UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
CORSO DI LAUREA IN INGEGNERIA DELL' AUTOMAZIONE
ANNO ACCADEMICO 2010/2011

Algoritmi di pattugliamento 2D per reti di videosorveglianza multicamera

Relatore: Ch.mo prof. LUCA SCHENATO

Correlatore: Dr. Davide Raimondo

Laureando: *Marco Pattarello*

Padova, 5 Aprile 2011

... A mio Padre

Indice

1 Introduzione

Il problema di videosorvegliare un ambiente bidimensionale esterno o interno al fine di trovare un possibile evasore utilizzando una rete di telecamere è tutt'ora di grande attualità viste le sue innumerevoli applicazioni. Queste ultime spaziano dall'ambito militare a quello industriale, passando anche attraverso i robot mobili per il monitoraggio.
Il tutto viene comandato da un operatore umano, che solitamente, risiede in una stanza con molteplici monitor e muove a proprio piacimento le telecamere. La videosorveglianza viene quindi centralizzata e governata da un utente che, molto spesso, soprattutto se l'area da videosorvegliare è molto ampia, avrà cali d'attenzione che possono portare ad una intrusione non segnalata.
La soluzione presentata in questa tesi vuole ovviare a questo inconveniente, innanzitutto in maniera centralizzata, facendo muovere la rete di telecamere autonomamente e fornendole di un software in grado di rintracciare un possibile intruso. Successivamente si presenta una soluzione decentralizzata.
Vista la scarsa documentazione in merito si è cercato di risolvere il problema adattando algoritmi già esistenti che presentano una forte base teorica.
Lo stato dell'arte attuale descrive il problema del patrolling bidimensionale utilizzando uno sciame di robot in grado di esplorare l'intero terreno di ricerca rilevando nel contempo anche una mappa del territorio. Tale soluzione prende il nome di pursuit-evasion game e il paper [1] è preso come punto di partenza.
La tesi offre una possibile soluzione riguardo al pattugliamento bidimensionale adottando una rete di telecamere, i contributi dati sono i seguenti:

1. Estensione della soluzione dal caso monodimensionale [4] al caso bidimensionale.

2. Adattamento del pursuit-evasion game [1] al problema.

Per quanto riguarda il primo punto si vedrà che, mentre nel caso monodimensionale l'evasore viene individuato in in un tempo finito, l'estensione al caso bidimensionale non risulterà possibile.
La seconda soluzione proposta, invece, risolverà il problema adattando la teoria del pursuit-evasion games [1], nel caso di evasore non intelligente (modello Markoviano), creando diversi algoritmi di ricerca (policy). Questi ultimi permetteranno di trovare l'intruso mediamente in un tempo finito.
La migliore tra le policy, individuata attraverso simulazioni effettuate con il metodo di Monte Carlo, verrà impiegata anche nel caso di evasore intelligente.
I risultati evidenzieranno che il miglior algoritmo risulterà essere quello che massimizza la probabilità di trovare l'evasore ad ogni passo, impiegando un tempo medio di cattura intorno ai 6[s], con velocità massima dell'evasore non intelligente pari a 0,6$[\frac{m}{s}]$, spazio di ricerca di $\approx 11m^2$ e due telecamere.
Tale algoritmo sarà poi implementato all'interno del Testbed fornito dalla Videotec[1] e permetterà di osservare la differenza tra risultati reali e quelli sperimentali.

Seguirà ora una breve spiegazione dei capitoli affrontati:
Nella prima sottosezione si spiega l'impossibilità di estendere il caso monodimensionale al caso bidimensionale e successivamente si introduce una breve panoramica sul pursuit-evasion game.
Nel secondo capitolo si introduce il Testbed adottato, descrivendo il modello delle telecamere al fine di introdurre, nel capitolo terzo, l'adattamento del pursuit evasion game al caso delle telecamere per poter risolvere il problema.
Chiaramente si vuole catturare l'evasore nel minor tempo possibile, è quindi indispensabile avere un indice di prestazioni che tenga conto di ciò e che permetta di differenziare le diverse policy. Tale indice viene spiegato nel capitolo

[1] Azienda con sede a Vicenza che si occupa di Videosorveglianza

quarto e affinchè risulti finito le policy devono soddisfare le proprietà di persistenza analizzate nel quinto capitolo.
Si spiega dettagliatamente, nel sesto capitolo, l'algoritmo utilizzato, sfruttando anche algoritmi inerenti alla teoria dei grafi.
Nel settimo capitolo si verifica la persistenza in media delle policy utilizzate e per ognuna di queste si valutano le prestazioni attraverso simulazioni di Monte Carlo illustrate nell'ottavo capitolo.
Infine, si descrivono i possibili sviluppi futuri, primo fra tutti la soluzione del problema decentralizzato.

1.1 Estensione caso monodimensionale

Il problema di videosorvegliare un area monodimensionale al fine di trovare un possibile evasore in un tempo finito è stato ampiamente discusso e risolto nel paper [4]. La soluzione individuata viene impiegata nel controllo di un corridoio: le telecamere vengono messe in serie ed ognuna ispeziona la sua porzione di terreno muovendosi in controfase rispetto alle altre permettendo l'individuazione certa dell'intruso in un tempo finito e non lasciandolo entrare in zone già osservate.

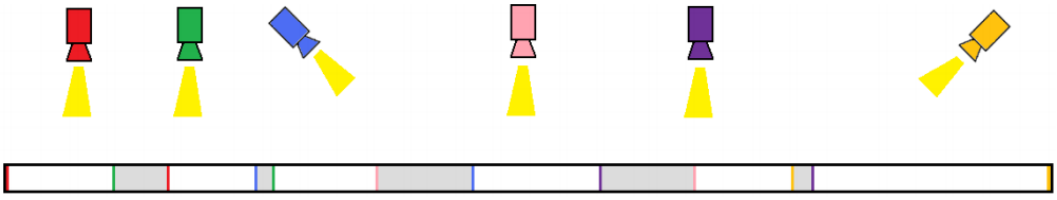

Figura 1: Caso Monodimensionale

Si è cercato di creare la stessa situazione, nella quale l'evasore non percorre zone già controllate, anche al caso bidimensionale.
Si illustra di seguito un esempio con due telecamere:

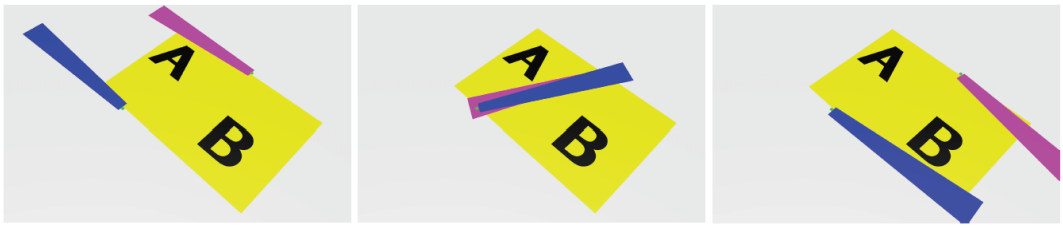

Figura 2: Caso bidimensionale

Nella figura di sinistra si inizia controllando la zona A fino a far incrociare i Field of View delle telecamere (figura centrale) per poi continuare allo stesso modo nella zona B (figura destra). Così facendo l'evasore non può entrare in aree già ispezionate e viene sempre catturato in un tempo finito. Ciò accade anche se si è in presenza di un evasore intelligente.
Tale soluzione, purtroppo non è implementabile nel caso 2D, infatti, la geometria delle zone controllate non rispetta il modello delle telecamere date in dotazione, inoltre, queste ultime, ispezionano un area molto vasta che rende difficile l'individuazione dell'evasore tramite algoritmi di visione.
Si è deciso, quindi, di sviluppare una soluzione che tenesse conto della vera dinamica delle telecamere e che fosse in grado di individuare un intruso all'interno del loro campo visivo.
Questa nuova soluzione basata sul pursuit-evasion games garantisce la cattura dell'evasore, in un tempo finito, con probabilità pari a uno.

1.2 Pursuit-evasion game

Accertati gli ostacoli sorti nel caso sopracitato, si è deciso di intraprendere una nuova strada adattando la teoria del paper [1] al problema.
Il pursuit evasion-game si basa sul problema di **controllare uno sciame di agenti autonomi al fine di catturare uno o più evasori** e può essere utile in numerose situazioni. Esempi tipici sono operazioni di ricerca e soccor-

so, controllo di intrusi all'interno di edifici, missioni di ricerca e cattura, ecc.
In alcuni casi gli evasori cercano attivamente di evitare la cattura (esempio missioni di ricerca e cattura), mentre in altri casi il loro movimento è praticamente randomico (esempio missioni di ricerca e soccorso). Quest'ultimo problema viene spesso chiamato games against nature.
Nel paper [1] per risolvere il pursuit-evasion game è proposto un approccio probabilistico al problema, nel quale si tiene conto del fatto che i sensori installati nei pursuer non portano informazioni precise e che si è a conoscenza solamente di una mappa a priori del terreno da esplorare.
Si dimostra che le greedy policy sono in grado di controllare uno sciame di agenti autonomi alla ricerca di uno o più evasori. Tali policy garantiscono che il **valore atteso** necessario per trovare l'evasore è finito.
Si mostra quindi il comportamento dei pursuer usando greedy policy nel pursuit evasion game:

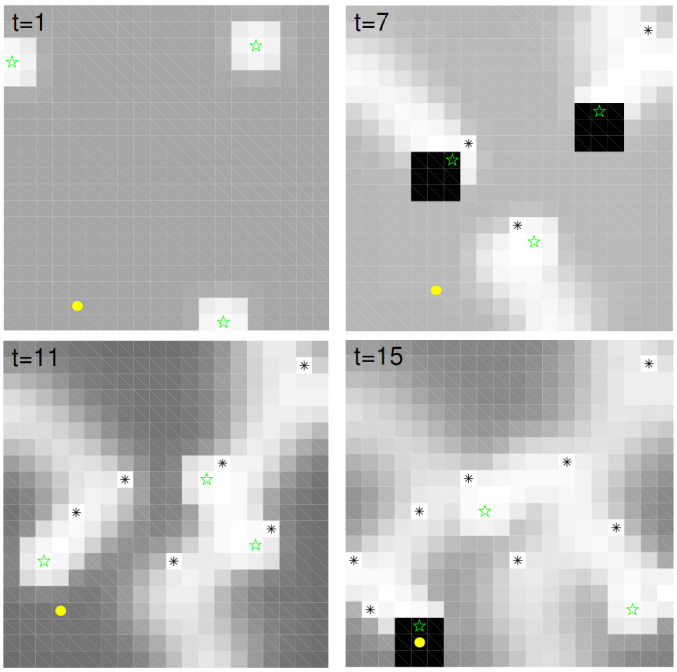

Figura 3: pursuit-evasion game con vincoli nelle greedy policy [1]

In questa figura si mostrano le simulazioni del pursuit evasion game con

$N_c := 400$ celle, $N_p := 3$ pursuer (rappresentati da stelle verdi), l'evasore (rappresentato da un cerchio giallo) e gli ostacoli trovati dai pursuer (raffigurati con asterischi neri).

