ELABORAZIONE DELLE IMMAGINI

PROGETTO ANNO 2020/2021 PUZZLE TETRIS

MANGANARO FRANCESCO – 845087

POZZI MICHELE – 845727

GARGARO DAVID – 845738

PRETELL KEVIN - 816725

Descrizione Obiettivi e Analisi dei Dati

L'obiettivo è quello di creare un algoritmo in grado di riconoscere e collocare correttamente dei tetramini all'interno di uno schema

Input:

Un'immagine di **scena** contenente tetramini sparsi su una superficie piana
Un'immagine di **schema** che mostra uno schema dove posizionare i tetramini

Output:

Un'immagine che mostra come sono stati **collocati** i tetramini rilevati nella **scena all'interno** dello **schema**

Immagine Scena

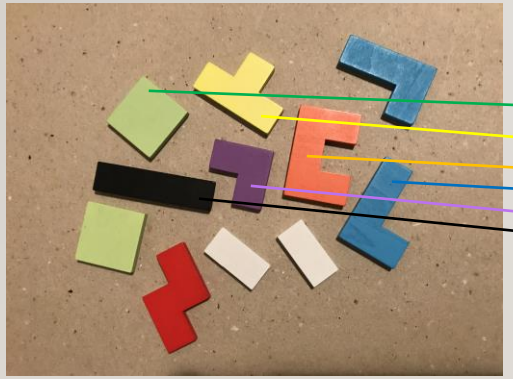
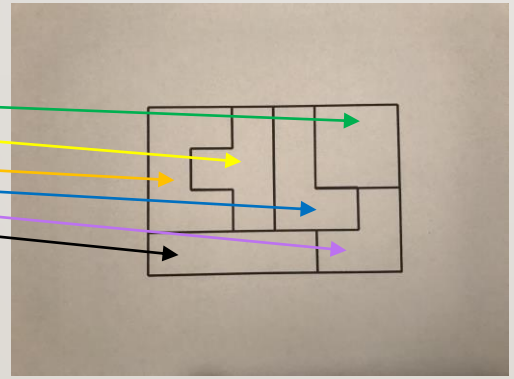
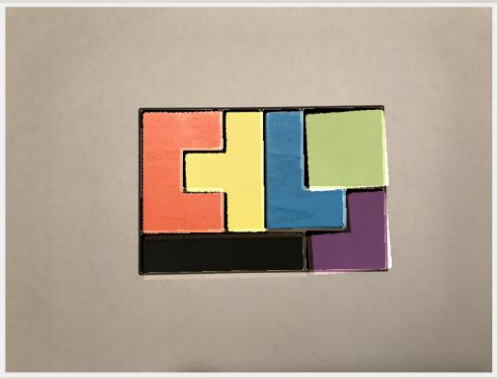


Immagine Schema

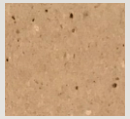


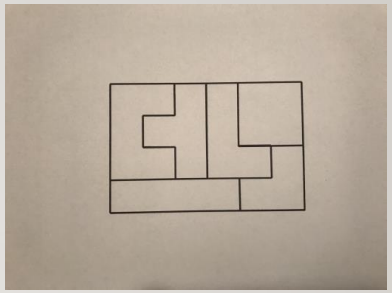
Svolgimento Algoritmo

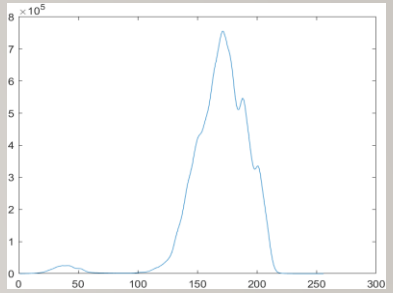
Immagine di Output



Analizzando le immagini fornite abbiamo notato che:

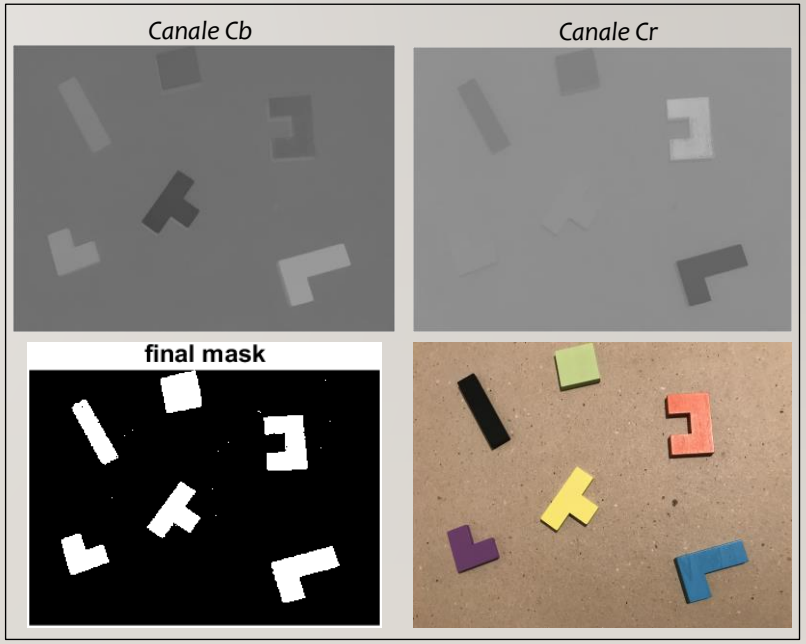
- Le immagini di **training** e **scena** presentano **lo stesso sfondo** 





- Gli istogrammi delle immagini di schema non presentano dei picchi ben definiti, quindi **escludiamo** una binarizzazione a **soglia costante** o una binarizzazione utilizzando il metodo di **Otsu.**

- Le immagini di scena **non** possono essere binarizzate usando una binarizzazione con **soglia costante,** anche considerando diversi canali colore.
  Sarebbe necessario utilizzare troppe soglie e arriveremmo a super modellare il training set



- I **colori** dei **tetramini** nelle immagini di **training** e di **scena** sono **simili.**
  Campionando il colore dei tetramini delle immagini di training possiamo usare un classificatore per riconoscere i tetramini di scena.

