo di correlazione viene calcolata in MATLAB tramite la funzione 'normxcorr2' a cui si deve passare il template e l'immagine su cui cercarla.

Creazione dataset dei templates - (3.4)

imadjust(imresize(img, [46, 46]));

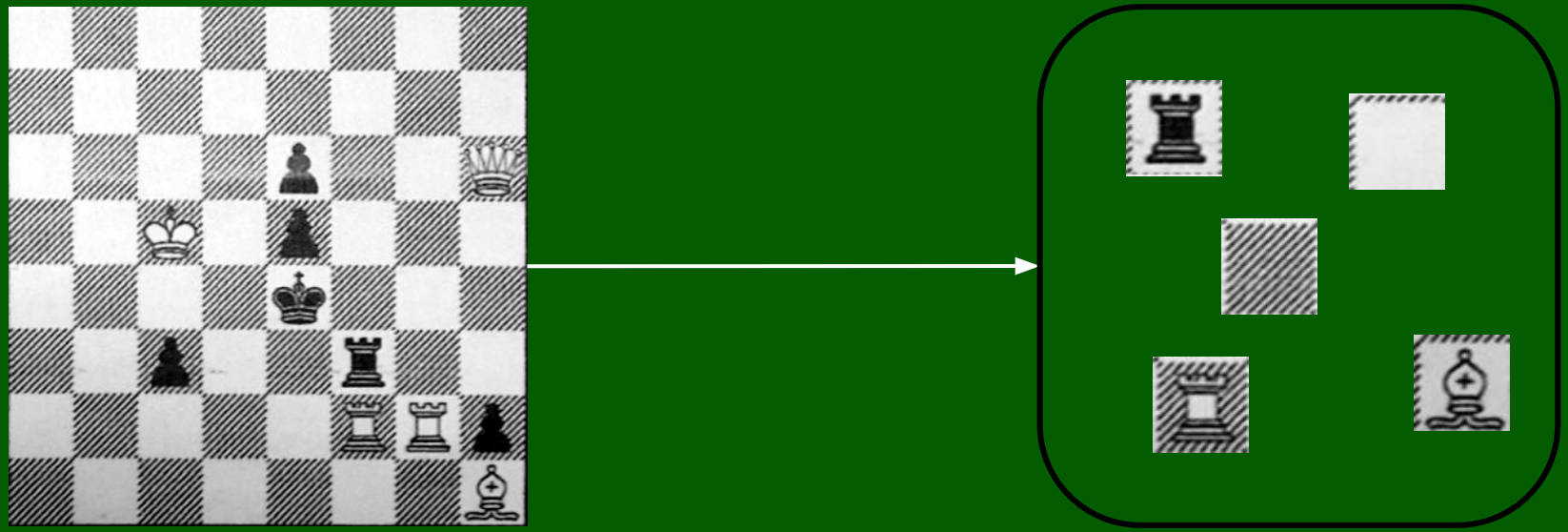
imgaussfilt(img, 1);

Preparazione foto al ritaglio delle celle - (3.5)

chessboard

```
chessboard(k+1:end-k,k+1:end-k, :);
chessboard = imadjust(rgb2gray(chessboard));
```

Ritaglio delle celle - (3.6)



Creazione della matrice con gli indici - (3.7)

celle ritagliate

normxcorr2(templates, cella)

templates
pezzi

32	17	32	17	32	17	32	17
17	32	17	32	17	32	17	32
31	17	32	17	24	17	32	10
17	32	8	32	7	32	17	32
31	17	32	17	1	17	32	17
17	32	7	32	17	21	17	32
31	17	32	17	32	12	27	7
17	32	17	32	17	32	17	28

Costruzione stringa FEN - (3.8)

Per costruire la stringa FEN siamo partiti dalla matrice contenente i vari indici dei pezzi riconosciuti ottenuta durante la fase del riconoscimento di quest'ultimi.

Sapendo a quali pezzi corrispondano i vari indici si può dunque generare la stringa FEN sostituendo ai vari indici i caratteri corrispondenti.

Questa stringa non è però ancora completa perché le celle vuote adiacenti tra loro sono da aggregare e sostituire con la cifra corrispondente al numero di celle vuote adiacenti.