Ad ogni timestep t i pursuer prendono le misure Y_t attraverso i loro sensori e viene calcolata per ogni cella $x \in \mathcal{X}$ la probabilità di trovare l'evasore nel prossimo passo, dato che, nessuno è stato trovato fino ad ora ($p_e(x, Y_t)$). Questo è possibile perchè i pursuer conoscono il modello Markoviano dell'evasore (non intelligente).

Il colore di ogni cella x sullo sfondo codifica la probabilità $p_e(x, Y_t)$ in scala di grigi: dove i colori sono chiari tale probabilità è bassa, mentre aumenta dove le zone sono più scure.

Iniziando da t=1 si vede la posizione dei pursuer, dell'evasore e la probabilità $p_e(x, Y_t)$, inizialmente distribuita in maniera uniforme. In $t = 7$ si può vedere un alto valore di probabilità $p_e(x, Y_t)$ vicino uno dei pursuer, anche se l'evasore è molto lontano. Questo è dovuto ai falsi positivi dati dai sensori probabilistici.

La velocità del pursuer e dell'evasore sono compatibili, in ogni timestep, entrambi si muovono di una cella e ad ogni passo la mappa degli ostacoli è aggiornata. Alla fine in $t = 15$ il pursuer cattura l'evasore.

Il primo quesito che il paper cerca di risolvere è: 'Come devono essere fatti i movimenti del pursuer per catturare l'evasore in un tempo finito con probabilità uno?' A tale scopo vengono costruite delle regole di movimento chiamate policy persistenti.

In questa tesi si adotta la teoria del paper sopracitato per risovere il problema del patrolling su un area bidimensionale, inoltre, si cercano policy sub-ottime in grado di minimizzare il valore atteso del tempo necessario a trovare l'evasore. Fatto questo si utilizza un evasore intelligente che evita la cattura.

Prima di entrare nei dettagli della soluzione si introduce il Testbed, il modello della telecamera e la dinamica di queste ultime.

2 Testbed

Il testbed viene fornito dalla Videotec ed è montato nel dipartimento Ifa della Swiss Federal Institute of technology of Zurich (ETH). Esso comprende:

- due telecamere PTZ installate a muro alla stessa altezza (H=2.5m), distanti 4.55m e comandate da PC
- una macchina radiocomandata che funge da evasore
- un foglio bianco (3.96m×2.73m) che delimita il terreno di ricerca delle telecamere, unico spazio in cui l'evasore può muoversi

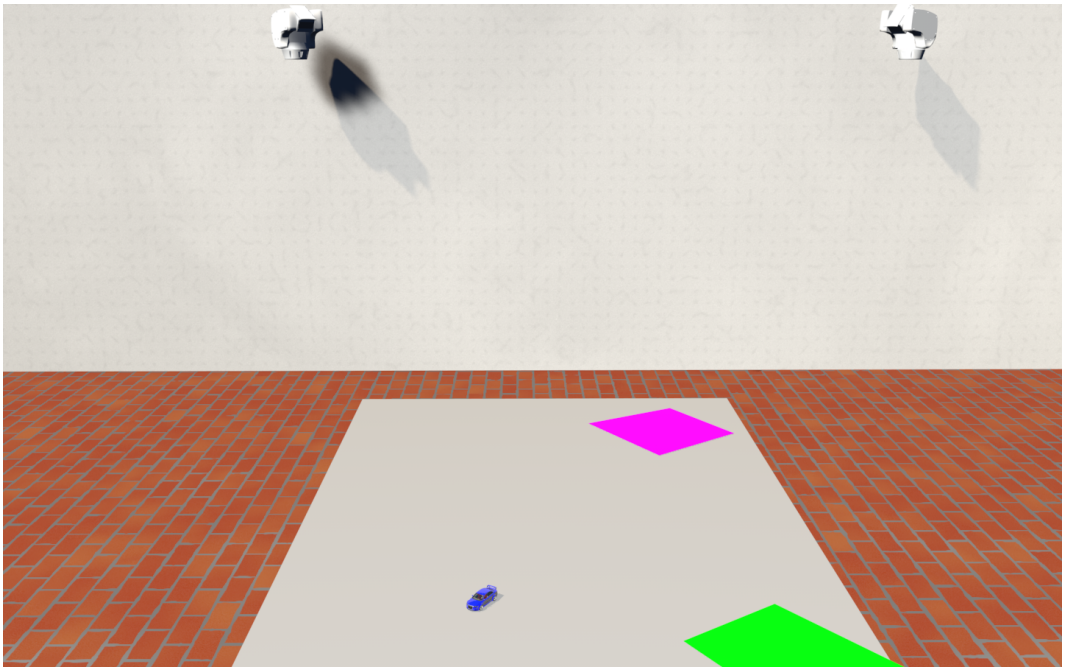

Figura 4: Testbed con rappresentazione Field Of View telecamere.

2.1 Telecamere (Pursuers)

Le telecamere utilizzate vengono definite PTZ in quanto sono in grado di ruotare lungo gli assi presenti in figura descrivendo diversi angoli di Pan (Θ)

e di Tilt (Ψ), inoltre, anche il valore dello Zoom (ζ) è variabile.

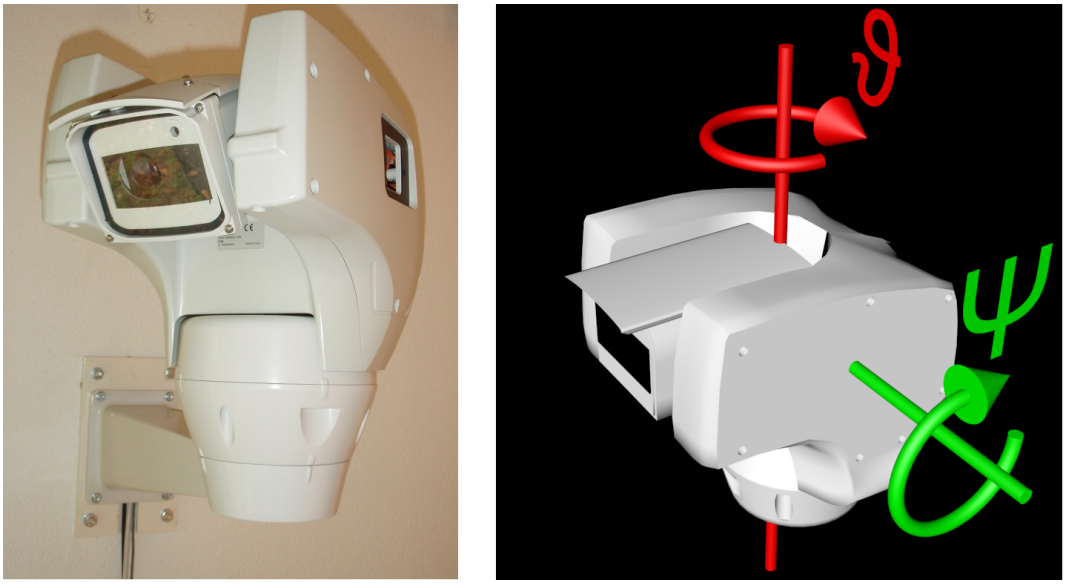

Figura 5: Telecamere 'PTZ' modello ULISSE

Questo tipo di telecamere, oltre che essere molto utile nel caso di tracking di un oggetto, si presta bene anche per la soluzione del problema, infatti, uno zoom variabile permette di mantenere la stessa risoluzione ad ogni istante di tempo (in base ai limiti fisici della telecamera) e quindi l'algoritmo di detection ha una maggiore robustezza nel trovare l'evasore all'interno dei Field Of View. Queste telecamere sono connesse attraverso un cavo analogico e trasmettono immagini al computer. Quest'ultimo è equipaggiato con una scheda video Matrox 'MeteorII' che digitalizza le immagini ricevute. I comandi per far muovere le telecamere sono mandati attraverso un cavo USB bidirezionale, che trasmette anche eventuali dati richiesti alla telecamera.
Il problema principale è richiedere l'attuale posizione della telecamera all'encoder attraverso la connessione (USB). Purtroppo, questa operazione è temporalmente molto inefficente, visto che le telecamere non sono costruite per applicazioni ad alta velocità, infatti il tempo richiesto utilizza il 40% del tempo di campionamento, il che è inaccettabile. Per questo motivo nell'implemen-

tazione dell' algoritmo si conoscono in ogni istante i valori di Θ, Ψ e ζ senza richiederli all'encoder e utilizzando il modello della telecamera si calcolano, abbastanza precisamente, i valori dei quattro punti che delimitano la proiezione del piano immagine (FOV) sul terreno di ricerca.
E', quindi, di fondamentale importanza avere un buon modello per la telecamera, anche molto utile nelle simulazioni.

2.1.1 Modello telecamera

Il modello utilizzato per la telecamera è il classico modello pinhole che permette di calcolare la posizione nell'immagine di un oggetto data la sua posizione nel mondo.

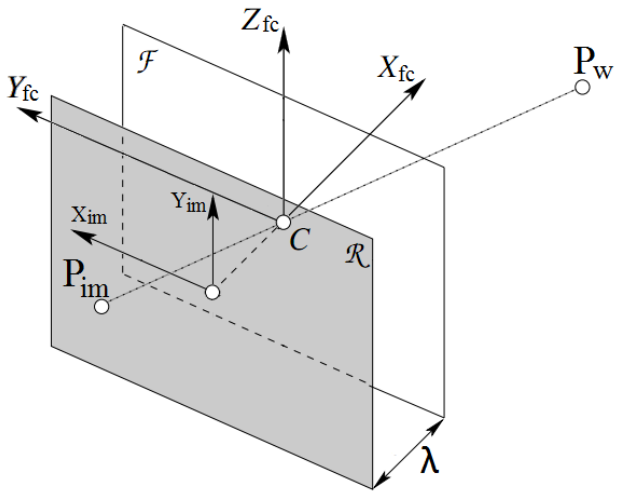

Figura 6: Modello pinhole telecamera

Esso approssima bene il comportamento della telecamera, a patto di avere una buona misura dei parametri intrinseci ed estrinseci (calibrazione). Il modello si usa non solo nell'algoritmo scritto in Matlab, ma anche nell'implementazione in C++ al fine di avere un comportamento analogo tra simulazioni e realtà.

