



UNIVERSITÀ DEGLI STUDI DI BERGAMO

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA
INFORMATICA

ROBOTICA (PRINCIPI E PROGETTO)

RELAZIONE PROGETTO

Navigazione con marcatori visivi

Autori:

Michele Brillante
Pietro Boni
Simone Giusso

Numero di matricola:

1026039
1029777
1034978

Indice

1	Introduzione	1
2	Filtraggio in base al colore	2
3	Filtraggio in base alla forma	5
4	Calcolo delle coordinate dei marcatori	8
4.1	Calcolo parametri del FOV	8
4.2	Conversione pixel-centimetri del centro di un marcitore	11
5	Filtraggio in base alla dimensione	13
5.1	Filtraggio in base al diametro	13
5.2	Filtraggio in base alla parziale osservabilità di un marcitore	14
6	Conclusioni	16
	Utilizzo del codice	19

1 Introduzione

Per il progetto del corso di Robotica è stato implementato un algoritmo in C++ che, a partire da un'immagine di input, è in grado di individuare marcatori di un determinato colore di forma circolare (sfruttando la libreria OpenCV), fornendo in output le coordinate dei centri di tali marcatori rispetto ad un sistema di riferimento. L'algoritmo è stato sviluppato col presupposto di avere a disposizione un sistema visivo composto da una sola telecamera RGB (nel caso della seguente trattazione è stata usata una webcam Spotlight). Inoltre, l'algoritmo fornisce la possibilità di impostare colore, dimensione del marcitore da rilevare e l'angolo che la telecamera forma con l'asse verticale. Per lo sviluppo dell'algoritmo sono state fatte alcune considerazioni necessarie a semplificare il problema, come ad esempio l'assenza di dislivelli e forte contrasto di colore tra terreno e marcatore. In Figura 1 sono mostrate le componenti sviluppate per la realizzazione del progetto.

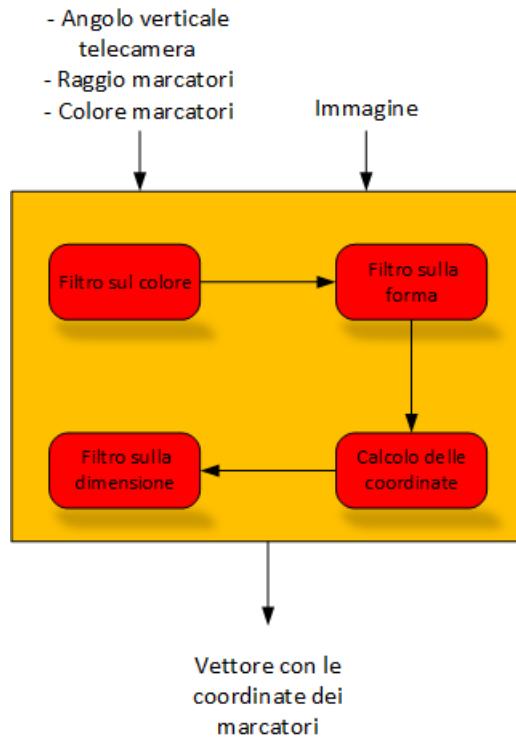


Figura 1: rappresentazione schematica dell'algoritmo realizzato.

2 Filtraggio in base al colore

Il primo problema affrontato è stato l'individuazione di un determinato colore in un'immagine. Per fare ciò è stata inizialmente convertita l'immagine RGB in input nel formato HSV. Il formato HSV permette di rappresentare i colori in un sistema digitale tramite i criteri di tonalità, saturazione e luminosità.

- **Tonalità:** indica il colore vero e proprio (rosso, giallo, verde, ecc...).
- **Saturazione:** indica la quantità di grigio nel colore.
- **Luminosità:** indica, in combinazione con la saturazione, la luminosità o l'intensità del colore.

Il formato HSV è stato scelto in quanto fornisce informazioni sul colore meno rumorose rispetto a quelle fornite da altri formati come RGB. Ad esempio, è in grado di rimuovere le ombre ed adattarsi a diverse condizioni di illuminazione, fonte principale di problemi nell'individuazione dei colori in questo progetto. Le impostazioni relative ad un colore sono avvenute in maniera sperimentale tramite codice, settando gli opportuni valori massimi e

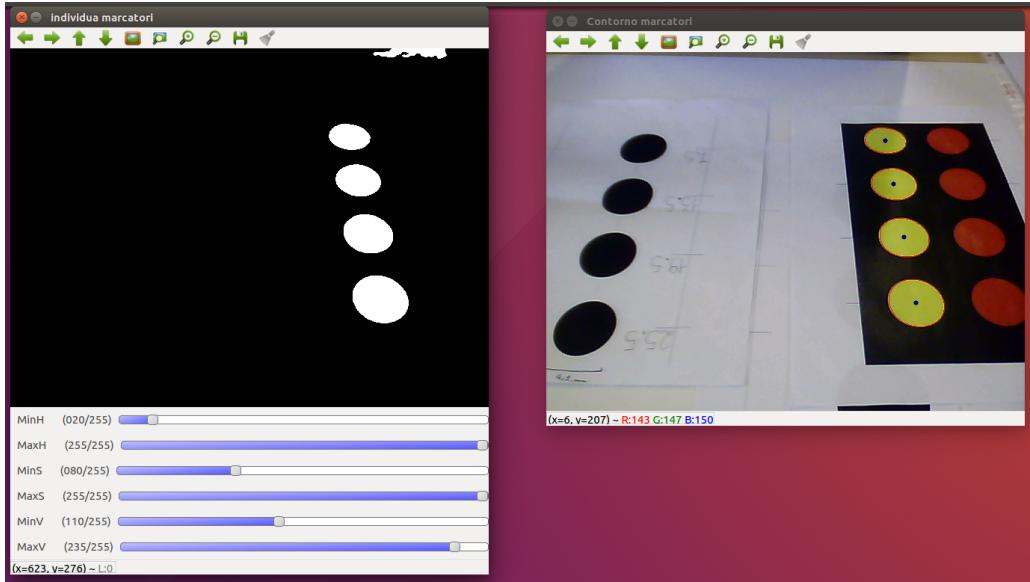


Figura 2: immagine HSV in bianco e nero di marcatori gialli su terreno di colore nero (il bianco rappresenta il colore giallo, il nero altri colori).