Assunzioni

- Le **foto** devono essere **dall'alto** *(perpendicolari alla scena)*, con **luce uniforme** *(come l'immagine di training)* e **non** devono presentare **rumore**
- Lo **sfondo** delle immagini di **scena** deve avere **texture uguale** allo sfondo delle immagini di **training**
- Lo **schema** *(nelle immagini di schema)* deve essere al **centro** dell'immagine
- La **distanza** di **scatto** delle immagini di **scena** deve essere la **stessa** rispetto a quella delle immagini di **training**

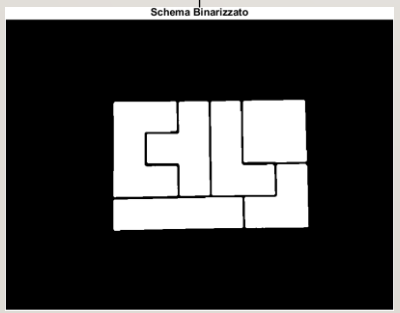
- **Non** ci devono essere **oggetti estranei** in scena e schema
- Le immagini di schema e scena devono essere della **stessa dimensione** delle immagini di training
- Le immagini di **scena** devono avere lo **stesso bilanciamento** dei **colori** della immagini di training
- I tetramini di **scena** possono essere **traslati** e **ruotati** per essere collocati nello **schema** *(non vengono specchiati)*

- Il colore dei tetramini deve essere **omogeneo** e **nell'insieme**: {Giallo, Arancione, Rosso, Nero, Bianco, Blu, Verde e Viola} *(colore dei tetramini presenti nell'immagini di training)*

Immagini in Input → **Addestramento Classificatore** → **Etichetta Schema** → **Etichetta Scena** → **Piazza Tetramino** → *Immagine in Output*

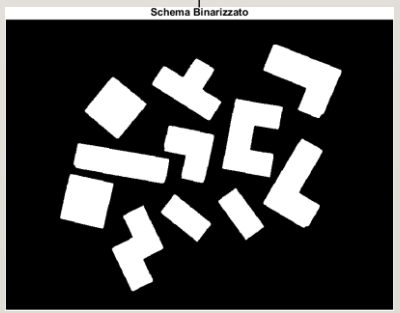
Addestramento di un classificatore KNN con 3 vicini per classificare nelle immagini di scena gli oggetti che sono sfondo oppure sono tetramini

Labelizzazione dell'immagine di schema



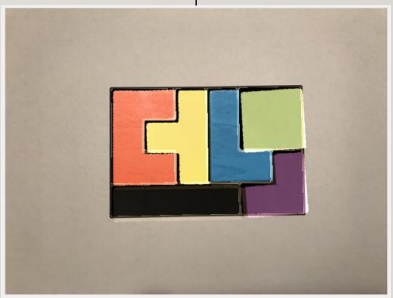
Labelizzazione dell'immagine di scena

Utilizzando il classificatore precedentemente addestrato



Calcolo correlazione tra tetramini di schema e scena

Collocamento dei tetramini di scena nelle regioni dello schema



Addestramento Classificatore

Utilizziamo un classificatore KNN con 3 vicini per **etichettare** le immagini di **scena**.
Abbiamo creato due immagini prelevando dei campioni dalle immagini di training.

Necessitiamo il riconoscimento di due classi quindi creiamo due immagini differenti che rappresentano le **due classi** che devono essere riconosciute

- Immagine *sfondo.jpg*
- Immagine *tetramini.jpg*

La prima contiene parti di sfondo.
La seconda contiene parti di tetramini nelle varie colorazioni e sfumature che ci sono.