Come è solito fare si distinguono i parametri intrinseci ed estrinseci. I parametri intrinseci sono necessari per collegare le coordinate in pixel di un punto nell'immagine con le corrispondenti coordinate nel sistema di riferimento della telecamera. I parametri estrinseci definiscono la posizione e l'orientazione del sistema di riferimento della telecamera rispetto al sistema di riferimento mondo. Siccome si utilizza il modello pinhole per la telecamera gli unici parametri da considerare sono la lunghezza focale λ (quindi la distanza tra il centro ottico e il sensore CCD) e la dimensione s_x e s_y del sensore CCD.
La lunghezza focale influenza il valore dello zoom, per questo motivo si userà in seguito ζ (valore dello zoom) al posto di λ e viceversa.
Per i parametri intrinseci la dimensione del sensore CCD è disponibile nel manuale di riferimento, mentre la lunghezza focale, λ, è inizialmente stimata attraverso il toolbox di Matlab.
Visto che il valore dello zoom viene cambiato quasi ad ogni timestep, al fine di avere la stessa area nel FOV, è necessario stimare diversi valori di λ per differenti livelli di zoom. Per aumentare l'efficienza questo viene fatto prendendo circa 15 immagini per ogni livello di zoom di un oggetto avente dimensioni note ponendolo in vari posti all'interno del field of view.
Infine, il valore di λ è quello che minimizza, nel senso dei minimi quadrati, la differenza tra la distanza vera e quella calcolata. Tale processo fornisce un legame tra alcuni valori di zoom nel range da [5000-16000] e i valori λ.
Per calcolare la lunghezza focale dato un livello di zoom generico viene utilizzata l'interpolazione lineare, questo valore viene calcolato al fine di avere la stessa area all'interno dei Field of View ($0.5m^2$), portando ad avere la stessa risoluzione ad ogni passo (dove è possibile).
I parametri estrinseci sono Θ,Ψ, H, D, x_{off}, y_{off} e z_{off} e vengono rappresentati in Figura (7). Le stime dei parametri estrinseci fissi (quindi senza Θ e Ψ) sono fatte attraverso il metodo dei minimi quadrati e utilizzano il camera calibration toolbox di Matlab implementato da Bouguet (2008).

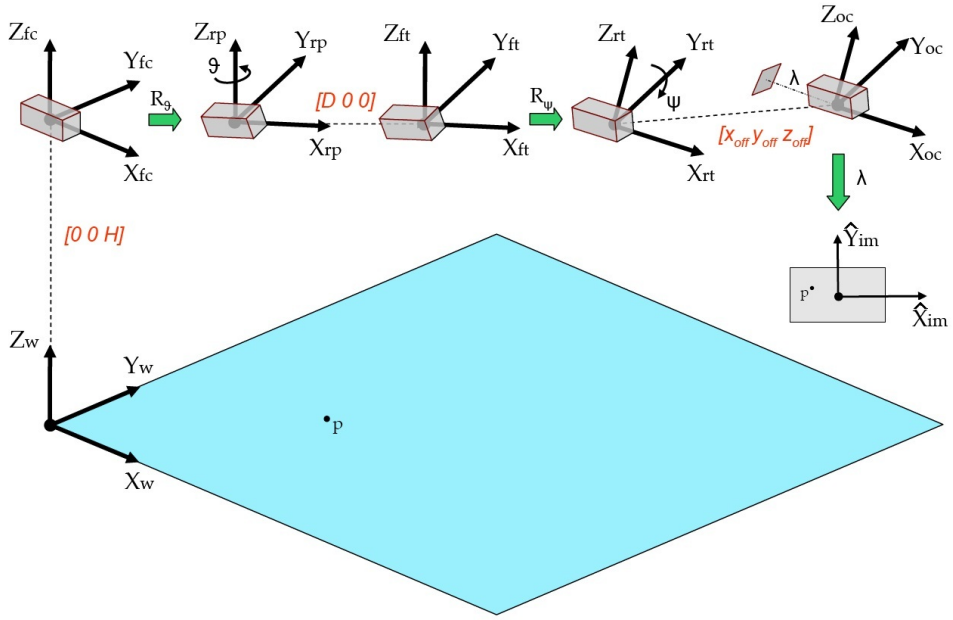

Figura 7: Geometria sistemi di riferimento.

Questa figura illustra la geometria del setup della camera. Per poter convertire un punto nel piano immagine date le coordinate in pixel $[\hat{x}_{im}, \hat{y}_{im}]$[2] in un punto nel sistema di riferimento mondo $[x_w, y_w, z_w]$ si necessita di un certo numero di trasformazioni. Notiamo che generalmente non si può ottenere la posizione di un oggetto nel sistema di riferimento mondo date le coordinate in pixel utilizzando solamente una telecamera, ma visto che il target si muove su un piano orizzontale si può imporre $z_w = 0$ riuscendo in questo modo a calcolarlo.

Come primo passo si converte il punto da coordinate pixel a coordinate nel piano immagine (pedice *im*) e quindi alle coordinate del centro ottico (pedice *oc*). La forma di queste relazioni dipende dal modello della telecamera utilizzato. Si utilizza il modello pinhole, il quale assume che un punto 3D è proiettato

[2]In questo contesto, il cappello sopra le coordinate simboleggia le coordinate in pixel.

nel piano immagine di un ideale[3] telecamera a buco di spillo. Questo modello porta le seguenti relazioni da pixel a coordinate nell'immagine.

$$y_{im} = s_x \cdot \hat{x}_{im}, \quad z_{im} = s_y \cdot \hat{y}_{im} \tag{1}$$

e dalle coordinate immagine alle coordinate del centro ottico

$$x_{oc} = -\lambda, \quad y_{oc} = y_{im}, \quad z_{oc} = z_{im} \tag{2}$$

Dove s_x, s_y sono le dimensioni di un pixel nel sensore CCD[4] rispettivamente nelle direzioni x and y, λ denota la lunghezza focale e $\hat{x}_{im}$ e $\hat{y}_{im}$ sono le coordinate in pixel. Si nota che in (2) è stato usato $x_{oc} = -\lambda$ in quanto un punto nel piano immagine è per definizione distante λ dal centro ottico. Si nota ancora che applicando (1) bisogna prestare attenzione alla scala dell'immagine.
Il secondo passo è di trasformare un punto da coordinate *oc* a coordinate mondo. La prima traslazione di $[\, x_{off}, y_{off}, z_{off}\,]$ converte i punti in coordinate di rotazione tilt (rt) e tiene conto del fatto che il centro ottico della telecamera non è nel centro del sistema di riferimento del tilt. Segue la rotazione di Ψ intorno Y_{rt} e una traslazione di D che è necessaria se l'asse di rotazione di pan e tilt non passa attraverso un centro comune. Infine la rotazione del pan di Θ attorno Z_{rp} e la traslazione di altezza di H dal pavimento in direzione Z_w porta il punto in coordinate mondo. La trasformazione di un punto $p_{oc} = [x_{oc}\ y_{oc}\ z_{oc}]$ in coordinate mondo $p_w = [x_w\ y_w\ z_w]$ è quindi data da:

[3] l'apertura della telecamera è un buco e non sono utilizzate lenti
[4] è Il microchip responsabile di convertire la luce in ingresso in un immagine digitale

$$\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ H \end{bmatrix} + R_\theta \left(\begin{bmatrix} D \\ 0 \\ 0 \end{bmatrix} + R_\psi \left(\begin{bmatrix} x_{off} \\ y_{off} \\ z_{off} \end{bmatrix} + \begin{bmatrix} x_{oc} \\ y_{oc} \\ z_{oc} \end{bmatrix} \right) \right) \tag{3}$$

Dove i parametri sono definiti come descritto sopra e le matrici di rotazione coinvolte sono date da:

$$R_\theta = \begin{bmatrix} c_\theta & -s_\theta & 0 \\ s_\theta & c_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; R_\psi = \begin{bmatrix} c_\psi & 0 & s_\psi \\ 0 & 1 & 0 \\ -s_\psi & 0 & c_\psi \end{bmatrix}.$$

Usando la parola pursuer, invece che telecamera, si intende che il pursuer è la proiezione del piano immagine (FOV) della telecamera nel terreno di ricerca.

2.1.2 Vincoli telecamera

Si hanno i seguenti vincoli nelle telecamere utilizzate:

$$\begin{gathered} \Theta_{min} \leq \Theta \leq \Theta_{max} \\ \Psi_{min} \leq \Psi \leq \Psi_{max} \\ \zeta_{min} \leq \zeta \leq \zeta_{max} \\ |\Theta_{k+1} - \Theta_k| \leq \Delta\Theta_{max} \\ |\Psi_{k+1} - \Psi_k| \leq \Delta\Psi_{max} \\ |\zeta_{k+1} - \zeta_k| \leq \Delta\zeta_{max} \end{gathered} \tag{4}$$

Dove k denota il tempo corrente e $k+1$ il tempo dopo un timestep. I vincoli sull'angolo di pan e di tilt derivano dal al fatto che la telecamera non può ruotare oltre un giro completo nella stessa direzione. Il valore dello zoom è limitato fisicamente tra i valori ζ_{min}, ζ_{max}. I vincoli sulla dinamica $\Delta\Theta_{max}$, $\Delta\Psi_{max}$ e $\Delta\zeta_{max}$ sono chiaramente dipendenti dal tempo di campionamento,

più tempo si ha a disposizione più la telecamera può muoversi. Per l'imprecisa posizione di misura risulta difficile stimare accuratamente valori per questi bound, ma nelle tabelle 1 and 2 si forniscono valori approssimativi del tempo necessario per muoversi di un certo angolo. Si nota che i valori $\Delta\Psi$ e $\Delta\Theta$ sono identici e per questo verranno raggruppati. Come si vede la velocità media $\frac{\Delta\Psi}{\Delta t}$, $\frac{\Delta\Theta}{\Delta t}$ e $\frac{\Delta\zeta}{\Delta t}$ cresce con l'aumentare del passo $\Delta\Psi$, $\Delta\Theta$ e $\Delta\zeta$. Questo implica che se si utilizza un tempo lungo di campionamento è possibile muovere la telecamera più velocemente in media. Si è scelto, dopo alcune prove, il tempo di campionamento più piccolo possibile $250ms$. In questo tempo si riesce a muovere la telecamera di $1,65°$ per l'angolo di pan e di tilt (anche contemporaneamente) al fine di catturare l'evasore avendo sempre un area del FOV di $0{,}5m^2$.

$\Delta\Psi, \Delta\Theta$	$0.25°$	$0.5°$	$1°$	$1.5°$	$\mathbf{1.65°}$	$2°$	$3°$	$5°$
Δt	$100ms$	$140ms$	$200ms$	$240ms$	**250ms**	$270ms$	$340ms$	$370ms$
$\frac{\Delta\Psi}{\Delta t}$, $\frac{\Delta\Theta}{\Delta t}$	$2.5°/s$	$3.6°/s$	$5°/s$	$6.25°/s$	$\mathbf{6.6°/s}$	$7.4°/s$	$8.8°/s$	$13.5°/s$

Tabella 1: Tempo Δt necessario per muovere $\Delta\Psi, \Delta\Theta$ e il rapporto $\frac{\Delta\Psi}{\Delta t}$, $\frac{\Delta\Theta}{\Delta t}$.