minimi di tonalità, saturazione e luminosità in tempo reale, fino ad ottenere nell’immagine HSV in bianco e nero il solo colore di interesse (come mostrato in Figura 2). La porzione di codice che offre questa funzionalità è la seguente:

```

146 // Settaggio trackbar per impostazione colore .
147 namedWindow( windowName );
148 createTrackbar( "MinH" , windowName , &minH , 255 );
149 createTrackbar( "MaxH" , windowName , &maxH , 255 );
150 createTrackbar( "MinS" , windowName , &minS , 255 );
151 createTrackbar( "MaxS" , windowName , &maxS , 255 );
152 createTrackbar( "MinV" , windowName , &minV , 255 );
153 createTrackbar( "MaxV" , windowName , &maxV , 255 );

```

Come mostrato in Figura 2, vengono generate sei trackbar: configurando in modo opportuno (perlopiù per tentativi) gli intervalli tramite le trackbar è possibile individuare diversi tipi di colore. In particolare, nell’immagine si possono osservare i range di tonalità, saturazione e luminosità trovati per l’individuazione del colore giallo (riportati in Tabella 1). Dopo che sono stati individuati i parametri del colore desiderato è necessario riportarli manualmente nel costrutto **switch-case** descritto nella sezione "Utilizzo del codice". Una volta impostato il range di interesse tale porzione di codice può essere commentata.

Nel progetto sono stati analizzati marcatori di quattro colori diversi: nero, giallo, rosso e bianco. Mentre i marcatori di colore nero sono stati analizzati su sfondo bianco, per gli altri colori è stato utilizzato uno sfondo nero. In particolare, la configurazione di marcatori di colore rosso si discosta dagli altri

	Tonalità		Saturazione		Luminosità	
	min	max	min	max	min	max
Bianco	50	150	0	75	130	210
Nero	0	255	0	255	0	85
Giallo	20	255	80	255	110	235
Rosso	0	30	70	255	50	255
	130	190	70	255	50	255

Tabella 1: range di individuazione dei quattro colori analizzati.

colori in quanto è stato necessario definire due diversi range di individuazione. Infatti, utilizzando un singolo intervallo vi è il rischio di rilevare colori diversi dal rosso o di non rilevare marcatori rossi soggetti ad una particolare illuminazione. In Tabella 1 sono riportati i range utilizzati per l’individuazione dei marcatori dei colori analizzati, espressi tramite i valori massimi e minimi di tonalità, saturazione e luminosità per singolo colore.

Oltre all’utilizzo di un filtro sul colore, è stato applicato un filtro lineare per uniformare l’immagine al fine di ridurre il rumore. In particolare, è stato applicato il filtro mediano, con l’utilizzo della funzione `medianBlur()`, il quale sostituisce ciascun pixel con la mediana dei pixel adiacenti (quelli che formano un quadrato attorno al pixel valutato). Inoltre, tramite l’utilizzo della funzione `findContours()`, vengono individuati i contorni degli oggetti di un determinato colore, mentre con la funzione `moments()` è possibile rilevare il centro di tali oggetti. In Figura 3 vengono mostrati i risultati ottenuti in questa prima fase, in cui ad ogni oggetto con un determinato colore viene sovrapposto un contorno di colore rosso.

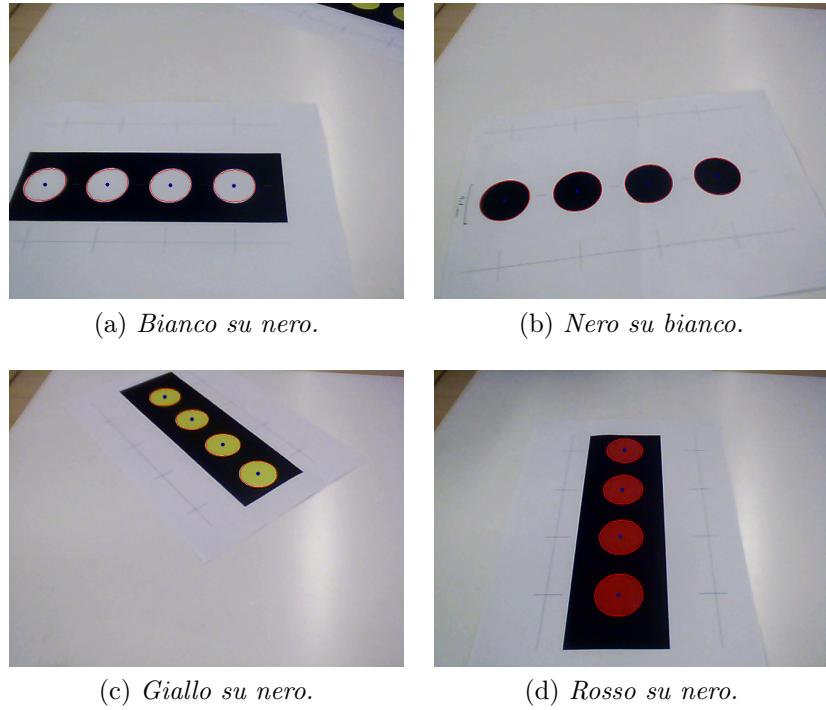


Figura 3: rilevamento di marcatori con diversi colori.

3 Filtraggio in base alla forma

Ai fini del progetto sono stati utilizzati semplici marcatori di forma circolare. In assenza di funzioni in grado di identificare e gestire questo tipo di marcatori, è stata adottata una strategia che consente di eliminare, in buona parte, oggetti con forma differente rispetto a quella circolare. Il primo approccio adottato è basato sull'utilizzo di un'apposita funzione di OpenCV per il riconoscimento di oggetti circolari. Questa prima soluzione è stata subito scartata in quanto, prospetticamente, i marcatori tendono ad assumere una forma ellissoidale (Figura 4).

