Riconoscimento di una griglia di gioco OpenCv

Problema
L' obbiettivo di questo progetto consiste nel riconoscimento di una griglia di gioco stile quella del sudoku, dove le celle sono di colore uniforme e non a scacchi. L'applicativo deve successivamente estrarne dei dati come per esempio la cifra o la pedina presente in ogni cella per poterli memorizzare e/o elaborare.

Obbiettivo del programma
Il progetto è stato pensato come interfaccia per un altro programma che necessita di sapere il dato contenuto in ogni cella della griglia. Si potrebbe quindi creare un programma che gioca per esempio a sudoku e che sfida l'umano.

Come è stato affrontato il problema?
Il programma si divide in più parti, delle quali le principali sono: *individuare la griglia*, *"raddrizzare"* quest'ultima, *individuare righe e colonne*, *estrarre le celle*, *estrarre il dato* contenuto nelle singole celle.
Il codice dell'applicazione si divide in due grandi parti, due metodi che risolvono il problema, che poi confluiscono andando a intrecciare i dati ottenuti per avere maggiori probabilità di successo di risoluzione del problema. L'estrazione dei dati avviene come fase finale, dopo aver ottenuto le singole celle della griglia.
Metodo 1
Prevede le seguenti fasi:

- Individuazione della griglia attraverso un algoritmo che trova l'oggetto maggiore presente nell'immagine; si assume che l'immagine contenga la griglia come oggetto principale.
- Isolamento di quest'ultima con approssimato raddrizzamento del verso (per esempio se questa fosse obliqua viene posizionata correttamente).
- Individuazione di tutte le linee presenti
- Scelta delle linee corrette e unione delle rette che descrivono la stessa linea
- Intersezione delle rette

Metodo 2
Prevede le seguenti fasi

- Individuazione della griglia e isolamento di quest'ultima dal resto della foto
- Raddrizzamento approssimato del verso (come accade per il metodo 1)
- Elaborazione dell'immagine in modo da eliminare il più possibile oggetti verticali o orizzontali non aventi le caratteristiche di una linea
- Individuazione delle linee
- Eliminazione di quelle linee che non rispettano la distanza tra colonne
- Intersezione delle rette

Una volta trovate le intersezioni delle rette in entrambi gli algoritmi, bisogna combinare i dati in modo da sopperire alle debolezze degli algoritmi, assicurandosi un'accuratezza maggiore del programma. *
Trovate quindi le celle derivanti dai punti, si deve isolare e "raddrizzare" ogni singola cella in modo da facilitare la rilevazione del dato contenuto. *

* le parti contrassegnate non sono ancora state implementate, quindi ci si può riferire ad esse esclusivamente in modo teorico e deduttivo

Ecco che il dato può essere essenzialmente di due tipi: una pedina, si assume per semplicità che sia tonda, o un numero o una lettera. Si dovrà inizialmente definire il tipo di dato del gioco oppure creare un algoritmo che tenti da solo di capire ciò. *

Problemi riscontrati

I problemi che ho incontrato sono numerosi dato che non è facile entrare nel mondo della computer vision con poca o nulla esperienza sulla visione e più in generale su immagini e video. Alcuni dei problemi riscontrati sono sicuro che potevano essere risolti brillantemente da chi avesse avuto maggior esperienza sull'argomento ma anche so OpenCv… per esempio, un problema insulso quale quello di riempire un contorno, che la libreria permette di risolvere ponendo *thickness=CV_FILLED* non sono riuscito a risolverlo, sicuramente perché mi sfugge qualcosa su come lavora quella funzione.
Parlando dell'algoritmo i principali problemi riscontrati sono i seguenti:

- un buon threshold che non cancellasse parte dei dati importanti
- una corretta individuazione delle linee presenti nell'immagine attraverso la funzione *HoughLines*
- individuazione delle rette che non centrano e di quelle che descrivono la stessa linea della griglia
- la scelta della retta che descrive meglio una linea tra tutte quelle che la descrivono (vedi quando la linea è deformata/storta)
- l'eliminazione, attraverso una trasformazione morfologica, degli oggetti che non sono linee orizzontali o verticali
- la rilevazione tra quelle "piccole" linee rimaste dalla trasformazione morphologica di quali non fanno parte della struttura della griglia
- dato che non sono riuscito a riempire i contorni individuati per le rette, visto anche il fatto che alcune erano storte e non riuscivo a trovare un punto interno al contorno (avevo pensato al punto centrale del rettangolo che approssima il contorno ma essendo i contorni molto stretti capitava che fosse fuori dal contorno), mi è toccato intersecare i contorni che descrivono le linee generando spesso 4 punti. Poi ho dovuto unire questi 4 punti e trovare quello medio (si noti che esisteva una lista contenente tutti i punti disordinati)
- presumo poi ci saranno ulteriori problemi nell'intrecciare i dati e trovare le celle e nel rilevare i numeri *

Requisiti per un corretto funzionamento

- OS: Windows, Linux, MacOs, …
- OpenCv 2.4.x (testato con la versione 2.4.13)
- Python2.x
- webcam (opzionale)

Funzioni di OpenCv utilizzate

- floodFill()
- GaussianBlur(), adaptiveThreshold()
- findContours(), contourArea(), drawContours(), minAreaRect(), moments(), cv.BoxPoints()
- HoughLines()
- bitwise_and()
- getStructuringElement(), morphologyEx(), dilate(), erode()
- circle(), line()
- getPerspectiveTransform(), warpPerspective()
- imread(), imwrite(), imshow()

* le parti contrassegnate non sono ancora state implementate, quindi ci si può riferire ad esse esclusivamente in modo teorico e deduttivo

- waitKey()
- destroyAllWindows()

Esempi
Nella cartella risultati si possono vedere alcune immagini che mostrano i progressi fatti.

Interfaccia
Si può comunque avviare il programma digitando da linea di comando:

```
python sudoku.py
```

che esegue l'immagine di default *sudoku.jpg* presente nella cartella *images*.

python sudoku.py img	esegue il programma elaborando l'immagine passata
python sudoku.py -a	mostra tutti i risultati (immagini)
python sudoku.py -m1	esegue solo il metodo 1

Durante l'esecuzione del programma con il tasto S si salvano i risultati ottenuti nella cartella esempi, mentre con ESC si esce dal programma.

* le parti contrassegnate non sono ancora state implementate, quindi ci si può riferire ad esse esclusivamente in modo teorico e deduttivo