$\Delta\zeta$	250	500	**540**	750	1000	1500	2000
Δt	$190ms$	$240ms$	**250ms**	$300ms$	$300ms$	$320ms$	$360ms$
$\frac{\Delta\zeta}{\Delta t}$	$1300/s$	$2100/s$	$\mathbf{2165/s}$	$2500/s$	$3000/s$	$4700/s$	$5500/s$

Tabella 2: Tempo Δt necessario per muovere $\Delta\zeta$ e il rapporto $\frac{\Delta\zeta}{\Delta t}$.

2.2 Car (Evasore)

L'evasore è rappresentato da una macchina blu ed ha le seguenti caratteristiche:

- E' guidato da un operatore umano tramite un radiocomando
- Dimensioni: lunghezza=10cm, larghezza=5cm
- Può essere individuata solo nel foglio bianco (Terreno di ricerca)

- Può raggiungere la velocità di 5m/s

Figura 8: Kyosho dNano RC cars

Si suppone, inizialmente, che l'evasore è intelligente, sebbene guidato da un operatore umano (la macchina è guidata non guardando i movimenti della telecamera).
Risulta evidente come i pursuer non possano conoscere la natura dell'evasore (una macchina, un uomo, una moto, ecc.) e nemmeno le sue intenzioni, per tale motivo il modello che si utilizza inizialmente è Markoviano e in seguito una volta identificato l'intruso verrà raffinato. Catturato l'evasore si utilizza, ad esempio, il modello dell'uniciclo per poterne fare il tracking.

2.2.1 Rilevazione immagine e bilanciamento colori

La figura 9 mostra l'immagine acquisita della telecamera. Il colore blu della macchina non è ben visibile, il foglio sul pavimento che dovrebbe essere bianco risulta invece sul rosso. Questo è dovuto all'irradiazione della luce interna e alla telecamera stessa. Per ottenenere una miglior robustezza nell'algoritmo di detection, in seguito ai cambiamenti nell'illuminazione, occorre eseguire un bilanciamento del bianco nell'immagine, come mostrato nella Figura 10.

Figura 9: Immagine grezza ottenuta dalla telecamera.

Figura 10: Bilanciamento del bianco.

L'idea di tale bilanciamento è quella di prendere come riferimento un'area nell'immagine di colore noto[5], calcolando la differenza media tra i colori di tale area con il suo vero colore e correggendo l'intera immagine con tale offset. Dato che i colori reali dell'area sotto investigazione sono noti e che l'intera immagine ha lo stesso offset, l'immagine bilanciata presenterà i colori reali di tutti gli oggetti. Questa implementazione può essere fatta in diversi modi, in seguito ne verranno spiegati due.

Prima di tutto, in entrambi gli approcci, le immagini vengono convertite in scala di grigi[6]. Successivamente si esegue un'operazione di sogliatura nell'immagine in scala di grigi che porta ad avere un immagine in bianco e nero, con pixel bianchi laddove il valore in scala di grigi supera la soglia thr_{white}. Tale soglia è presa grande abbastanza in modo che tutti i pixel bianchi risultati possano essere considerati facenti parte del foglio bianco, quindi, si calcola la media dei valori **R**, **G** e **B** di tutti questi pixel. Tali valori medi $\bar{r}_{white}$, $\bar{g}_{white}$ e $\bar{b}_{white}$ sarebbero uguali al massimo valore se nell'immagine il foglio fosse completamente bianco. Chiaramente non è questo il caso. Si corregge quindi la

[5] In questo contesto, il vero colore di un oggetto si riferisce ai valori rosso-verde-blu (**RGB**) che approssimano lo spettro della luce riflessa dell'oggetto nel modo migliore, illuminandolo da una perfetta sorgente di luce bianca.

[6] La scala di grigi di un immagine può essere ottenuta calcolando la media aritmetica dei tre colori.

figura in modo da rendere il foglio completamente bianco, per bilanciare l'immagine i valori **R**,**G** e **B** devono assumere lo stesso valore in tale foglio.
Per farlo ci sono due modi possibili. Il primo adotta una trasformazione che conserva la luminosità[7], adottando le seguenti operazioni per ogni pixel px_i.

$$\begin{aligned} px_i.r &= px_i.r - (\bar{r}_{white} - \frac{\bar{r}_{white} + \bar{g}_{white} + \bar{b}_{white}}{3}) \\ px_i.g &= px_i.g - (\bar{g}_{white} - \frac{\bar{r}_{white} + \bar{g}_{white} + \bar{b}_{white}}{3}) \\ px_i.b &= px_i.b - (\bar{b}_{white} - \frac{\bar{r}_{white} + \bar{g}_{white} + \bar{b}_{white}}{3}). \end{aligned} \tag{5}$$

Dove $px_i.r$, $px_i.g$ e $px_i.b$ denotano i valori **RGB** del pixel i. Questa operazione fa mantenere la stessa luminosità nei pixel dopo la trasformazione, ma se i pixel hanno valori troppo vicini allo zero, in qualche canale si corre il rischio che la sottrazione porti ad avere un valore negativo e tale valore non essendo accettabile verrà portato a zero. Purtroppo, questo porta ad avere una 'assimmetria' che provoca la distorsione dei colori nell'imagine.
Il secondo approccio è :

$$\begin{aligned} px_i.r &= px_i.r - (\bar{r}_{white} - \max(\bar{r}_{white}, \bar{g}_{white}, \bar{b}_{white})) \\ px_i.g &= px_i.g - (\bar{g}_{white} - \max(\bar{r}_{white}, \bar{g}_{white}, \bar{b}_{white})) \\ px_i.b &= px_i.b - (\bar{b}_{white} - \max(\bar{r}_{white}, \bar{g}_{white}, \bar{b}_{white})). \end{aligned} \tag{6}$$

Questo permette di non ottenere valori negativi nei singoli pixel. Tale trasformazione può, invece, portare il problema opposto, cioè alla saturazione di un canale. Questo si evita perchè nessun canale ha valori inizialmente elevati, pertanto si utilizza (6) al posto di (5).

Il risultato del bilanciamento della figura 9 è mostrato nell'immagine 10.

[7] la media aritmetica del colore del canale di un pixel resta la stessa.

2.2.2 Car Detection

L'algoritmo di visione che si utilizza per identificare la macchina è la blob detection. Regioni connesse, che risultano particolarmente scure, sono canditate per essere **blob**. Tali blob devono superare una serie di criteri per essere considerati i blob relativi alle macchine.
Il primo passo è di utilizzare un ulteriore sogliatura sull'immagine che genera una figura in bianco e nero, dove le regioni bianche sono considerate quelle meno luminose. Per far ciò si converte l'immagine (Fig.10) in scala di grigi (Fig.11) e l'operazione di sogliatura porta ad avere la figura 12.

Figura 11: Immagine in scala di grigi

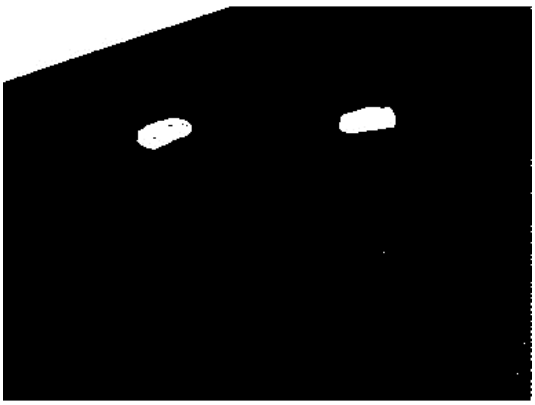

Figura 12: Sogliatura immagine

Fatto ciò l'algoritmo di blob detection trova tutte le regioni connesse, di colore bianco, e le inserisce dentro una lista di candidate che dovranno soddisfare i seguenti criteri al fine di essere i blob relativi alla macchina.
Il primo criterio è la dimensione del blob. Per questo vengono calcolati i limiti massimo e minimo sulla dimensione dei blob (in pixel). Tali limiti dipendono, chiaramente, dal livello di zoom e non possono essere troppo vicini, in quanto devono tener conto anche dell'orientazione della macchina che viene riconosciuta anche se non si trova interamente all'interno del FOV. Il calcolo di tali limiti è dato dalle seguenti:

$$lowerBound = A_{car,meter}/f^2_{m/px} \cdot (1 - tol) \cdot tol_{cutoff} \quad (7)$$

$$upperBound = A_{car,meter}/f^2_{m/px} \cdot (1 + tol) \quad (8)$$

Dove $A_{car,meter}$ è l'area, in metri, della macchina come verrebbe vista dall'alto. $f^2_{m/px}$ è la misura per la risoluzione dell'immagine e l'area fisica vista da un pixel nel sensore CCD della telecamera. In questo parametro è incluso l'aggiustamento per i diversi livelli di zoom. $f_{m/px}$ è calcolato semplicemente tenendo in considerazione la posizione rispetto al sistema di riferimento mondo dei due pixel più esterni visti dalla telecamera e quindi costruendo il rapporto tra distanza mondo con distanza tra i pixel. Infine, tol_{cutoff} compensa il fatto che la macchina a volte non è interamente all'interno dell'immagine e *tol* è un parametro di aggiustamento.

Il secondo criterio è la compattezza del blob, che in questo contesto si riferisce al rapporto tra l'area del blob e l'area dell'ellisse che lo circonda[8]. Per soddisfare il criterio di compattezza il rapporto tra queste aree deve essere più grande di una certa soglia, in modo da eliminare possibili candidati a blob con geometrie inaspettate.

Infine, la posizione del centro del blob rispetto al sistema di riferimento mondo deve essere all'interno di una certa zona chiamata 'field of detection' che praticamente consiste nel foglio bianco nel quale l'evasore si muove. Incorporando tali vincoli ai blob si evita di trovare blob esterni a tale area.

La figura 13 mostra i risultati delle operazioni descritte in questa parte.