Poiché per ogni oggetto rilevato si possono ottenere informazioni su area e perimetro in termini di pixel, rispettivamente tramite le funzioni `arcLength()` e `contourArea()`, si è deciso di confrontare (in prima approssimazione) il perimetro degli oggetti rilevati con la formula del perimetro di un cerchio, dipendente dall'area del cerchio stesso:

$$p = \sqrt{4 \cdot \pi \cdot A}$$

Poiché nella pratica i marcatori circolari rilevati dalla cam risultano avere forma ellissoidale, è stato trovato sperimentalmente un range entro il quale il perimetro degli oggetti rilevati si deve trovare affinché un oggetto venga

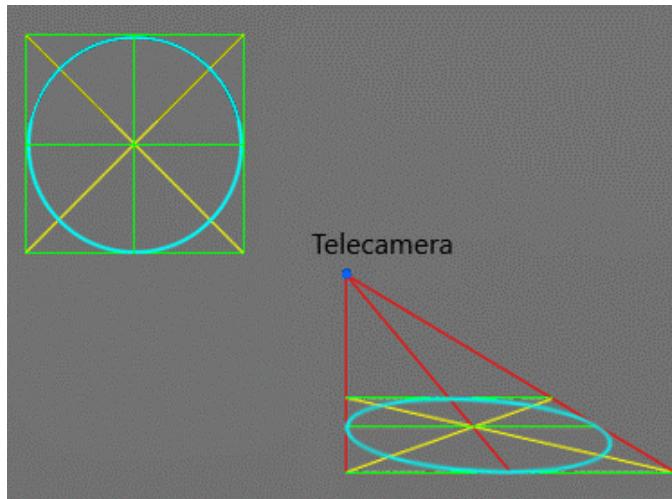


Figura 4: rappresentazione grafica della visione di una forma circolare da parte di una telecamera (prospetticamente si ha l'equivalente di un'ellisse).

considerato approssimativamente di forma circolare:

$$\sqrt{4 \cdot \pi \cdot A_o} - 15 \leq p_o \leq \sqrt{4 \cdot \pi \cdot A_o} + 28$$

dove p_o e A_o sono rispettivamente il perimetro e l'area dell'oggetto rilevato.

Un altro approccio poteva essere dato dal confronto tra il perimetro di un'ellisse generica (sulla falsariga di quanto discusso per un cerchio generico) con il perimetro calcolato dell'oggetto, ma ciò presenta due problemi: la mancanza di una relazione che lega area e perimetro di un'ellissi e la difficoltà nel determinare gli eventuali semiassi dell'oggetto rilevato per il calcolo del perimetro. Perciò è stato deciso di utilizzare le più fruibili formule della circonferenza (con i dovuti accorgimenti).

Oltre che il perimetro, per il filtraggio in base alla forma è stata analizzata anche l'area dei marcatori, per aggiungere maggiore robustezza al filtraggio. In particolare, sono state confrontate l'area ricavata tramite comando `contourArea()` e l'area attesa del marcatore (in pixel) ricavata dalla seguente formula:

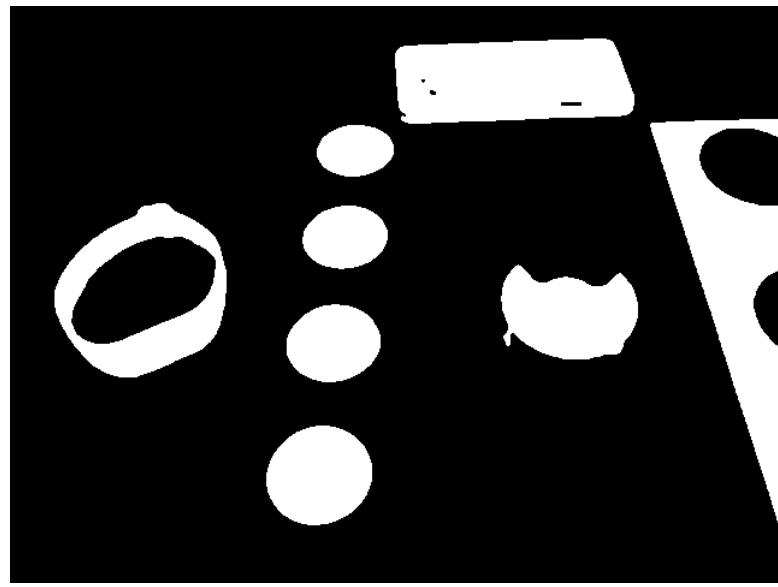
$$A_c = \frac{\frac{1}{h_{cam} - 23} \cdot 4000 - h_{cam} + 80}{\tan(\alpha_{cam} - 7)}$$

con α_{cam} l'angolo formato tra asse della telecamera e asse verticale e h_{cam} altezza della telecamera da terra. Tale formula è stata ricavata sperimentalmente per interpolazione basandosi sulla variazione dell'area di un marcitore posto a diverse distanze rispetto all'inquadratura della cam e tenendo conto di inclinazione ed altezza della cam stessa. Infatti, più l'angolo α_{cam} o l'altezza h_{cam} aumentano, più l'area di un marcitore inquadrato ad una distanza fissa tende a diminuire. In particolare, tale formula tende a dare una stima a ribasso dell'area desiderata, per cui affinché un oggetto venga considerato un marcitore la sua area calcolata deve rispettare la seguente relazione:

$$A_o > A_c$$

altrimenti viene scartato.

Come mostrato in Figura 5, per testare la bontà del filtraggio in base ad area e perimetro, sono stati inquadrati con la telecamera diversi oggetti con lo stesso colore ma di forma differente. Come desiderato sono stati scartati gli oggetti con una forma differente rispetto a quella circolare dei marcatori di interesse, che risultano evidenziati da un contorno rosso.



(a) *Identificazione degli oggetti in base al colore.*



(b) *Identificazione degli oggetti in base a colore e forma.*

Figura 5: dimostrazione del funzionamento del filtro in base alla forma, in base al quale gli oggetti con forma non approssimabile a quella di un cerchio vengono scartati.