sfondo.jpg

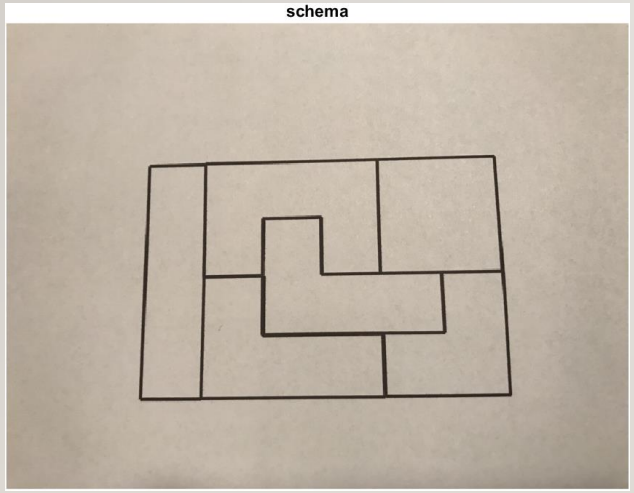
teramini.jpg

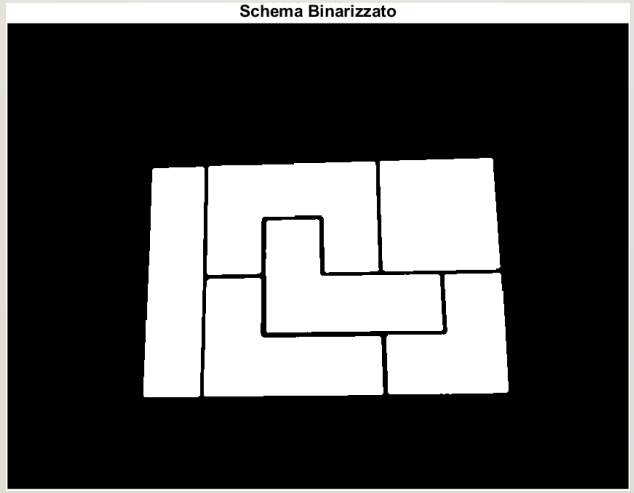
Etichetta Schema

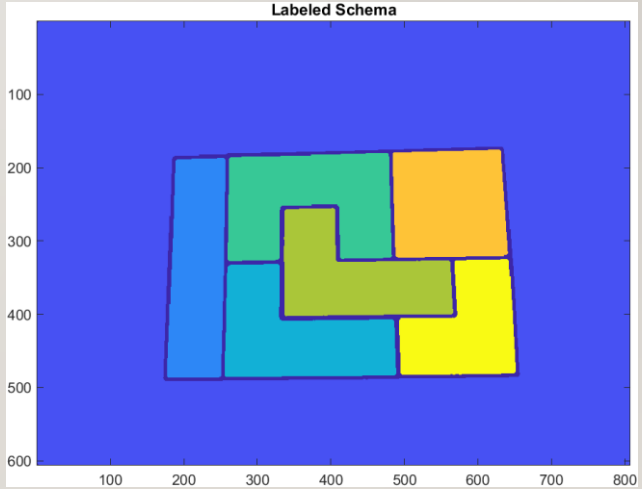
Utilizziamo il metodo di **Sauvola** considerando che nell'immagine di schema lo sfondo ha una illuminazione non omogenea.
Applichiamo Sauvola sull'immagine di schema (trasformata a livello di grigio) con un filtro 31x31.

Effettuiamo anche una pulizia della maschera ottenuta con un **filtro mediano** di dimensione 5x5.

Infine con un **labeling** delle **componenti connesse** ottengo uno schema dove ogni regione trovata è etichettata con un'etichetta differente.







Etichetta Scena

Applichiamo un **filtro** di **media** di dimensione 3x3 all'immagine per amalgamare i colori dei tetramini e dello sfondo.
Utilizzando il classificatore KNN precedentemente addestrato siamo in grado di **riconoscere** e **etichettare** i **tetramini**.

Effettuiamo un'operazione di **morfologia matematica** *(OPEN usando un operatore quadrato di dimensione 3x3)* in modo da staccare l'ombra da ogni tetramino.

Successivamente eliminiamo dalla maschera trovata tutte le regioni che hanno area inferiore ad una determinata soglia *(2000 px)*
(La soglia è stata calcolata a mano considerando che l'immagine è stata ridimensionata con una scala di 0.2 e ha una dimensione di 605x807 Considerando 2000 pixel di area posso essere sicuro di eliminare solamente le piccole regioni di ombra)

(eliminano ombre dei tetramini e eventuali errori di binarizzazione)

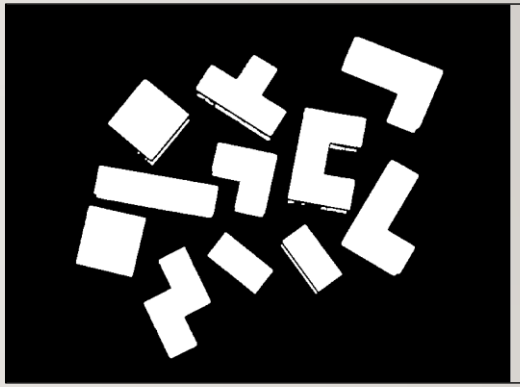
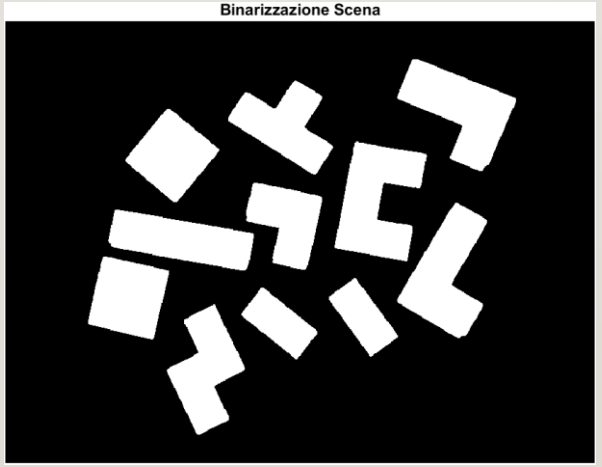
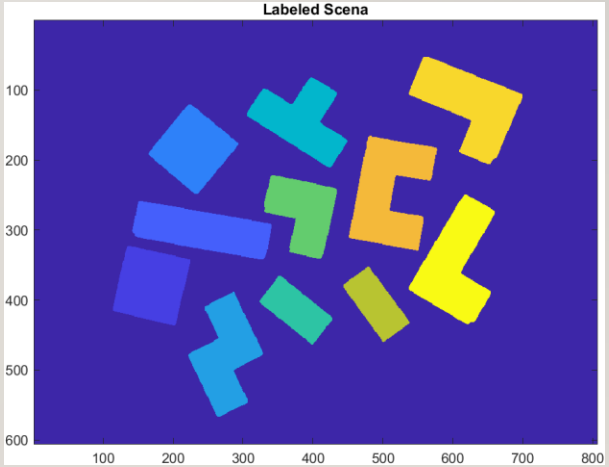


Immagine binarizzata dopo l'applicazione dell'operatore morfologico.