[8] L'ellisse che avvolge il blob ha l'area più piccola possibile al fine di contenere tutti i pixel del blob

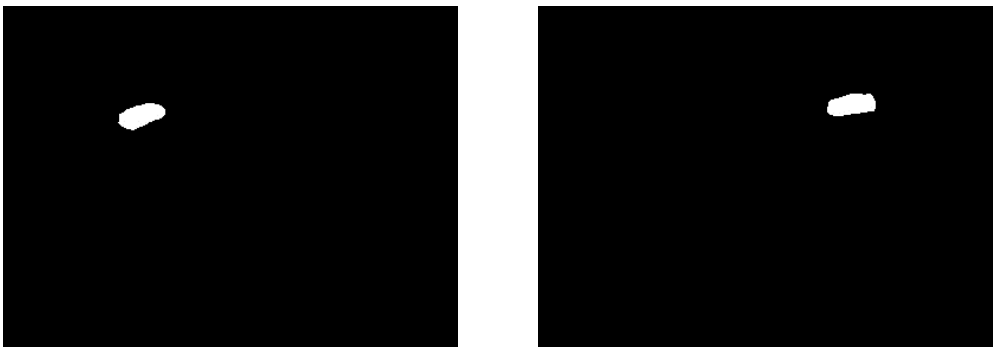

Figura 13: Blobs Macchine

2.2.3 Conversion to HSV

Nel caso in cui fossero presenti più evasori per poterli distinguere si utilizza la color detection basata sul concetto di **HSV** color. Sebbene nel caso in esame sia presente un unico evasore, questa sezione spiegherà per quale motivo si è scelta la macchina di colore blu piuttosto che quella di colore verde.
Per la color detection è vantaggioso che l'immagine 10 sia trasformata nello spazio di colori **HSV**.
In [5] viene proposto per la prima volta il modello di colore **HSV**. Questo consiste in una rappresentazione in coordinate cilindrinche dello spazio di colori **RGB**. I singoli componenti sono chiamati hue (tonalità), saturation (saturazione) e value (luminosità). Per saturazione si intende l'intensità e la purezza del colore, mentre la luminosità (valore) è un'indicazione della sua brillantezza. Ovviamente la tonalità indica il colore stesso.
Si descrive nell' equazione (9) la trasformazione fra spazio **RGB** in **HSV**.

$$
\begin{aligned}
M &= \max(R, G, B) \\
m &= \min(R, G, B) \\
C &= M - m \\
H' &= \begin{cases} \text{undefined}, & \text{if } C = 0 \\ \frac{G-B}{C} \mod 6, & \text{if } M = R \\ \frac{B-R}{C} + 2, & \text{if } M = G \\ \frac{R-G}{C} + 4, & \text{if } M = B \end{cases} \\
H &= 60^\circ \cdot H' \\
S &= \begin{cases} 0, & \text{if } C = 0 \\ \frac{C}{M}, & \text{otherwise} \end{cases} \\
V &= M
\end{aligned}
\tag{9}
$$

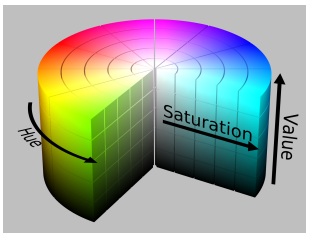

Figura 14: cilindro **HSV** [6]

In quest'ultima figura è rappresentato lo spazio **HSV** come un cilindro. Le motivazioni originali per utilizzare un sistema di color alternativo erano di riarrangiare la geometria **RGB** al fine di avere una percezione più rilevante rispetto a quella cartesiana [6].
L'approccio **HSV** è vantaggioso per questo schema di detection visto il modo semplice di specificare una regione colorata.

La figura 15 mostra la decomposizione della figura 10 nelle componenti **H**,**S**,**V**. Si nota che sussistono molti punti nella parte **H**: questo è dovuto al fatto che il valore di hue è sensibile ai cambiamenti dei valori **RGB** per colori con bassa saturazione (vicini all'asse del cilindro in Figura 14). Inoltre, si nota che la macchina blu ha una saturazione più alta rispetto a quella verde, in questo modo il suo valore di hue è meno sensibile alle misure rumorose che possono essere viste nella parte **H**. Per questo motivo la macchina blu è più facile da identificare ed è perciò stata scelta.

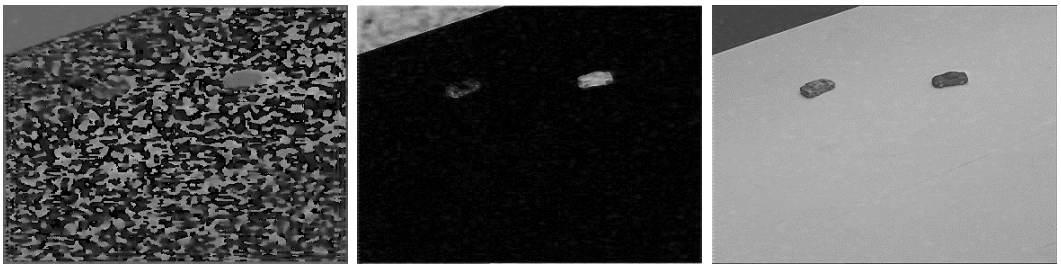

Figura 15: Parti **H**, **S** e **V** della Figura 10

3 Soluzione Problema attraverso il Pursuit-evasion game

Ora che si è descritto il Testbed si adatterà il pursuit-evasion game al problema del Patrolling bidimensionale introducendo la modellizzazione del problema e le variabili più importanti.

3.1 Modellizzazione

Il foglio bianco presentato nella sezione (2) è il terreno di ricerca nel quale l'evasore è libero di muoversi, cioè dove i pursuer devono effettuare la loro ispezione.
Tale piano viene diviso in N_c celle quadrate x, identificate attraverso righe (R$\in \mathbb{N}$) e colonne (C$\in \mathbb{N}$), $x \in \mathcal{X}= \{(m,n) : m \in \{1,...,R\} \vee n \in \{1,...,C\}$ dove $R \times$C$= N_c\}$.

Tali celle hanno la dimensione di 5cm, in questo modo si ottiene un terreno di ricerca simile a quello nella sezione (1.2) con la sola differenza che, in questo caso, non sono presenti ostacoli.
Per semplificare ulteriormente il problema e introdurre la dinamica degli N_p pursuers e dell'evasore si è deciso di discretizzare il tempo di gioco $\mathcal{T}=\{1,2,..\}$ dando un **timestep di 250 ms**.
Si inizia creando un modello dinamico per le telecamere, che possono muoversi di 1.65° sia per l'angolo di Pan, $\Theta=[\theta_1, \theta_2, ...\theta_{N_p}]^T \in \mathbb{R}^{N_p}$, che per l'angolo di tilt, $\Psi=[\psi_1, \psi_2, ...\psi_{N_p}]^T \in \mathbb{R}^{N_p}$ ad ogni timestep si ha:

$$\theta_i(t+1) = \theta_i(t) \pm 1.65^\circ \quad 0^\circ \leq \theta_i \leq 90^\circ \tag{10}$$

$$\psi_i(t+1) = \psi_i(t) \pm 1.65^\circ \quad 20^\circ \leq \psi_i \leq 90^\circ \tag{11}$$

Quindi la telecamera 1 partendo da qualsiasi posizione θ_1(t), ψ_1(t) può muoversi al massimo in otto posizioni (adiacenti) in un timestep, date dall'unione degli angoli precedenti.
In base a questi movimenti, creati da una legge di controllo $\bar{g}$, si ha la proiezione del piano immagine sul terreno di ricerca (FOV), il quale risulta essere un trapezoide. L'area di questa figura varia a seconda del livello di zoom adottato e dove la telecamera lo permette tale superficie deve essere costantemente di $0.5m^2$, visto che i limiti dello Zoom sono $\zeta \in [5000, 16000]$.
Durante la trattazione della tesi si utilizzerà volutamente la parola telecamera come sinonimo di pursuer in quanto, questi ultimi, sono identificati con la proiezione del piano immagine delle singole telecamere sul terreno di ricerca e indicati con $\mathbf{v}=[v_1\ v_2...\ v_{N_p}]^T \in \mathcal{V}$ (nel caso in esame $N_p = 2$) dove $v_1(t) \in \mathcal{X}^{|C_1|}$ con $C_1=\{$x $\subset \mathcal{X}$ contenute o parzialmente contenute all'interno del FOV v_1 al tempo $\mathbf{t}\}$.

Ognuno dei pursuer prende le misure $\mathbf{y}(t)=\big[y_1(t)\ y_2(t)\ ...y_{N_p}(t)\big]^T \in \mathcal{Y}$ dove $y_1(t) \in \mathbb{R}^{|C_1|}$ con $y_1(t) = [y_{(1,1)}, ..., y_{(1,C_1)}]^T$ ogni elemento di tale vettore corrisponde ad una casella $\in C_1$ e assume i seguenti valori se l'evasore non è contenuto all'interno:

$$y_{(i,j)} = \begin{cases} 0, & \text{se } y_{(i,j)} \cap v_i(\text{t}) = y_{(i,j)} \\ \frac{(Area\ Casella - Area\ interna)}{Area\ Casella}, & \text{se } y_{(i,j)} \cap v_i(\text{t}) \neq \oslash \\ 1, & \text{se } y_{(i,j)} \cap v_i(\text{t}) = \oslash \end{cases} \tag{12}$$

Inoltre, si denoterà con $Y_t \in \mathcal{Y}^*$ la sequenza di misure $\{\mathbf{y}(1), \mathbf{y}(2), .., \mathbf{y}(t)\}$.
Per quanto rigurarda l'evasore non conoscendo nulla sulla sua natura si adotta il modello Markoviano, quest'ultimo viene usato anche nel paper [1].
I parametri necessari per descrivere i movimenti dell'evasore sono:

- il numero di celle adiacenti alla posizione attuale dell'intruso ($|A(x)| = \{8, 24, 48, 80\}$, tra le quali quest'ultimo può spostarsi in un timestep
- la probabilità ρ di saltare in celle adiacenti

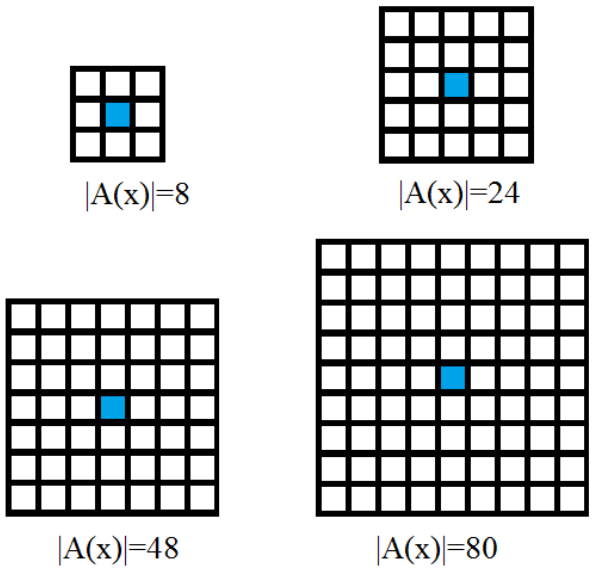

Figura 16: celle adiacenti al variare di $|A(x)|$

Con $|A(x)|$ si identifica automaticamente anche la velocità dell'evasore, infatti, a parità di tempo, avere molte caselle adiacenti significa avere una maggior velocità.
Pertanto si ha che $\rho \in [0, \frac{1}{|A(x)|})$, mentre la probabilità che l'evasore resti fermo nella stessa posizione è dato in ogni istante di tempo da 1-$\rho|A(x)|$. E' quindi ovvio che l'evasore tende a stare più fermo se ha meno adiacenti, come accade ad esempio negli angoli o nei lati dell'intero terreno di ricerca.