4 Calcolo delle coordinate dei marcatori

4.1 Calcolo parametri del FOV

Dopo l'identificazione dei marcatori di interesse per forma e colore, il passo successivo è il calcolo delle coordinate del centro di tali marcatori. Per tale calcolo sono state fatte diverse considerazioni geometriche basate su:

- apertura focale orizzontale e verticale della cam
- posizione della cam rispetto al sistema di riferimento
- inclinazione α_{cam} della telecamera rispetto all'asse verticale, intesa come angolo fra la verticale e l'asse della telecamera
- posizione dei marcatori nelle immagini catturate dalla cam.

In particolare, si è scelto come sistema di riferimento con cui rilevare le coordinate dei marcatori il punto di intersezione tra il piano su cui giacciono i marcatori e la retta verticale passante per il centro della cam utilizzata,

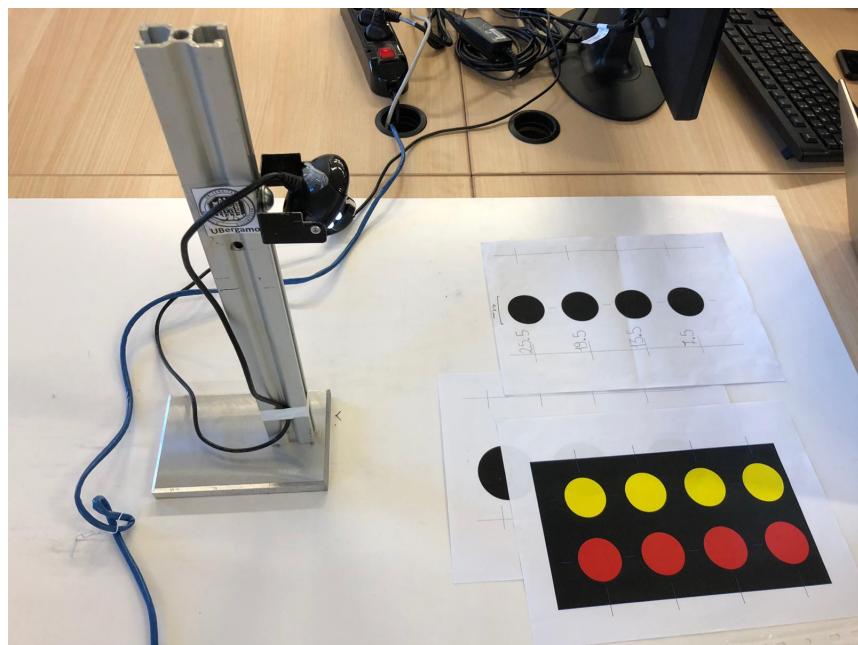


Figura 6: postazione di lavoro per il rilevamento dei marcatori.

in modo da esprimere la posizione della telecamera in soli termini di altezza da terra¹. Come altezza è stata mantenuta una distanza dal terreno (sul quale si trovano i marcatori) di 30.8 cm, mentre per l'inclinazione rispetto la verticale sono stati usati principalmente tre valori: 30°, 45° e 60°. Una tipica configurazione della postazione di lavoro per il rilevamento dei marcatori è riportata in Figura 6.

Le aperture focali della telecamera (orizzontale e verticale) sono state calcolate ponendo la camera parallela al piano inquadrato, ad una distanza nota di 10 cm. Quindi, misurando i lati del campo visivo inquadrato a tale distanza è stato possibile ricavare le aperture angolari della cam tramite formule trigonometriche, da cui è risultata un'apertura focale orizzontale $f_x \simeq 49^\circ$ ed un'apertura focale verticale $f_y \simeq 37.7^\circ$.

Calcolate le aperture angolari f_x e f_y è quindi possibile effettuare i calcoli necessari per le misure relative al FOV (*Field of View*). In particolare, posta la cam ad una determinata altezza e con una determinata inclinazione rispetto la verticale, la situazione che viene a crearsi è simile a quella rappresentata in Figura 7, in cui si ha una piramide retta con altezza corrispondente all'altezza della cam rispetto al terreno (indicata da qui in poi con h_{cam}) e FOV compreso nella faccia di base (rappresentato dalla zona in blu).

Quindi, noti h_{cam} e α_{cam} a priori ed avendo calcolato l'apertura focale verticale f_y , è possibile calcolare la distanza $y_{FOV,0}$ tra sistema di riferimento e FOV e la lunghezza y_{FOV} dello stesso campo visivo tramite le seguenti relazioni:

$$y_{FOV,0} = h_{cam} \cdot \tan \left(\alpha_{cam} - \frac{f_y}{2} \right) \quad (1)$$

$$y_{FOV} = h_{cam} \cdot \tan \left(\alpha_{cam} + \frac{f_y}{2} \right) - y_{FOV,0} \quad (2)$$

In particolare, il campo visivo inquadrato dalla telecamera può essere considerato come un trapezio isoscele di cui possono essere calcolate la base maggiore B (quella più distante dalla cam) e la base minore b (quella più vicina alla cam). Per il calcolo di B è stata anzitutto calcolata la distanza y_{sup} tra la telecamera e B stessa, corrispondente all'altezza del triangolo

¹Nel caso di una cam montata su di un robot navigatore il sistema di riferimento è generalmente posto all'interno del robot stesso; si ha solo una variazione della posizione della cam rispetto al sistema di riferimento, i ragionamenti relativi al calcolo delle coordinate dei marcatori restano inalterati.