Piazza Tetramino

Considero separatamente ogni tetramino di schema
Per ognuno ne calcolo la **correlazione** con **tutti** i **tetramini** di **scena** trovati

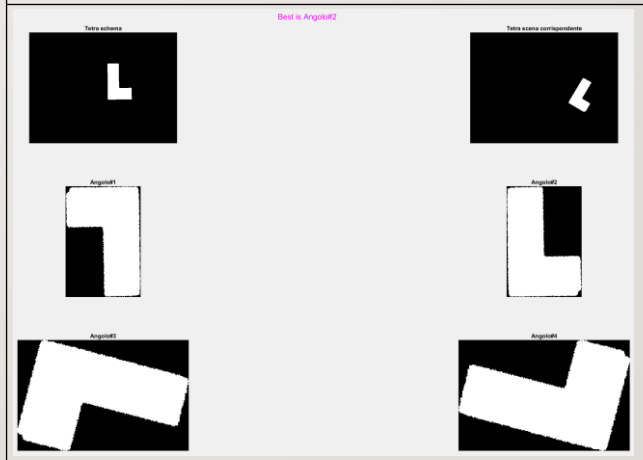
Per effettuare la correlazione necessito di **ruotare** il tetramino di scena affinché combaci con il tetramino di schema.

Effettuo questa operazione considerando 5 angolazioni differenti e calcolando per ognuna la correlazione.

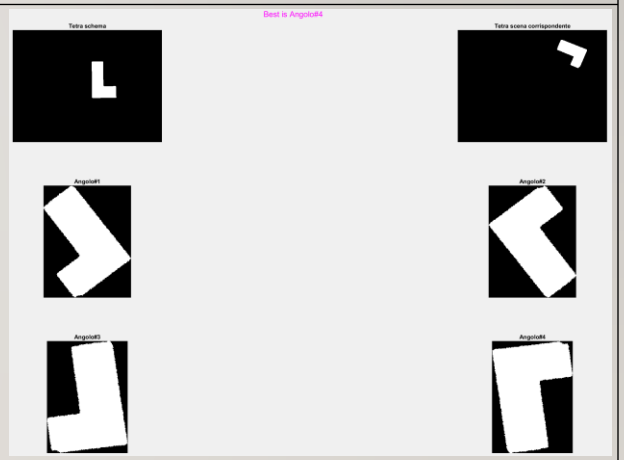
Al termine del processo **salvo** la **correlazione maggiore** e **l'angolo** che l'ha prodotta.

Operazioni effettuate dal metodo calcola_angolo.m

Il tetramino in alto a sinistra è di schema
Il tetramino il alto a destro è di scena



Correlazione maggiore ottenuta con l'angolo numero 2.



Correlazione maggiore ottenuta con l'angolo numero 4.

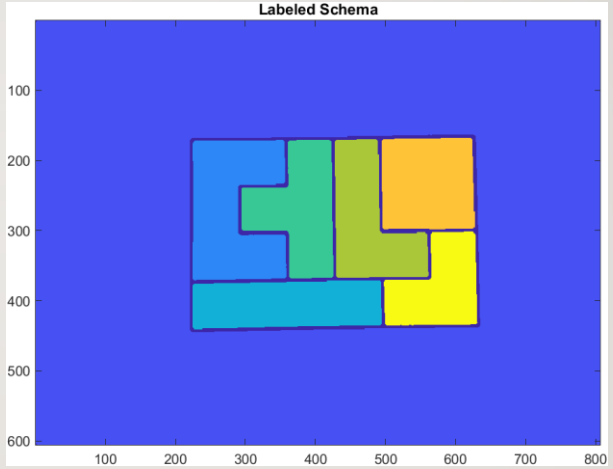
Tra i due tetramini, l'algoritmo sceglie quello di sinistra (avendo correlazione maggiore)

Al termine del confronto tra i tetramini ottengo una **matrice** dove ogni elemento corrisponde alla **massima correlazione** trovata tra tetramino di schema e scena.

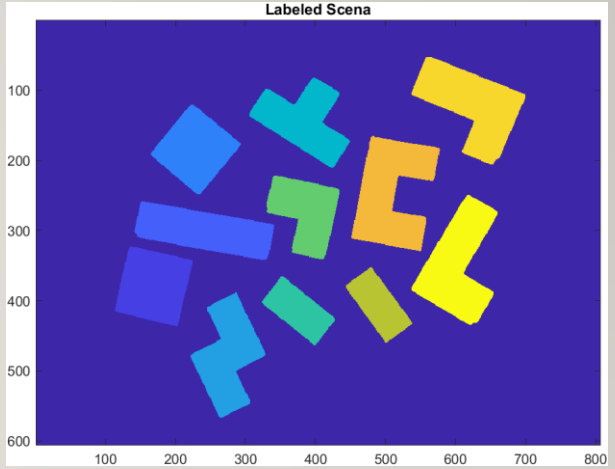
Con N tetramini di schema e M tetramini di scena ottengo una matrice NxM.