L'obiettivo del problema, come nel pursuit-evasion game, è quello di trovare l'evasore all'interno del terreno di ricerca mediamente nel minor tempo possibile. Per farlo si introduce una variabile aleatoria $\mathbf{T}^*$ che indica la probabilità di trovare l'evasore ad ogni istante di tempo t $\in \mathcal{T}$.
Si vorrà pertanto trovare una legge di controllo, $\bar{g}$, che minimizzi il funzionale

di costo $E[T^*]$. In ogni istante di tempo $t \in \mathcal{T}$ è possibile eseguire un'azione di controllo $\mathbf{u}(t) = [v_1(t)\ v_2(t)...v_{N_p}(t)]^T \in \mathcal{U}$, che è funzione delle misure precedenti al tempo t e che influenzerà la posizione dei pursuer al tempo $\tau \geq t$.
Si definisce pursuit policy la funzione che, in base alle misurazioni prese fino a t, decide quale azione di controllo intraprendere al passo successivo $\bar{g}$:$\mathcal{Y}^* \rightarrow \mathcal{U}$ i.e:

$$g(Y_t) = u(t+1) \tag{13}$$

Le decisioni prese in ogni istante di tempo **t** dalle policy $\bar{g} \in \Gamma$, al fine di minimizzare il funzionale di costo, sono basate su **stime** riguardanti la casella in cui può trovarsi l'evasore al tempo **t+1**, date le misure fino a **t** $(p_e(x, Y_t))$.
L'insieme Γ che contiene le policy persistenti o persistenti in media è costruito in modo da garantire che il funzionale assuma valore finito.
Per quanto spiegato, il problema trova una formulazione evidente nel modello di stima Bayesiana.

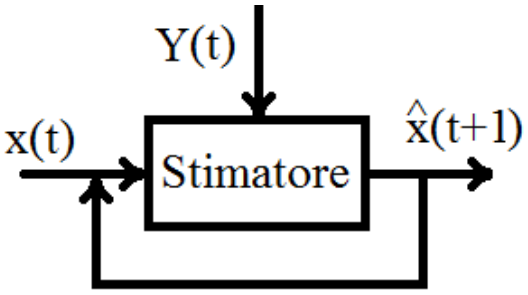

Figura 17: stimatore bayesiano

Il modello viene risolto iterativamente eseguendo i seguenti passi ad ogni timestep e ciò si presta ad essere risolto tramite la **programmazione dinamica**.

1. Passo di Misura
2. Passo di Predizione

3. Passo Decisionale

Dato lo stato del sistema $\mathbf{x} \in \mathbb{R}^{N_c}$, vettore aleatorio che denota la probabilità $p_e(x, Y_{t-1})$ su ogni cella x del terreno di ricerca, si esegue il passo di misura che consiste nel prelevare tutte le misure Y_t (la cui struttura dati corrisponderà ad una matrice) dai pursuer, fino al tempo **t**. Ciò permette di calcolare, su ogni casella x, la probabilità che l'evasore la occupi (x_e(t)=x) date le misure fino al tempo **t**:

$$P(x_e(t) = x|Y_t) = \alpha p_e(x, Y_{t-1})Y_t \tag{14}$$

Dove α è il coefficiente di normalizzazione tale che $\sum_x$ P(x_e(t)=x | Y_t)=1.
Partendo da queste ultime probabilità, nel caso di **evasore non intelligente**, conoscendo semplicemente il modello Markoviano si riesce a calcolare per ogni casella la probabilità che l'evasore, **al tempo t+1**, le occupi.
Per fare ciò basta utilizzare la seguente formula per ogni casella attiva x:

$$p_e(x, Y_t) = (1 - |\mathbf{A}(x)|\,\rho)P(x_e(t) = x|Y_t) + \rho \sum_{\bar{x} \in \mathbf{A}(x)} P(x_e(t) = \bar{x}|Y_t) \tag{15}$$

In questo modo si è utilizzato il passo di predizione per stimare dove possa trovarsi l'evasore $\hat{\mathbf{x}}$(t+1) $\in \mathbb{R}^{N_c}$.
Nel passo decisionale le stime calcolate influenzeranno la legge di controllo che opera al fine di risolvere il problema riguardante la minimizzazione del funzionale di costo.
Vista la complessità di tale problema non si riesce a trovare una policy che riesca a minimizzare il funzionale, ci si accontenta pertanto di costruire policy $\in \Gamma$ vincolate a muoversi di 1.65° in un singolo timestep e tra queste (sub-ottimo) scegliere quella che minimizza E[$\mathbf{T}^*$].
Si inseriscono di seguito immagini per chiarire meglio ciò che si è appena spiegato mappando, come per il pursuer-evasion games, la probabilità $p_e(x, Y_t)$ in scala di grigi per ogni casella.

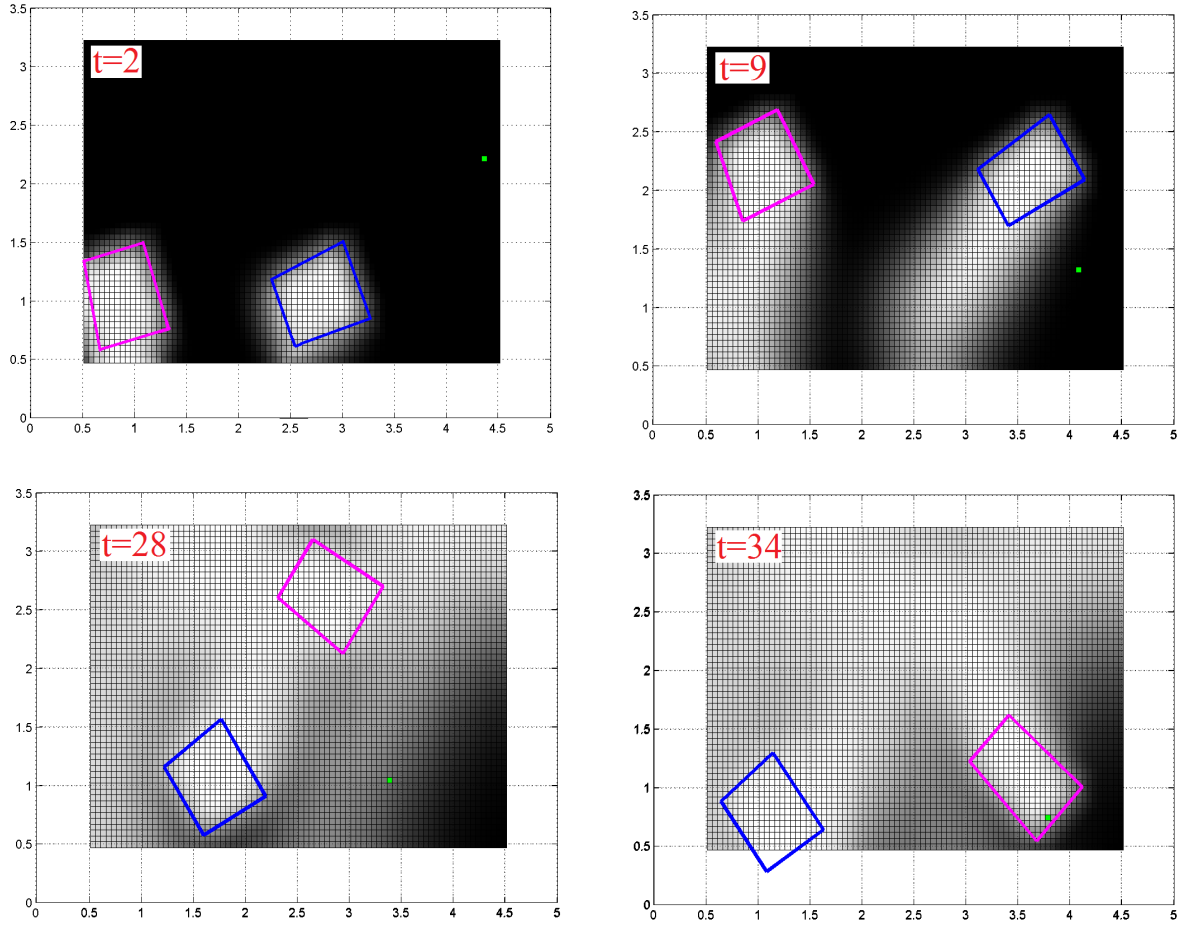

Figura 18: Adattamento del pursuit-evasion con vincoli nelle greedy policy.

Iniziando da $t = 2$ si vedono i FOV (Blu e Magenta), l'evasore (in verde) e la probabilità $p_e(x, Y_t)$ mappata sullo sfondo in scala di grigi, che inizialmente è distribuita in maniera uniforme. In $t = 9$ si vede la bianca coda dei pursuer nella quale $p_e(x, Y_t)$ è bassa. In ogni passo la coda diventa sempre più scura. Alla fine in $t = 34$ l'evasore è catturato.

Nell'adattare il pursuit-evasion game al problema, la differenza di velocità fra pursuer ed evasore, si è dimostrata un grosso ostacolo.

Nel pursuit evasion game tale ostacolo non sussiste in quanto evasore e pursuer hanno la stessa velocità.

3.1.1 Problema delle velocità

Avere velocità non compatibili fra pursuer ed evasore può portare a due problemi:

1. Se l'evasore è troppo veloce ed **intelligente** si corre il rischio di non prenderelo mai
2. Se l'evasore è troppo veloce e **non intelligente** si corre il rischio di non sorvegliare l'intero terreno di ricerca

Mentre, il primo punto è ovvio e verrà ampiamente dimostrato con le simulazioni finali, il secondo non è di facile intuizione.
Se l'evasore è troppo veloce, rispetto alle telecamere, in un timestep potrebbe trovarsi in qualsiasi cella all'interno del terreno di ricerca. Questo implica che la probabilità $p_e(x, Y_t)$ tende ad uniformarsi in tutto lo spazio molto velocemente e, arrivati al Passo Decisionale, non si ottiene nessuna prefenza riguardo la direzione presa dai pursuer facendo risultare casuali i loro movimenti. Le telecamere di conseguenza ricoprono difficilmente l'intero terreno di ricerca.
Tuttavia, anche se si rendono compatibili le velocità, non si può dimostrare algebricamente che si riesce a videosorvegliare l'intero spazio. Se le policy non sono randomiche, ciò è dimostrabile sperimentalmente attraverso le simulazioni, come illustrato nella seguente immagine.