Una volta compiuta questa sostituzione si ha una stringa FEN completa per quanto riguarda le posizioni dei vari pezzi sulla scacchiera. Manca però la parte che riguarda lo stato della partita all'ultima mossa.

Nel nostro caso questa parte di stringa è una costante e quindi è bastato concatenare alla stringa che rappresenta le varie posizioni dei pezzi i caratteri "_- _1_0" (dove '_' rappresenta uno spazio) per avere una stringa FEN completa in ogni parte.

Controllo della stringa FEN generata - (3.9)

Una volta generata la stringa FEN abbiamo quindi effettuato il confronto tra la stringa FEN generata e quella inserita nel file di groundtruth corrispondente all'immagine analizzata.

Qui si aprono due strade:

1. Se la stringa risulta corretta la procedura di controllo viene terminata e ritorna la stringa FEN.
2. Se la stringa risulta errata la procedura verifica quanti sono i caratteri giusti, per calcolare la percentuale di correttezza della stringa FEN generata, e genera una stringa contenente la lista di pezzi rilevati erroneamente e i pezzi reali. Quest'ultima stringa viene dunque ritornata insieme alla stringa FEN generata.

Passaggi per creare la stringa FEN - (3.10)

Matrice generata dalla funzione fenGenerator

32	17	32	17	32	17	32	17
17	32	17	32	17	32	17	32
31	17	32	17	24	17	32	10
17	32	8	32	7	32	17	32
31	17	32	17	1	17	32	17
17	32	7	32	17	21	17	32
31	17	32	17	32	12	27	7
17	32	17	32	17	32	17	28

Stringa generata dalla funzione fefenString

aaaaaaaaaaaaaaaaaaaapaaQaaKapaaaaaaakaa
aaapaaraaaaaaaRRpaaaaaaaB

Stringa generata dalla funzione fenStringApp

8/8/4p2Q/2K1p3/4k3/2p2r2/5RRp/7B

Stringa generata dalla funzione fenStringApp

8/8/4p2Q/2K1p3/4k3/2p2r2/5RRp/7B - 0 1

Passaggi per creare la stringa FEN - (3.11)

Rilevazione corretta

immagine 1:
corretta al = 100%
FEN = 8/8/4p2Q/2K1p3/4k3/2p2r2/5RRp/7B - 0 1
tempo impiegato = 4.63 seconds.

Rilevazione errata

immagine 11:
corretta al = 98.44%
FEN = 2bb1K2/4p3/5k1n/3qNP2/5R1P/8/8/8 - 0 1
pezzi sbagliati= q -> Q
tempo impiegato = 4.63 seconds.

In questa immagine la regina bianca viene riconosciuta come regina nera.
Questo errore è legato al fatto che l'immagine risulta molto sfocata e le due regine risultano quindi molto simili tra loro (anche ad occhio si potrebbero scambiare).

Passaggi per creare la stringa FEN - (3.12)

In questa immagine vengono commessi numerosi errori (6) perché la scacchiera risulta ancora leggermente distorta e il metodo di estrazione non riesce a catturare adeguatamente le varie caselle che risultano leggermente tagliate

In questa immagine vengono commessi due errori (blanck -> B, blanck -> R). Quesiti sono dovuti al fatto che la scacchiera risulta particolarmente sfocata e dunque il riconoscimento risulta più difficoltoso.

In questa immagine l'alfiere bianco in alto a sinistra viene scambiato per una cella bianca. In questo caso la leggera ombra e il fatto che l'alfiere sia già di norma abbastanza "simile" alla casella nera infastidisce l'algoritmo.