Estraggo iterativamente dalla matrice i primi N massimi globali e li confronto con una soglia (**0.825**)
(La soglia è stata calcolata a mano controllando singolarmente ogni coppia di tetramini cercando un valore ottimale che non causasse falsi positivi o falsi negativi)

Se la **correlazione** è **maggiore** di tale **soglia** allora considero la coppia tetramino (scena, schema) come valida e la colloco all'interno dello schema, eliminando la coppia dalla matrice per evitare che sia riconsiderata.



Ho rilevato 6 tetramini di schema



Ho rilevato 11 tetramini di scena

Viene prodotta una matrice 6x11 e vengono estratti i 6 valori maggiori (a patto che siano maggiori della soglia)

E le relative coppie migliori.

Matrice della correlazione

```
bw1xbw2(:,:,1) =

    0.6497    0.7010    0.6310    0.5453    0.6684    0.8005    0.6597    0.7977    0.9687    0.7312    0.6525
    0.4619    0.9049    0.4387    0.2448    0.6097    0.6589    0.4428    0.6798    0.6244    0.4239    0.3790
    0.6709    0.3358    0.6666    0.5725    0.9431    0.5385    0.7898    0.6262    0.5825    0.6494    0.6769
    0.5406    0.6963    0.5524    0.5863    0.6196    0.7342    0.5941    0.7447    0.6829    0.6249    0.9394
    0.9334    0.5176    0.9602    0.6547    0.7249    0.6938    0.7949    0.6773    0.6856    0.6863    0.6213
    0.7372    0.6084    0.7465    0.7169    0.7864    0.7746    0.9331    0.7683    0.7524    0.8065    0.7897
```

Collocamento Tetramino nella Scena

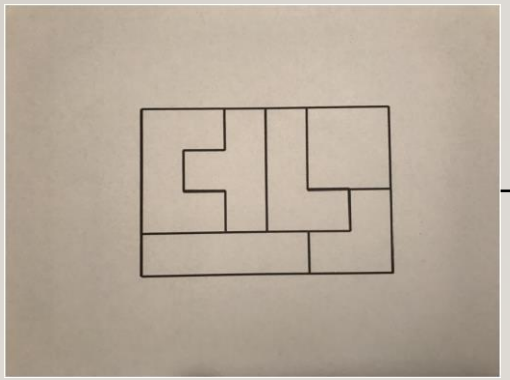
```
bw1xbw2(:,:,3) =

   11.9542  -79.7533   37.1243 -174.1554  -57.9454  129.2697  110.4151  143.9523   10.8483  120.9091   12.1106
  102.8338  191.1263  128.0039   96.7242  212.9342   40.1493  109.7420  234.8318  -78.2721  120.2360  -73.8681
   12.9840  -62.2405 -141.8458         0  123.0845         0  151.9961         0         0  131.6652         0
  -72.2180  -63.7341  -46.6908 -154.7399  -50.8030  132.7456  109.5678  -32.3681         0  120.0618   31.5261
  -72.2180   -9.1257  -46.6908  -52.5360         0         0  109.5678         0  -76.8298         0  105.9577
  -71.7776   55.9546  -46.2504  141.5524  257.7624  264.9775  286.6740  279.6601  146.5561  266.3431  147.8184
```

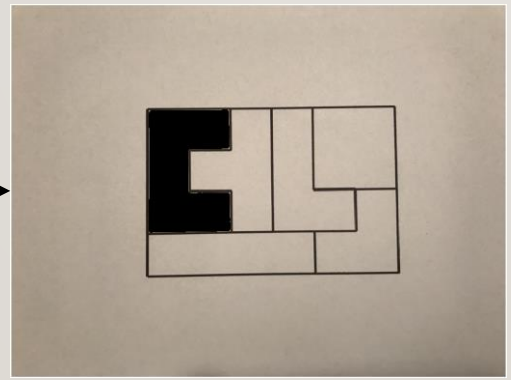
Matrice degli angoli ottimali

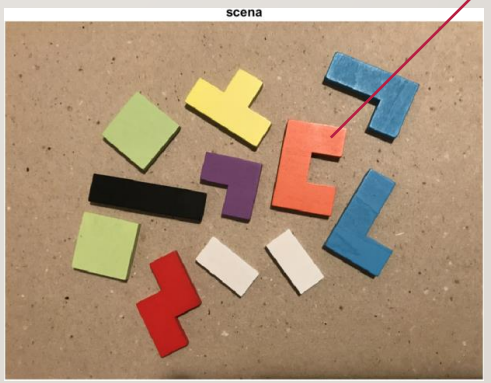
Considerando le coppie trovate utilizzo:

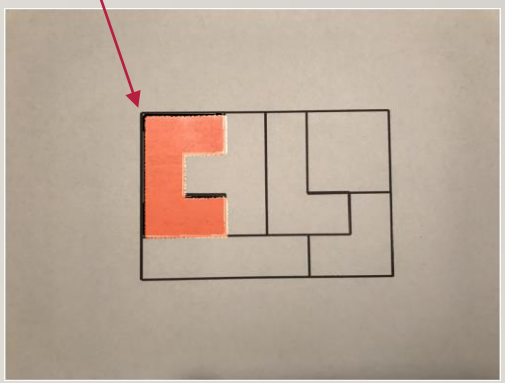
- La maschera del tetramino di scena per **ritagliare** il **tetramino reale** dall'immagine RGB
  - *Che poi verrà ruotato usando l'angolo ottimale calcolato precedentemente*
- La maschera del tetramino di schema per trovare la **posizione esatta** nello **schema** dove applicare il tetramino
  - *Metto a 0 tutta le regione che rappresenta il tetramino per evitare sovrapposizioni*
- Piazzo il tetramino reale nello schema considerando le coordinate trovate nel passaggio precedente

Itero il procedimento finché non ho finito le coppie (schema, scena) da collocare



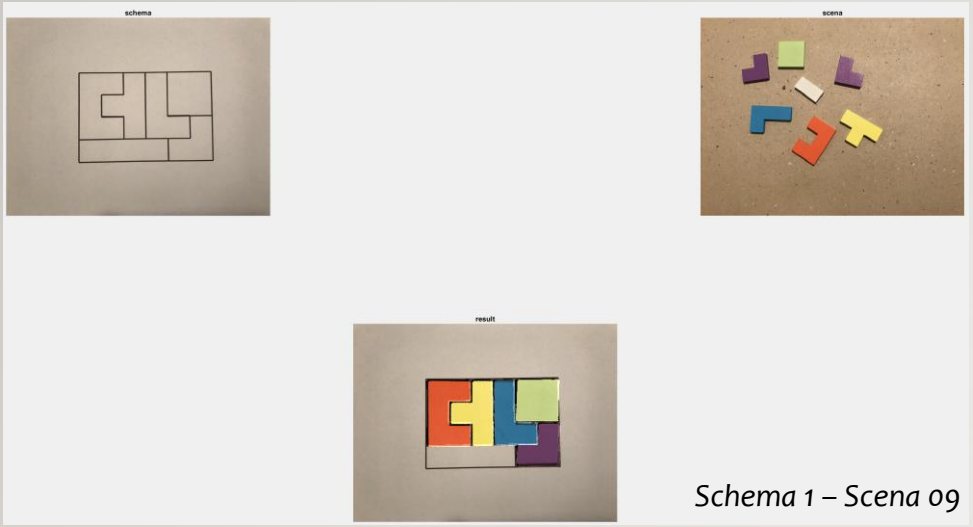




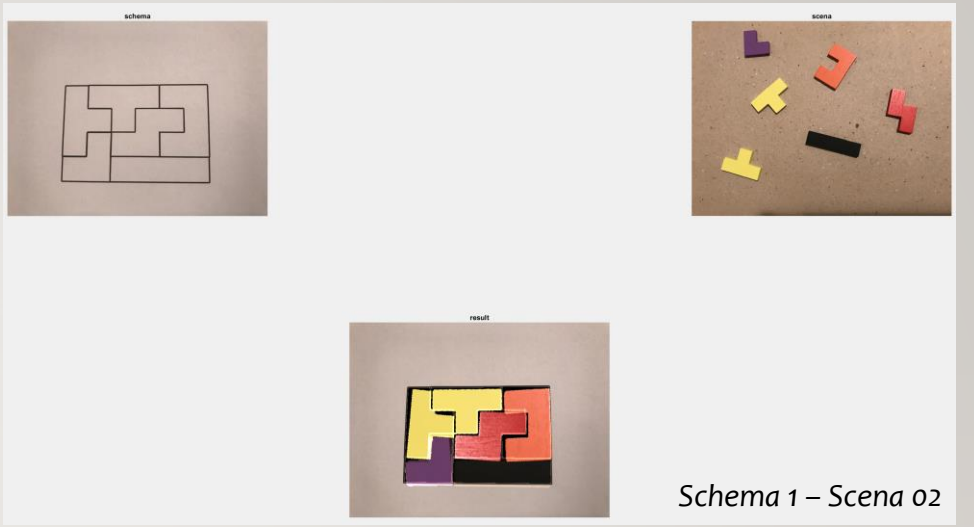


Tetramino ritagliato e raddrizzato

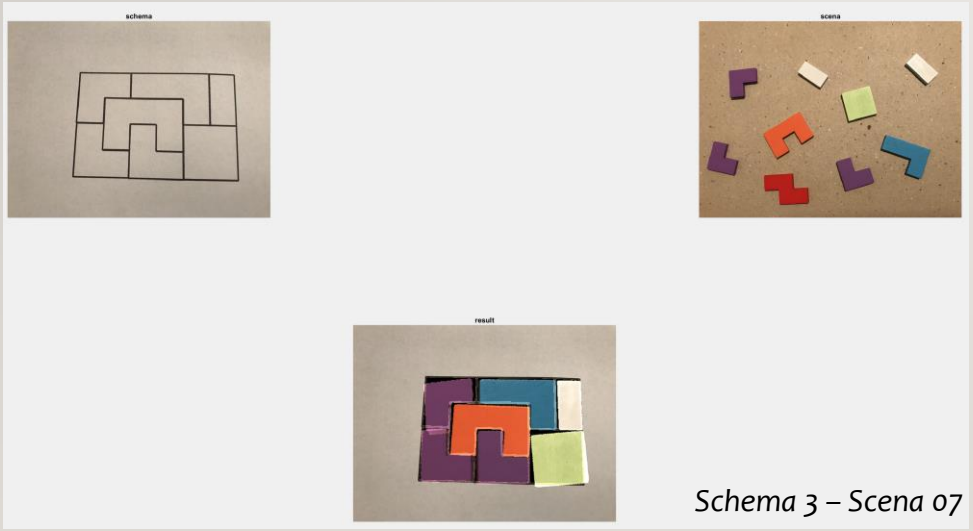
Alcuni risultati ottenuti



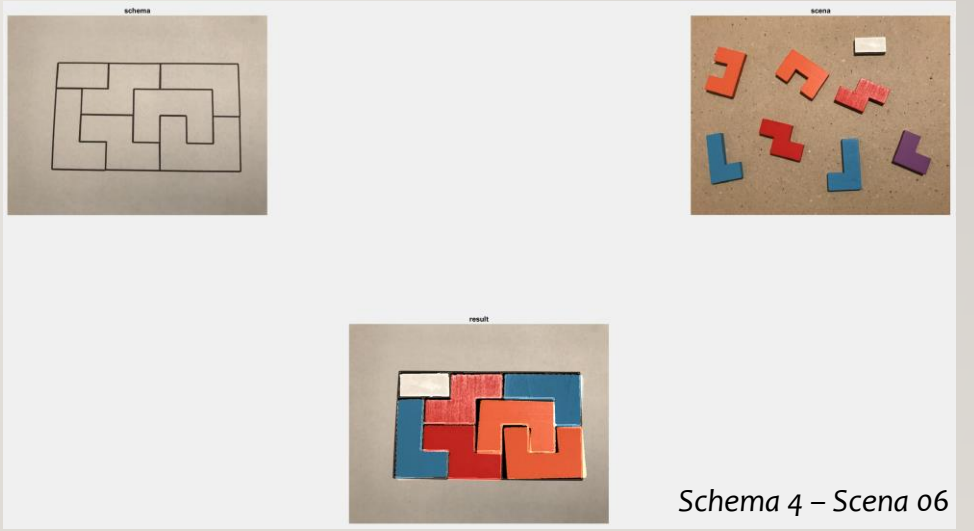
Schema 1 – Scena 09



Schema 1 – Scena 02



Schema 3 – Scena 07



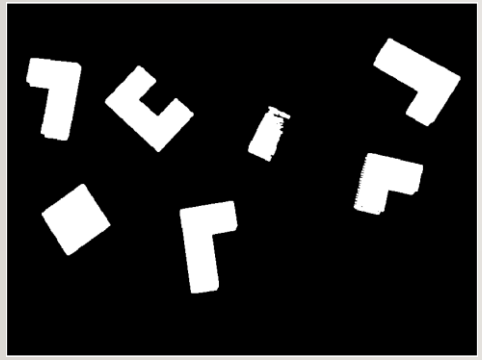
Schema 4 – Scena 06

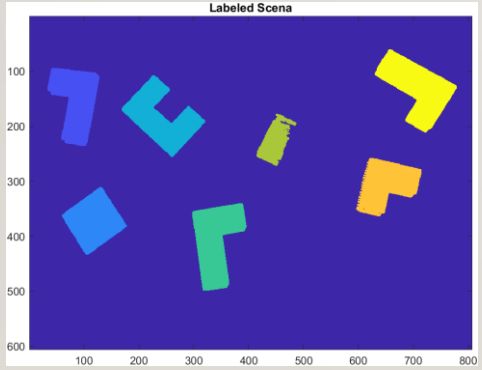