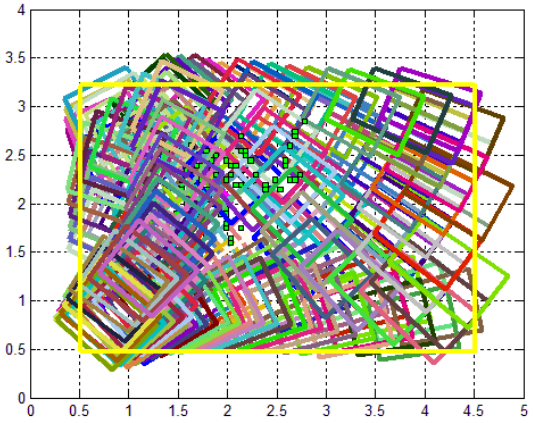

Figura 19: Copertura totale del terreno di ricerca in 80 passi con vincoli nella pursuit policy

Si cerca quindi di rendere compatibili le velocità lineari massime tra pursuer ed evader:

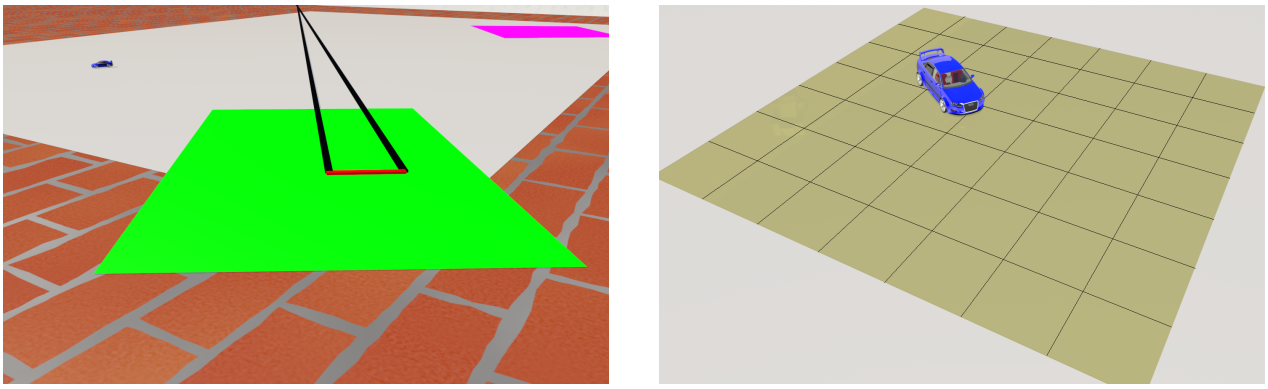

Figura 20: Calcolo delle massime velocità lineari

Le telecamere compiono $1,65°$ in 250ms (si veda Tab (1)), per calcolare la massima velocità lineare si utilizzano l'angolo di Pan (worst case) e il teorema della corda, laddove il raggio è massimo:

$$\frac{2R\sin\frac{1,65°}{2}}{0.25} = 0.59\ \frac{[m]}{[s]} \tag{16}$$

Il raggio massimo (R=5,17m) si ha nella posizione in figura dove, in rosso, viene evidenziata la corda. Vista la non linearità della telecamera l'angolo di

Tilt sottende una corda più grande (Best case), a parità di angolo, quindi si avrà una maggiore velocità lineare.
Per rendere compatibili le velocità bisogna muovere l'evasore al massimo di tre caselle per ogni timestep. Ciò permette all'evasore di avere 48 caselle adiacenti (Figura 20), pertanto esso potrà spostarsi al massimo di 15cm in 250ms, il che permette di raggiungere una velocità **lineare** massima pari a $0.6[\frac{m}{s}]$. Le velocità sono a questo punto compatibili.

4 Indice di prestazioni policy

Come già spiegato il passo Decisionale serve per definire la policy, ovvero la regola di movimento dei pursuers.
Si devono a questo punto trovare le policy che permettono di scovare l'evasore in un tempo mediamente finito e successivamente scegliere la migliore.
Per avere un buon indice di prestazioni si introduce $\mathbf{T}^*$, una variabile aleatoria discreta che tiene conto del primo istante di tempo nel quale si identifica l'intruso. Se esso non viene trovato in un tempo finito si avrà $\mathbf{T}^*=+\infty$, pertanto $\mathbf{T}^*$ è una v.a. che assume valori in $\overline{\mathcal{T}}=\mathcal{T}\cup\{+\infty\}$ e si indica con $F_{\bar{g}}:\overline{\mathcal{T}}\rightarrow[0,1]$ la sua funzione di distribuzione $F_{\bar{g}}(\mathrm{t})=P_{\bar{g}}(\mathbf{T}^*\leq \mathrm{t})$ al fine di calcolarne l'aspettazione.
Dato un qualsiasi $t > 1$ si ha:

$$F_{\bar{g}}(t) = P_{\bar{g}}(\mathbf{T}^* < t) + P_{\bar{g}}(\mathbf{T}^* = t) \tag{17}$$

Dove $P_{\bar{g}}(\mathbf{T}^* = t)$ è la probabilità di trovare l'evasore esattamente al tempo t dato che fino ad ora non si è trovato, inoltre:

$$P_{\bar{g}}(\mathbf{T}^* = t) = f_{\bar{g}}(t)P_{\bar{g}}(\mathbf{T}^* \geq t) = f_{\bar{g}}(t)(1 - P_{\bar{g}}(\mathbf{T}^* < t)) \tag{18}$$

Dove, per ogni $t \in \mathcal{T}$, $f_{\bar{g}}(t)$ denota la probabilità condizionata di trovare l'evasore al tempo t, dato che nessuno è stato trovato fino a quel tempo, quindi, $f_{\bar{g}}(t) = P_{\bar{g}}(\mathbf{T}^* = t \,|\mathbf{T}^* \geq t)$. Da (17) e si può concludere che:

$$F_{\bar{g}}(t) = F_{\bar{g}}(t-1) + f_{\bar{g}}(t)(1 - F_{\bar{g}}(t-1)) \;\; t > 1 \tag{19}$$

La precedente espressione può anche essere scritta come

$$1 - F_{\bar{g}}(\tau) = (1 - f_{\bar{g}}(\tau))(1 - F_{\bar{g}}(\tau-1)) \;\; \tau > 1 \tag{20}$$

Che può essere iterata da $\tau = t_0 + 1 > 1$ a $\tau = t \geq t_0$ per concludere che

$$1 - F_{\bar{g}}(t) = (1 - F_{\bar{g}}(t_0)) \prod_{\tau=t_0+1}^{t} (1 - f_{\bar{g}}(\tau)) \;\; t \geq t_0 \geq 1 \tag{21}$$

Siccome $F_{\bar{g}}(1) = f_{\bar{g}}(1)$, dalla precedente espressione si può anche concludere che

$$1 - F_{\bar{g}}(t) = \prod_{\tau=1}^{t}(1 - f_{\bar{g}}(\tau)) \quad t \geq 1 \tag{22}$$

Quindi

$$F_{\bar{g}}(t) = 1 - \prod_{\tau=1}^{t}(1 - f_{\bar{g}}(\tau)) \quad t \in \mathcal{T} \tag{23}$$

Supponiamo ora che la probabilità di $\mathbf{T}^*$, essendo finita, è uguale a 1 e quindi che $P_{\bar{g}}(\mathbf{T}^* = +\infty) = 0$. Il valore atteso di $\mathbf{T}^*$ è allora uguale a:

$$E_{\bar{g}}[\mathbf{T}^*] = \sum_{t=1}^{\infty} tP_{\bar{g}}(\mathbf{T}^* = t) = F_{\bar{g}}(1) + \sum_{t=2}^{\infty} t(F_{\bar{g}}(t) - F_{\bar{g}}(t-1)) \tag{24}$$

Da questa e (22) si ottiene

$$\begin{aligned} E_{\bar{g}}[\mathbf{T}^*] =& f_{\bar{g}}(1) - \sum_{t=2}^{\infty} t \left(\prod_{\tau=1}^{t}(1 - f_{\bar{g}}(\tau)) - \prod_{\tau=1}^{t-1}(1 - f_{\bar{g}}(\tau)) \right) \\ =& f_{\bar{g}}(1) - \sum_{t=2}^{\infty} t \left((1 - f_{\bar{g}}(t)) - 1 \right) \left(\prod_{\tau=1}^{t-1}(1 - f_{\bar{g}}(\tau)) \right) \end{aligned} \tag{25}$$

che può essere semplicemente scritta come:

$$E_{\bar{g}}[\mathbf{T}^*] = \sum_{t=1}^{\infty} tf_{\bar{g}}(t) \left(\prod_{\tau=1}^{t-1}(1 - f_{\bar{g}}(\tau)) \right) \tag{26}$$

Si è trovato, così, il valore atteso di $\mathbf{T}^*$, che rappresenta l'indice sulle performance della policy utilizzata. Vale la pena notare come $\mathbf{T}^*$ non è altro che una variabile aleatoria geometrica nella quale ad ogni istante cambia la probabilità di avere un successo.

La prima cosa che si dimostra è come questo indice sia finito: a tale scopo si cerca un upper bound della (26), in modo da poter sempre garantire di trovare l'evasore mediamente in un tempo finito per la policy in esame. Affinchè ciò avvenga è necessario avere una policy **persistente o persistente in media**. Ottenute queste policy si utilizza sempre l'indice $E_{\bar{g}}[\mathbf{T}^*]$ per individuare la migliore, cioè quella che lo minimizza.

La metodologia utilizzata è del tutto empirica e sviluppata attraverso simulazioni effettuate con il metodo di Monte Carlo. Se si volesse risolvere il problema analiticamente le difficoltà sarebbero notevoli.
Si cercano, quindi, una categoria di policy che sono persistenti o persistenti in media.

5 Policy persistenti e persistenti in media

La pursuit policy $\bar{g}$:$\mathcal{Y}^* \rightarrow \mathcal{U}$ è detta persistente se esiste qualche $\epsilon > 0$ tale che:

$$f_{\bar{g}}(t) \geq \epsilon \quad \forall t \in \mathcal{T} \tag{27}$$

Dalla (23) risulta evidente che per ogni t la funzione di distribuzione $F_{\bar{g}}(t)$ è monotona non decrescente rispetto a $f_{\bar{g}}(\tau)$, quindi per una policy $\bar{g}$ persistente si avrà $F_{\bar{g}}(t) \geq 1 - (1-\epsilon)^t$, t $\in \mathcal{T}$, con ϵ come in (27).
Pertanto, si può concludere che $sup_{t<\infty} F_{\bar{g}}(t) = 1$ e quindi la probabilità che $\mathbf{T}^*$ sia finito è uguale a 1.
Dall'altra parte il valore atteso $E_{\bar{g}}[\mathbf{T}^*]$ è monotono non crescente (vedi Lemma 4 in appendice) rispetto qualsiasi $f_{\bar{g}}(t)$, t $\in \mathcal{T}$ e per (63) si avrà:

$$E_{\bar{g}}[\mathbf{T}^*] \leq \epsilon \sum_{t=1}^{\infty} t(1-\epsilon)^{t-1} = \epsilon^{-1} \tag{28}$$

Si è quindi dimostrato il seguente Lemma:

Lemma 1
Per una policy persistente $\bar{g}$:$\mathcal{Y}^* \rightarrow \mathcal{U}$, $P_{\bar{g}}(\mathbf{T}^* < \infty) = 1$, $F_{\bar{g}}(t) \geq 1 - (1-\epsilon)^t$, $t \in \mathcal{T}$ and $E_{\bar{g}}[\mathbf{T}^*] \leq \epsilon^{-1}$ con ϵ come in (27).