1. Introduzione
2. Parte prima
   - Individuazione scacchiera
   - Raddrizzamento scacchiera
3. Parte seconda
   - individuazione pezzi
   - generazione stringa FEN
4. **Analisi dei dati**
5. Conclusioni

Analisi dei risultati - (4.1)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Parte 1																				
tempo	0,89	0,9	0,92	0,88	0,91	0,88	0,91	0,92	0,74	0,9	1,15	1,77	0,9	0,88	0,85	1,32	1,07	1,08	1,08	1,02
Parte 2																				
tempo	5	4,49	4,44	4,67	4,49	4,65	4,46	4,61	4,65	4,52	4,51	4,63	4,48	4,49	4,56	4,58	4,49	4,5	4,46	4,52
tempo tot	5,9	5,39	5,36	5,55	5,39	5,53	5,38	5,54	5,39	5,42	5,67	6,39	5,38	5,37	5,41	5,9	5,57	5,58	5,54	5,54
% riconoscimento	100	100	100	100	100	100	100	100	100	96,88	98,44	95,31	100	100	100	68,75	76,56	100	100	90,63

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1,09	1,41	1,09	1,34	1,17	1,21	1,19	1,53	1,18	1,14	1,2	1,19	0,84	0,81	1,2	1,18	1,14	1,77	3,39	0,88
5,14	4,6	4,48	4,79	4,67	4,88	4,9	5,34	5,52	5,81	4,91	4,48	4,51	4,47	4,58	4,59	4,57	4,7	4,67	4,65
6,23	6,01	5,57	6,13	5,84	6,09	6,09	6,86	6,7	6,94	6,11	5,67	5,35	5,28	5,78	5,77	5,71	6,47	8,05	5,53
98,44	100	100	98,44	100	100	98,44	93,75	100	100	100	100	100	100	100	78,13	100	100	100	96,88

41	42	43	44	45	46	47	48
0,89	0,83	0,88	1,25	1,2	0,86	0,84	0,86
5,15	4,58	5,1	4,96	4,53	4,54	4,45	4,57
6,04	5,41	5,98	6,21	5,73	5,4	5,29	5,43
100	100	100	100	100	100	96,88	100

immagini aggiunte da noi

49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70
0,64	1,17	1,13	0,87	0,86	0,86	0,89	1,27	0,87	1,23	0,91	0,91	0,94	0,86	0,93	0,87	0,89	1,17	0,87	0,87	0,91	0,88
5,24	5,18	5,49	5,53	5,32	4,88	4,68	4,53	4,53	4,58	4,55	4,46	5,36	4,79	4,41	4,46	4,45	4,76	4,53	4,51	4,52	4,47
5,88	6,34	6,61	6,4	6,17	5,74	5,58	5,8	5,4	5,81	5,46	5,38	6,3	5,65	5,34	5,34	5,33	5,92	5,4	5,38	5,42	5,35
100	100	98,44	100	100	100	100	53,13	100	85,94	100	100	100	100	100	96,88	100	100	100	100	84,38	100

tempo medio di esecuzione: 5,78 s
% scacchiere date dal prof : 72,92%
% scacchiere tot: 75,71%

Matrice di confusione (foto consegna) - (4.2)

predetti

reali

[]	'blank'	'K'	'k'	'Q'	'q'	'R'	'r'	'B'	'b'	'N'	'n'	'P'	'p'
'blank'	0.9925	0	0.0012	0	0	0	0.0043	0	3.9216e-...	0	0	0	0.0016
'K'	0.0625	0.9375	0	0	0	0	0	0	0	0	0	0	0
'k'	0.0417	0	0.9375	0	0.0208	0	0	0	0	0	0	0	0
'Q'	0.1190	0	0	0.8571	0.0238	0	0	0	0	0	0	0	0
'q'	0	0	0	0	0	0	0	0	0	0	0	0	0
'R'	0.1212	0	0	0	0	0.8485	0.0303	0	0	0	0	0	0
'r'	0	0	0	0	0	0	0.9583	0	0	0	0	0	0.0417
'B'	0.0833	0	0	0	0	0	0.0208	0.8958	0	0	0	0	0
'b'	0.0556	0	0	0	0	0	0.0556	0	0.8889	0	0	0	0
'N'	0.1111	0	0	0	0	0	0	0	0	0.8889	0	0	0
'n'	0.0476	0	0	0	0	0	0	0	0	0	0.9524	0	0
'P'	0.1667	0	0	0	0	0	0	0	0	0	0	0.8333	0
'p'	0.0500	0	0	0	0	0	0.0167	0	0	0	0	0	0.9333

Analisi della Matrice di confusione - (4.3)

Da questi risultati si può evincere come le celle vuote siano quelle che generano più falsi positivi.