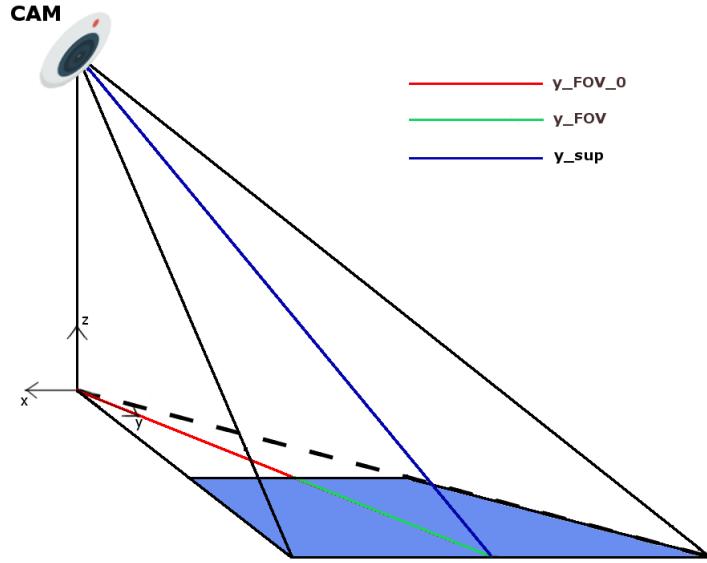


Figura 7: piramide retta determinata da posizione ed inclinazione della cam.

che costituisce la faccia superiore della piramide retta in Figura 7, tramite teorema di Pitagora:

$$y_{sup} = \sqrt{(y_{FOV,0} + y_{FOV})^2 + h_{cam}^2}$$

con la quale è possibile calcolare il valore di B :

$$B = 2 \cdot y_{sup} \cdot \tan\left(\frac{f_x}{2}\right)$$

Invece, per il calcolo di b è stato necessario calcolare l'angolo a terra α_t corrispondente all'apertura focale orizzontale della cam. Tale angolo è stato calcolato con la seguente relazione:

$$\alpha_t = 2 \cdot \arctan\left(\frac{B}{2(y_{FOV,0} + y_{FOV})}\right)$$

da cui la base minore b del FOV risulta essere data da:

$$b = 2 \cdot y_{FOV,0} \cdot \tan\left(\frac{\alpha_t}{2}\right)$$

4.2 Conversione pixel-centimetri del centro di un marcatore

Il passo successivo è dato dalla conversione in centimetri delle coordinate in pixel dei centri dei marcatori (trovati tramite la funzione `moments()`, come spiegato nel capitolo 2). Per fare ciò è stato anzitutto tenuto conto della risoluzione della cam, corrispondente ad una matrice di 640x480 pixel. In particolare, risoluzione orizzontale e verticale sono state utilizzate distintamente, ponendo $res_x = 640$ e $res_y = 480$. Inoltre, i pixel della cam sono stati considerati singolarmente, ognuno con una propria inclinazione caratteristica corrispondente ad una porzione dell'apertura focale verticale f_y e dell'angolo a terra α_t . In particolare, l'angolo caratteristico verticale di un pixel è stato chiamato β , mentre quello orizzontale γ . In Figura 8 è mostrata l'associazione di un pixel corrispondente al centro di un marcitore con il relativo angolo β (l'associazione relativa all'angolo γ è analoga).

Quindi, trovate le coordinate del centro di un marcitore po_x e po_y in pixel, è possibile calcolare gli angoli caratteristici β e γ per proporzione:

$$res_y : po_y = f_y : \beta$$

$$res_x : po_x = \alpha_t : \gamma$$

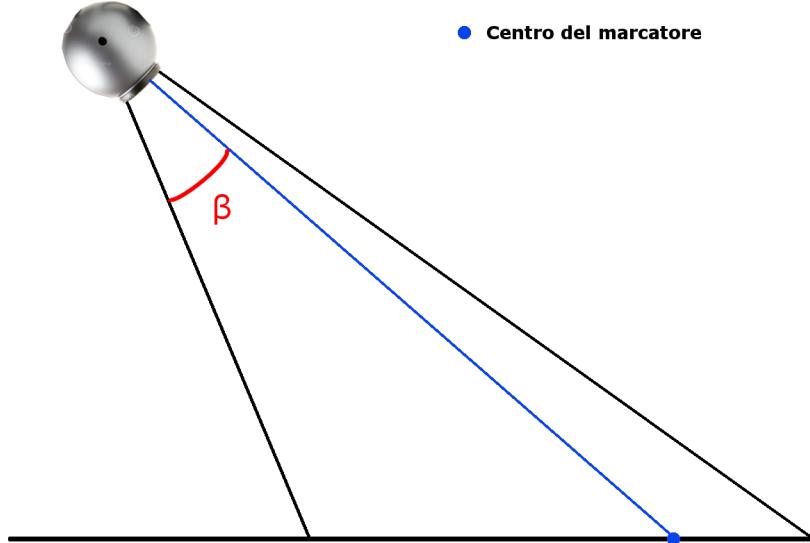


Figura 8: rappresentazione grafica dell'angolo β relativo al pixel corrispondente al centro di un marcitore.

da cui si ricava:

$$\beta = f_y \cdot \frac{po_y}{res_y}$$

$$\gamma = \alpha_t \cdot \frac{po_x}{res_x}$$

Trovati β e γ è quindi possibile il passaggio dalle coordinate in pixel alle coordinate in centimetri rispetto al sistema di riferimento. In particolare, per la coordinata del centro di un marcatore y_{TOT} il ragionamento è equivalente a quello utilizzato per il calcolo di $y_{FOV,0}$ e y_{FOV} (equazioni 1 e 2):

$$y_{TOT} = h_{cam} \cdot \tan \left(\alpha_{cam} + \beta - \frac{f_y}{2} \right)$$

Per la coordinata del centro x_{TOT} è stato fatto un ragionamento differente: mentre y_{TOT} avrà sempre valore positivo, x_{TOT} può assumere anche valori negativi, in quanto l'asse y del sistema di riferimento rappresenta la proiezione dell'asse della cam a terra, tagliando quindi a metà le immagini da essa acquisita. Per questo motivo, la x_{TOT} è stata calcolata basandosi sull'angolo a terra α_t dimezzato e sulla y_{TOT} già calcolata, tramite le formule trigonometriche dei triangoli rettangoli:

$$x_{TOT} = \tan \left| \frac{\alpha_t}{2} - \gamma \right| \cdot y_{TOT}$$

L'utilizzo del valore assoluto per il calcolo della tangente è vincolato al fatto che non sapendo se la coordinata sia positiva o negativa la si calcola anzitutto come valore positivo. Per determinare se x_{TOT} sia positiva o negativa si ragiona sulla coordinata in pixel corrispondente; in particolar modo, il ragionamento è il seguente:

$$x_{TOT} = \begin{cases} x_{TOT} & \text{se } po_x > \frac{res_x}{2} \\ -x_{TOT} & \text{se } po_x < \frac{res_x}{2} \end{cases} \quad (3)$$