Analisi Critica dei Dati e Errori Riscontrati

Abbiamo considerato errore quando l'algoritmo non riconosce o non riesce a collocare anche un solo tetramino correttamente.

Analizzando i risultati ottenuti abbiamo riscontrato un accuratezza del **91.66%**

Dove gli **errori** si sono concentrati tutti nella **scena 5,** questo perché c'è un errore di binarizzazione del rettangolo nero.





Questo è dovuto dal fatto che il tetramino nero ha un'illuminazione non costante e ha un colore differente (più chiaro) rispetto ai tetramini neri forniti nelle immagini di training.

Questo non permette al classificatore di riconoscere completamente il tetramino.

Calcolo Accuratezza

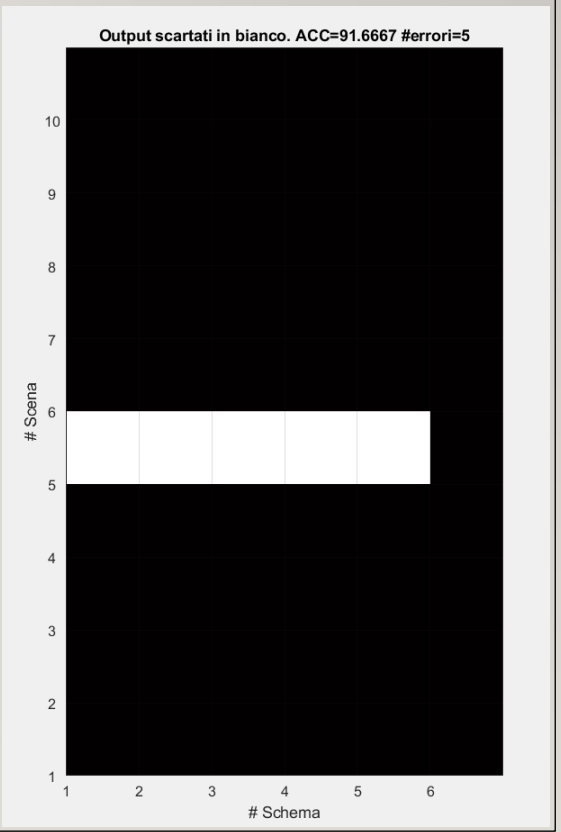
Per calcolare l'accuratezza abbiamo utilizzato un **metodo supervisionato**
Tramite uno script matlab ci venivano mostrate a schermo tutte le combinazioni possibili tra immagini di schema e scena *(quality_control.m)*

Tramite un click del mouse potevamo segnalare quali immagini presentavano degli errori.