Questo risultato è da attribuire soprattutto ai pezzi bianchi su celle con sfondo nero che, in immagini un po' sfocate, vengono viste come vuote anche se contengono dei pezzi (questo succede prevalentemente con i pezzi bianchi come regina, torre ed alfiere).

Si può anche vedere come nel dataset di immagini che ci è stato fornito non compaia mai la regina nera (per testare anche la sua corretta rilevazione abbiamo aggiunto delle foto in cui comparisse).

Questi dati potrebbero essere leggermente alterati dal fatto che non siamo riusciti a trovare un metodo ben funzionante che fermasse il procedimento nel caso la scacchiera non venisse correttamente trovata e ritagliata.

Matrice di confusione foto consegna + nostre - (4.4)

predetti

reali

[]	'blank'	'K'	'k'	'Q'	'q'	'R'	'r'	'B'	'b'	'N'	'n'	'P'	'p'
blank'	0.9887	0	0.0016	0	0	0	0.0062	0	2.6976e-...	0	5.3952e-...	0	0.0027
K'	0.0857	0.9143	0	0	0	0	0	0	0	0	0	0	0
k'	0.0571	0	0.9143	0	0.0143	0	0.0143	0	0	0	0	0	0
Q'	0.1270	0	0	0.8413	0.0159	0	0.0159	0	0	0	0	0	0
q'	0	0	0	0	1	0	0	0	0	0	0	0	0
R'	0.1209	0	0	0	0	0.8571	0.0220	0	0	0	0	0	0
r'	0	0	0	0	0	0	0.9722	0	0	0	0	0	0.0278
B'	0.1096	0	0	0	0	0	0.0137	0.8767	0	0	0	0	0
b'	0.0385	0	0	0	0	0	0.0385	0	0.9231	0	0	0	0
N'	0.1558	0	0	0	0	0	0	0	0	0.8442	0	0	0
n'	0.0678	0	0	0	0	0	0	0	0	0	0.9153	0	0.0169
P'	0.1579	0	0	0	0	0	0	0	0	0	0	0.8421	0
p'	0.0891	0	0	0	0.0099	0	0.0198	0	0	0	0.0099	0	0.8713

In questa matrice si può vedere come anche la regina nera venga correttamente riconosciuta e come i dati non si discostino troppo da quelli precedenti. Dunque per l'analisi di questi dati vale quanto detto nella slide precedente.

1. Introduzione
2. Parte prima
   - Individuazione scacchiera
   - Raddrizzamento scacchiera
3. Parte seconda
   - individuazione pezzi
   - generazione stringa FEN
4. Analisi dei dati
5. **Conclusioni**

Dove l’algoritmo fallisce? - (5.1)

La fase di pre-elaborazione, con le due variazioni, copre molti casi, ma ancora fallisce nel caso di luce molto intensa che si riflette sulla carta, o anche in caso di ombre molto forti che corrompono la scacchiera dopo la fase di pre-elaborazione.

Il raddrizzamento fallisce in caso di scacchiera ruotata con un angolo vicino ai 45° (come è possibile vedere dagli errori della prima parte).

Il nostro algoritmo fallisce particolarmente nelle immagini sfocate e nelle caselle dove c’è una particolare variazione delle ombre.

Sbaglia anche il riconoscimento delle celle quando i lati della scacchiera risultano curvi. Questo perché il metodo di estrazione celle avviene tramite spostamento rigido di un quadrato.

Miglioramenti possibili? - (5.2)

- Aggiungere altri metodi di pre-elaborazione, e scegliere quale applicare non più per tentativi, ma facendo uno studio delle caratteristiche dell'immagine (es. analisi dell'istogramma).
- considerare anche l'eventualità in cui la scacchiera sia ruotata a 45°.
- Fermare l'algoritmo in caso di rilevazione di più pezzi identici sulla stessa scacchiera quando questa soglia supera il numero non consentito.
- Migliorare il riconoscimento dei pezzi in zone con differente luminosità tramite una migliore equalizzazione dell'immagine.