5 Filtraggio in base alla dimensione

5.1 Filtraggio in base al diametro

L'algoritmo sviluppato consente di dare in input per il filtraggio, oltre al colore, anche la dimensione dei marcatori da rilevare, tramite l'indicazione del raggio r . Tale filtraggio aggiuntivo permette di distinguere marcatori di dimensioni diverse, mantenendo unicamente quelli con la dimensione desiderata. In particolare, tale filtraggio viene applicato subito prima del passaggio descritto dall'equazione 3.

Per fare ciò vengono anzitutto inizializzati gli angoli γ_{max} e γ_{min} , corrispondenti agli angoli caratteristici dei pixel relativi all'estremo destro max_x e sinistro min_x (in pixel) del contorno rilevato di un marcitore:

$$\gamma_{max} = max_x \cdot \frac{\alpha_t}{res_x}$$

$$\gamma_{min} = min_x \cdot \frac{\alpha_t}{res_x}$$

Una volta ottenuti gli angoli γ_{max} e γ_{min} è possibile convertire le coordinate max_x e min_x in centimetri (rispettivamente x_{max} e x_{min}) nel seguente modo:

$$x_{max} = y_{TOT} \cdot \tan \left| \frac{\alpha_t}{2} - \gamma_{max} \right|$$

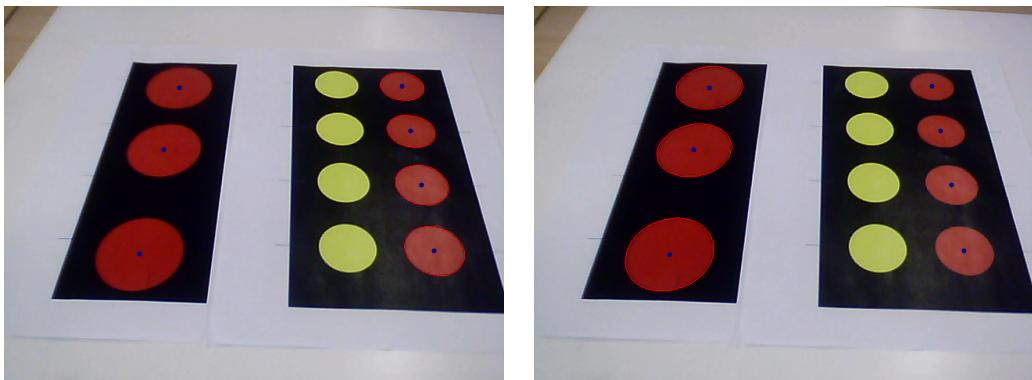


Figura 9: rappresentazione del filtraggio basato sul diametro di un marcatore.

$$x_{min} = y_{TOT} \cdot \tan \left| \frac{\alpha_t}{2} - \gamma_{min} \right|$$

A questo punto il diametro d di un marcitore può essere espresso come:

$$d = |x_{max} - x_{min}| \cdot (1.2 + \cos(60^\circ + \beta))$$

dove il secondo fattore rappresenta un termine di correzione, dipendente dall'angolo β corrente.

Quindi, ricavato il diametro d (in centimetri) è possibile effettuare il filtraggio sulla base del raggio reale del marcitore. In particolare, dati gli eventuali errori di misura e di arrotondamento, è stato utilizzato un determinato range entro il quale d deve trovarsi, ossia:

$$1.75 \cdot r < d < 2.75 \cdot r$$

Come mostrato in Figura 9 il filtraggio così imposto permette la distinzione tra marcatori di forma circolare dello stesso colore ma di differenti dimensioni.

5.2 Filtraggio in base alla parziale osservabilità di un marcitore

Nel caso un marcitore sia solo parzialmente presente nell'inquadratura della telecamera (ossia se risulta tagliato ai bordi dell'immagine) può capitare che, nonostante i filtri discussi in precedenza, esso venga comunque riconosciuto come un marcitore nonostante la forma non circolare. Ciò potrebbe essere visto come un vantaggio, ma potrebbe altresì portare alla rilevazione di oggetti a bordo immagine che non rappresentano effettivamente un marcitore. Per ovviare a questo problema è stato quindi introdotto un ulteriore filtro per garantire una ancora maggiore robustezza all'algoritmo. In particolare, anche questo filtro viene applicato prima del passaggio descritto dall'equazione 3, per cui x_{TOT} risulta essere sempre positiva.

Tale filtraggio è sempre basato sul raggio r di input. In particolare, viene anzitutto calcolata la larghezza dell'immagine x_{imm} in corrispondenza del centro di un marcitore:

$$x_{imm} = \frac{B \cdot y_{TOT}}{y_{FOV,0} + y_{FOV}}$$

Quindi, sfruttando il risultato ottenuto, la coordinata x_{TOT} del centro di un marcitore ed il raggio reale r dei marcatori di interesse, il filtraggio è rappresentato dalla condizione seguente:

$$\frac{x_{imm}}{2} - x_{TOT} > 0.65 \cdot r$$

ossia viene calcolata la distanza del marcitore rilevato dalla telecamera da bordo immagine (termine a sinistra della disequazione) ed esso viene confrontato con il raggio del marcitore reale entro una certa tolleranza (termine a destra della disequazione). Tale restrizione permette comunque la rilevazione di marcatori tagliati dal bordo destro e sinistro dell'immagine (come mostrato in Figura 10), ma solo entro una certa tolleranza decisa sperimentalmente a priori. In particolare, tale filtraggio è stato effettuato solo rispetto ai bordi destro e sinistro dell'immagine di acquisizione in quanto sono risultati essere i più critici; per i bordi superiore ed inferiore sono risultati sufficienti i filtri descritti in precedenza.