In questo esempio abbiamo selezionato l'output2, dato che il rettangolo non è stato collocato correttamente

Al termine dell'esecuzione abbiamo ottenuto questo schema:

- 0 = L'immagine è **corretta**
- 1 = L'immagine presenta un **errore**



Considerando i casi nel dettaglio:

*Il rettangolo nello schema **1,2** e **5** non viene posizionato.*

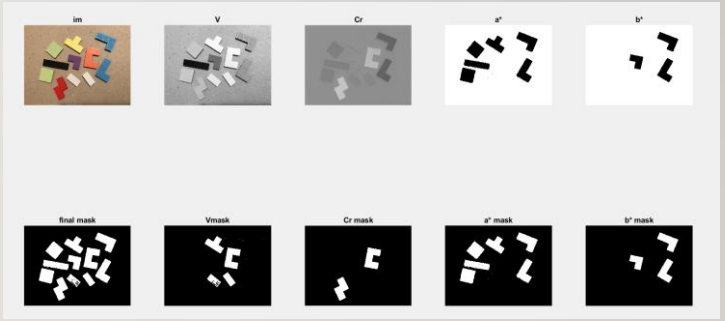
*Nello schema **3** e **4** viene riconosciuto come un falso positivo, posizionandolo al posto del rettangolo piccolo*

Metodi Scartati

Durante lo sviluppo dell'algoritmo abbiamo provato ad utilizzare alti metodi per risolvere i problemi che riscontravamo, in particolare abbiamo provato:

- **Etichetta Schema - Binarizzazione** dello **schema** usando **Otsu**
  Produceva un'immagine binarizzata male, dato che l'istogramma non presenta dei picchi netti.

- **Etichetta Scena - Binarizzazione** della **scena** usando **canali colore** *(YCbCr, RGB, LAB, HSV)* (anche combinati)
  Si aveva la necessità di impostare troppe soglie manualmente (super modellare il training set) e alcuni tetramini rimanevano comunque fusi con lo sfondo e quindi difficilmente binarizzabili.



- **Etichetta Scena -** Aggiungere **pixel** di **ombra** nell'immagine che riconosce lo sfondo per evitare che venga etichettato come tetramino
  Genera dei problemi quando si cerca di classificare i tetramini neri (confondeva sfondo e tetramino)

- **Riconosci Tetramino - Differenziazione** degli oggetti con la **compattezza**
  Difficile trovare una soglia affidabile per ogni tetramino

- **Riconosci Tetramino - Differenziazione** degli oggetti con **Momenti** di **Hu**
  Metodo effettivamente corretto, ma con una percentuale di errore maggiore rispetto alla correlazione

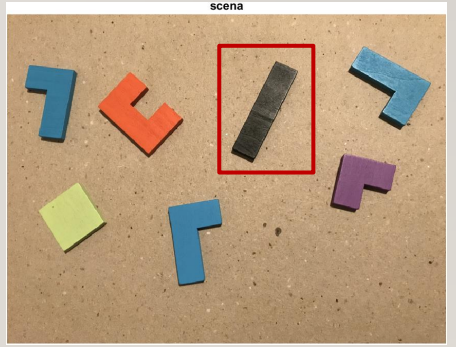
Conclusione

In conclusione l'algoritmo nella situazione ottimale produce un immagine che **non** ha elementi **sovrapposti,** ha **tutti** i **tetramini** collocati **correttamente** e **non** commette **errori** di **posizionamento**.

Ci sono però anche schemi dove la **rotazione** di alcuni tetramini è **leggermente sbagliata** e quindi vengono collocati storti di qualche grado.
Nell'immagine di schema questo si nota dalla presenza di un bordo bianco o nero nell'intorno dell'oggetto

Dove siamo sicuri che l'algoritmo **sbagli** è nella binarizzazione del rettangolo nero della scena 5.

Non binarizzandolo correttamente non viene riconosciuto e non viene posizionato nello schema.



Percentuale di Partecipazione del Gruppo

** = metodi sviluppati ma scartati, ma su cui è stato investito del tempo*

- **Manganaro Francesco** ~ 30%
  - *Etichetta Schema, Scena**
  - *Differenziazione con compattezza**
  - *Differenziazione con proprietà delle regioni**
  - *Organizzazione Progetto (main.m)*
  - *Analisi e controllo errori*
  - *Presentazione PT*

- **Gargaro David** ~ 10%
  - *Ricerca per implementare il codice*
  - *Idee per risoluzione problemi*

- **Pozzi Michele** ~ 30%
  - *Etichetta Schema, Scena**
  - *Piazza Tetramino*
  - *Creazione di main_batch.m*
  - *Script per Accuratezza*

- **Pretell Kevin** ~ 30%
  - *Etichetta Schema, Scena*
  - *Classificatore KNN*
  - *Differenziazione con Momenti di Hu**