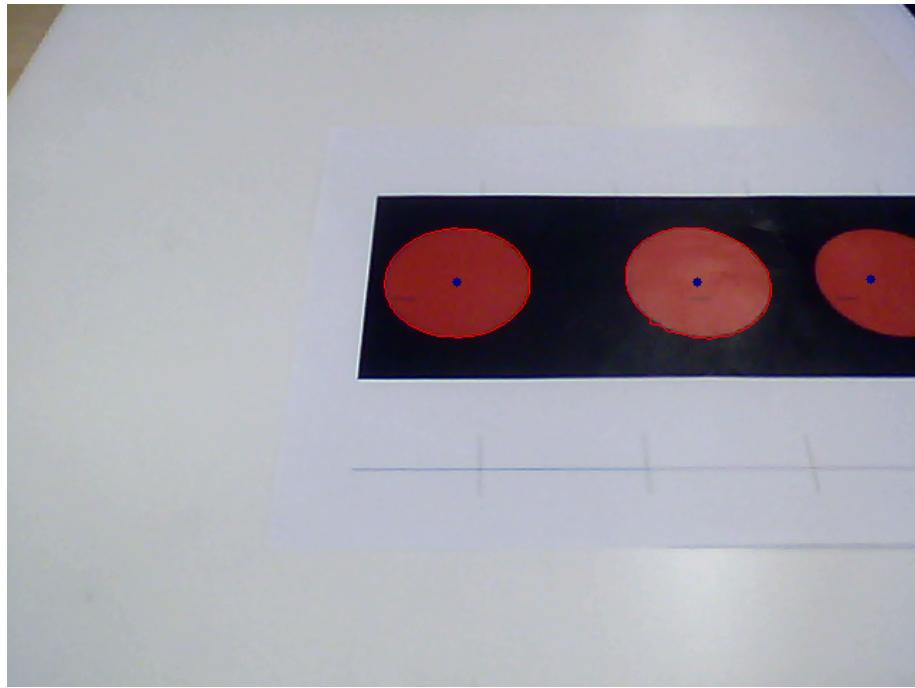


Figura 10: rappresentazione del filtraggio basato sulla parziale osservabilità di un marcatore.

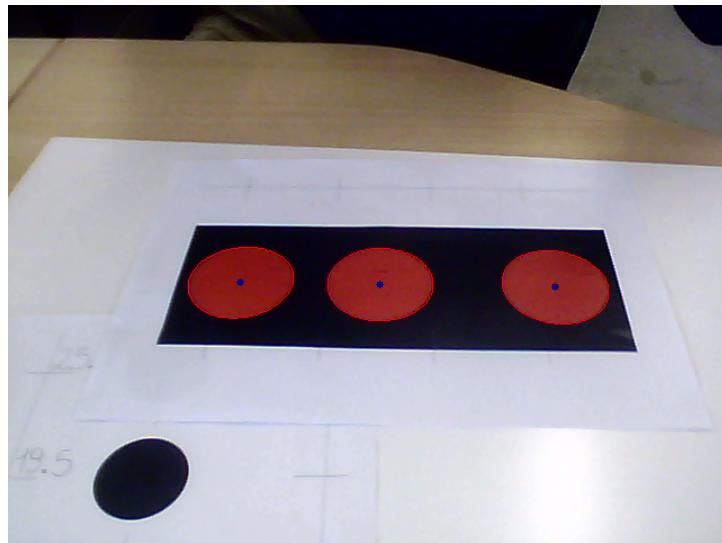
6 Conclusioni

L'algoritmo, dopo le operazioni di filtraggio e calcolo delle coordinate, produce in output un vettore di oggetti con le coordinate x e y (in metri) del centro dei marcatori rilevati. Immaginando che vengano rilevati n marcatori, l'output sarà il seguente:

$$output = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix}$$

In particolare, in Figura 11 e 12 sono riportati due esempi di rilevamento dei marcatori e calcolo delle coordinate, stampate in output su linea di comando. In particolare, l'algoritmo opera in modo iterativo: il codice viene eseguito ciclicamente, consentendo un filtraggio e un ricalcolo continuo delle coordinate dei marcatori rilevati, permettendo la gestione della variazione di posizione del robot rispetto ai marcatori, dovuta al movimento del robot stesso.

L'algoritmo si è rivelato essere abbastanza robusto, specialmente per il calcolo delle coordinate dei marcatori filtrati, in quanto basate su considerazioni geometriche, per cui soggette a soli errori di tipo sistematico (accuratezza nell'indicazione dell'altezza della telecamera, calcolo delle aperture focali della telecamera ecc.), per cui in qualche misura compensabili. Il limite maggiore è dato dalle condizioni di luce dell'ambiente in cui si trovano i marcatori da rilevare: infatti, il filtraggio in base al colore è risultato essere il più suscettibile ad errori (mancata rilevazione dei marcatori, rilevazione errata di marcatori di altro colore ecc.). Ciò accade in quanto il colore rilevato, nonostante sia effettivamente il colore voluto dal filtraggio, a seconda della luce incidente, può tendere ad assumere tonalità e/o gradazioni fuori range rispetto a quelle indicate in Tabella 1. Tali errori sono stati comunque limitati sperimentando il funzionamento dell'algoritmo sotto varie condizioni di luce ed "aggiustando" il range di rilevamento di conseguenza.



(a) Rilevamento marcatori.

```

Coordinata x: -0.0720965 m
Coordinata y: 0.302975 m
Distanza puntatore-telegamma : 0.311435 m

Coordinata x: 0.104757 m
Coordinata y: 0.299669 m
Distanza puntatore-telegamma : 0.317452 m

Coordinata x: 0.00624131 m
Coordinata y: 0.301318 m
Distanza puntatore-telegamma : 0.301382 m

Coordinata x: -0.0720965 m
Coordinata y: 0.302975 m
Distanza puntatore-telegamma : 0.311435 m

Coordinata x: 0.104757 m
Coordinata y: 0.299669 m
Distanza puntatore-telegamma : 0.317452 m

Coordinata x: 0.00624131 m
Coordinata y: 0.301318 m
Distanza puntatore-telegamma : 0.301382 m

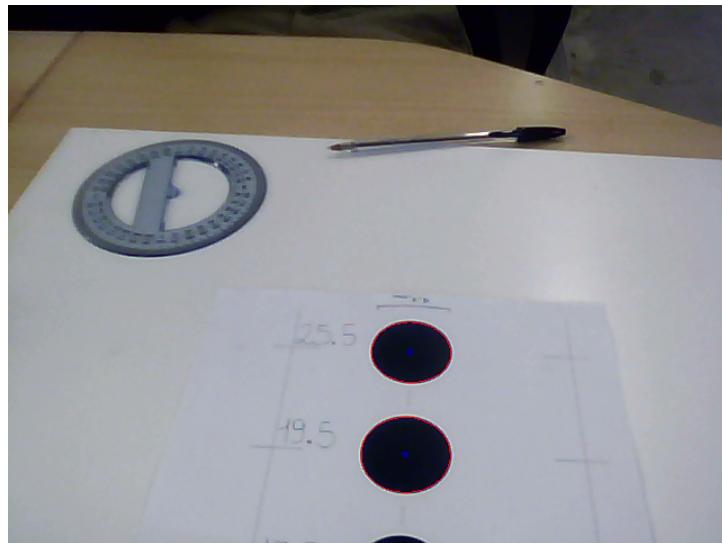
Coordinata x: -0.0720965 m
Coordinata y: 0.302975 m
Distanza puntatore-telegamma : 0.311435 m

[MarkerSpotRtUnit] : task()

MARKER_SPOT_DETECTOR: done
star_marker_spot_node: done
QObject::~QObject: Timers cannot be stopped from another thread
shutting down processing monitor...
... shutting down processing monitor complete
done
ser1@caronte03:~/STAR/ROS/src/star_marker_spot/launchs ]
```

(b) Output delle coordinate.

Figura 11: calcolo delle coordinate di marcatori rossi con la stessa coordinata y e diverse coordinate x.



(a) Rilevamento marcatori.

```

Coordinata x: 0.0126346 m
Coordinata y: 0.255336 m
Distanza puntatore-telegamma : 0.255648 m

Coordinata x: 0.00866215 m
Coordinata y: 0.196307 m
Distanza puntatore-telegamma : 0.196498 m

Coordinata x: 0.0126346 m
Coordinata y: 0.255336 m
Distanza puntatore-telegamma : 0.255648 m

Coordinata x: 0.00866215 m
Coordinata y: 0.196307 m
Distanza puntatore-telegamma : 0.196498 m

Coordinata x: 0.0126346 m
Coordinata y: 0.255336 m
Distanza puntatore-telegamma : 0.255648 m
Coordinata x: 0.00866215 m
Coordinata y: 0.196307 m
Distanza puntatore-telegamma : 0.196498 m

[star_marker_spot-1] killing on exit
[MarkerSpotRTUnit] : task()

MARKER_SPOT_DETECTOR: done
star_marker_spot_node: done
Qobject::~Qobject: Timers cannot be stopped from another thread
shutting down processing monitor...
... shutting down processing monitor complete
done
serl@caronte03:~/STAR/ROS/src/star_marker_spot/launch$ █

```

(b) Output delle coordinate.

Figura 12: calcolo delle coordinate di marcatori neri con coordinate x simili e coordinate y differenti.

Utilizzo del codice

L'intero algoritmo è stato implementato in un'unica funzione. Le azioni di filtraggio sopra riportate sono implementate nel codice tramite costrutti **if-else** annidati (secondo l'ordine di presentazione).

È possibile apportare delle modifiche al codice affinché possa essere utilizzato in condizioni diverse rispetto a quelle con cui è stato progettato e testato. In particolare, il costrutto switch-case riportato di seguito permette il settaggio del range entro cui individuare il colore immesso in input da linea di comando (la modalità di individuazione di un range per un determinato colore di interesse è descritta nella sezione 2).

```
95 switch (scelta)
96 {
97     case 'w':
98     case 'W': minH=50, maxH=150, minS=0, maxS=75,
99         minV=130, maxV=210; //Bianco su nero
100    break;
101
102    case 'b':
103    case 'B': minH=0, maxH=255, minS=0, maxS=255,
104        minV=0, maxV=85; //Nero su bianco
105    break;
106
107    case 'y':
108    case 'Y': minH=20, maxH=255, minS=80, maxS=255,
109        minV=110, maxV=235; //Giallo su nero
110    break;
111
112 }
```

È infatti possibile aggiungere ulteriori colori a quelli già definiti tramite gli intervalli di tonalità, saturazione e luminosità per un colore specifico. In

particolare, tramite la variabile **scelta**, contenente l'input fornito dall'utente, viene selezionato il colore dei marcatori che si vogliono individuare. Le variabili utilizzate hanno il seguente significato:

- **minH**: valore minimo di tonalità.
- **maxH**: valore massimo di tonalità.
- **minS**: valore minimo di saturazione.
- **maxS**: valore massimo di saturazione.
- **minV**: valore minimo di luminosità.
- **maxV**: valore massimo di luminosità.

Tramite la seguente porzione di codice è invece possibile inizializzare i parametri costanti relativi la telecamera.

```

123 //Parametri
124 float h_cam = 30.8;    //Altezza cam in cm
125 float f_y = 37.7;      //Apertura focale verticale cam
126 float f_x = 49.0;      //Apertura focale orizzontale cam
127 float res_x = 640.0;   //Risoluzione orizzontale cam
128 float res_y = 480.0;   //Risoluzione verticale cam

```

In particolare, ogni parametro è riportato nel codice con uguale denominazione rispetto alla presentazione nel testo, secondo la seguente modalità:

- α_{cam} → **alpha_cam**
- β → **beta**
- h_{cam} → **h_cam**
- $y_{FOV,0}$ → **y_FOV_0**
- ecc.