L'importanza della funzione $f_{\bar{g}}(t)$, al fine di avere un bound sul tempo medio per trovare l'evasore, è di grande importanza.

5.1 Funzione $f_{\bar{g}}(t)$

La funzione $f_{\bar{g}}(t)$ può essere scomposta e legata alle misure Y_t dei pursuer.
Il fatto che l'evasore sia o meno trovato al tempo t è completamente determinato dalle misure prese fino a quel tempo.

Vi è quindi la possibilità di calcolare la probabilità condizionata $f_{\bar{g}}(t)$ di trovare l'intruso al tempo t, dato che fino a t-1 nessuno è stato trovato, come funzione della probabilità condizionata di trovare l'evasore per la prima volta al tempo t, date le misure prese fino al tempo $t-1$.
Ragionando in questo modo le misure prese fino a t, $Y_t \in \mathcal{Y}^*$ sono solo una possibile realizzazione fra tutte le altre che si potrebbero presentare fino a quel tempo. Ad ognuna di queste misure Y_t è possibile dare una probabilità.
Ha pertanto senso introdurre una nuova variabile aleatoria $\mathbf{Y}_t$ che ha come possibili eventi appunto Y_t.
Si denota con $\mathcal{Y}_\tau^{\neg fnd} \subset \mathcal{Y}^*$, $\tau \in \mathcal{T}$, l'insieme di tutte le sequenze di misure di lunghezza τ nelle quali l'evasore non è stato trovato e con Ω lo spazio campionario di tutti i possibili eventi relativi al pursuit-evasion game, ottenendo:

$$\mathcal{Y}_\tau^{\neg fnd} = \mathbf{Y}_\tau = \{\omega \in \Omega : \mathbf{T}^*(\omega) > \tau\} \tag{29}$$

Legando in questo modo la variabile aleatoria $\mathbf{T}^*$ alle misure raccolte dai pursuers $\{\omega \in \Omega : \mathbf{T}^*(\omega) \geq \tau\} = \{\omega \in \Omega : \mathbf{Y}_{\tau-1} \in \mathcal{Y}_{\tau-1}^{\neg fnd}\}$, $\tau \in \mathcal{T}$.
Si riesce quindi ad espandere $f_{\bar{g}}(t)$ come:

$$\begin{aligned} &f_{\bar{g}}(t) = P_{\bar{g}}(\mathbf{T}^* = t | \mathbf{T}^* \geq t) = \\ &\sum_{Y_{t-1} \in \mathcal{Y}_\tau^{\neg fnd}} h_{\bar{g}}(Y_{t-1}) P_{\bar{g}}(\mathbf{Y}_{t-1} = Y_{t-1} \,\big|\, \mathbf{Y}_{t-1} \in \mathcal{Y}_\tau^{\neg fnd}) \quad t \in \mathcal{T} \end{aligned} \tag{30}$$

Dove $h_{\bar{g}}(t)$:$\mathcal{Y}^* \rightarrow$ [0,1] è una funzione che mappa ogni sequenza $Y \in \mathcal{Y}^*$ di $\tau = |Y|$ misurazioni nella probabilità condizionata di trovare l'evasore per la prima volta in $\tau + 1$, date le misure $\mathbf{Y}_\tau$=Y raccolte fino a τ, quindi, $h_{\bar{g}}(Y) = P_{\bar{g}}(\mathbf{T}^* = |Y| + 1 \,\big|\, \mathbf{Y}_{|Y|} = Y)$. L' equazione (30) mostra che $f_{\bar{g}}(t)$ è uguale al valore atteso di $f_{\bar{g}}(t)$ quando la misura di probabilità è condizionata da $\mathbf{Y}_{t-1} \in \mathcal{Y}_{\tau-1}^{\neg fnd}$. Questo può essere scritto come:

$$f_{\bar{g}}(t) = E_{\bar{g}}[h_{\bar{g}}(\mathbf{Y}_{t-1}) | \mathbf{Y}_{t-1} \in \mathcal{Y}_{\tau-1}^{\neg fnd}] \tag{31}$$

Questa equazione permette di calcolare la probabilità $f_{\bar{g}}(t)$ usando la funzione $h_{\bar{g}}(t)$. Quest'ultima viene ampiamente utilizzata in seguito per garantire la persistenza in media.

5.2 Persistenza in media

Avendo precedentemente esaminato la persistenza delle policy ci si accorge che l'ipotesi (27) difficilmente si realizza perchè deve valere $\forall\ t \in \mathcal{T}$. Per tale motivo, spesso le pursuit policy non sono persistenti, ma solo persistenti in media.
Per persistenza in media si intende che, per ogni istante di tempo t esiste una finestra temporale nella quale esiste un intero T e $\epsilon > 0$ tali che:

$$\bar{f}_{\bar{g}}(t) = P_{\bar{g}}(\mathbf{T}^* \in \{t, t+1, ..., t+T-1\} \,|\mathbf{T}^* \geq t) \geq \epsilon \quad \forall t \in \mathcal{T} \quad (32)$$

Dove con T si indica il periodo di persistenza.
Per analizzare le policy, che sono persistenti in media con periodo T, si definisce:

$$\bar{F}(k) = F_{\bar{g}}(kT) \quad \bar{f}(k) = \bar{f}_{\bar{g}}((k-1)T+1) \quad k \in \{1, 2, ..\} \quad (33)$$

Questa funzione può essere interpretata come la funzione di distribuzione di $\mathbf{T}^*$ per un gioco simile che va T volte più veloce. Ragionando come in precedenza (23), dove $\bar{F}$ e $\bar{f}$ ora hanno il ruolo di $F_{\bar{g}}$ e $f_{\bar{g}}$, si può concludere che:

$$\bar{F}(k) = 1 - \prod_{i=1}^{k}(1 - \bar{f}(i)) \quad k \in \{1, 2, ..\} \quad (34)$$

Chiaramente se la pursuit policy è persistente in media, ogni $\bar{f}(i) \geq \epsilon$, e ciò significa che il gioco T volte più veloce è persistente. Riflettendo dunque, come nella dimostrazione del **Lemma 1**, si può concludere che se la policy è **persistente in media** di periodo T, aumentando di T volte la velocità del gioco

questo risulta **persistente** ottenendo per ogni $k \in \{1, 2, ..\}$:

$$F_{\bar{g}}(kT) \geq 1 - (1 - \epsilon)^k, \tag{35}$$

Mentre l'aspettazione del gioco T volte più veloce è

$$\sum_{k=1}^{\infty} kP_{\bar{g}}(\mathbf{T}^* \in \{(k-1)T + 1, (k-1)T + 2, ..kT\}) \leq \epsilon^{-1} \tag{36}$$

Ma allora riportando il gioco a velocità normale (sostituendo $t = kT$) si ottiene $F_{\bar{g}}(t) \geq 1 - (1 - \epsilon)^{\lfloor \frac{t}{T} \rfloor}$, t $\in \mathcal{T}$, e:

$$\begin{aligned} E_{\bar{g}}[\mathbf{T}^*] &= \sum_{k=1}^{\infty} \sum_{i=1}^{T} ((k-1)T + i) P_{\bar{g}}(\mathbf{T}^* = (k-1)T + i) \\ &\leq T \sum_{k=1}^{\infty} k \sum_{i=1}^{T} P_{\bar{g}}(\mathbf{T}^* = (k-1)T + i) \leq T\epsilon^{-1} \end{aligned} \tag{37}$$

Si dimostra quindi:

LEMMA 2

Per una policy persistente in media $\bar{g}$:$\mathcal{Y}^* \rightarrow \mathcal{U}$, con periodo di persistenza T $P_{\bar{g}}(\mathbf{T}^* < \infty) = 1$, $F_{\bar{g}}(t) \geq= 1 - (1 - \epsilon)^{\lfloor \frac{t}{T} \rfloor}$, $t \in \mathcal{T}$ e $E_{\bar{g}}[\mathbf{T}^*] \leq \mathrm{T}\epsilon^{-1}$ con ϵ come in (32).

Concludendo, il **Lemma 1** e il **Lemma 2** garantiscono che una policy persistente o persistente in media permette di trovare l'evasore mediamente in un tempo finito, motivo per cui nell'algoritmo si cerca di ottenere policy con tali caratteristiche.

5.3 Metodo operativo per calcolare la persistenza

Compresa l'importanza di avere una policy persistente si cerca di determinare una condizione sufficiente in modo da garantire operativamente tale proprietà. Si utilizza, a tale scopo, la probabilità condizionale $h_{\bar{g}}$, precedentemente introdotta.

Una condizione sufficiente affinchè la (27) sia verificata è:

$$h_{\bar{g}}(Y) \geq \epsilon \quad \forall t \in \mathcal{T}, Y \in \mathcal{Y}_{t-1}^{\neg fnd} \tag{38}$$

questa è diretta conseguenza della (30) e del fatto che:

$$\sum_{Y_{t-1} \in \mathcal{Y}_{\tau}^{\neg fnd}} P_{\bar{g}}(\mathbf{Y}_{t-1} = Y_{t-1} | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd}) = 1 \ \ \forall t \in \mathcal{T} \tag{39}$$

Quest'ultima risulta chiara in quanto somma le probabilità di tutti gli eventi della v.a $\mathbf{Y}_{t-1}$.

Per la persistenza in media si introduce il seguente lemma [9]:

Lemma 3

Una condizione sufficiente, affinchè la policy $\bar{g}$ sia persistente in media con periodo T, è l'esistenza di $\delta > 0$ tale che, per ogni $t \in \mathcal{T}$ e ogni Y $\in \mathcal{Y}_{t+T-2}^{\neg fnd}$, c'è qualche $\tau \in \{t-1, t, ..., t+T-2\}$ per il quale $h_{\bar{g}}(Y_\tau) \geq \delta$ dove Y_τ denota la sequenza che consiste delle prime τ misurazioni in Y. In questo caso la (32) tiene con:

$$\epsilon = \begin{cases} \frac{1}{T}(1-\frac{1}{T})^{T-1}, & \delta \geq \frac{1}{T} \\ \delta(1-\delta)^{T-1}, & \delta < \frac{1}{T} \end{cases} \tag{40}$$

A seconda del valore di δ si ottengono due diversi bound. δ va, infatti, a influire sul bound della variabile $\bar{f}_{\bar{g}}(t)$ come spiegato in appendice. Tali bound sono conservativi e il primo è impossibile da raggiungere.

[9] Dimostrazione in appendice

6 Algoritmo utilizzato

L'algoritmo utilizzato si compone essenzialmente di due programmi: il primo contiene tutte le informazioni necessarie per poter fare il patrolling (dimensione terreno di ricerca, angoli FOVs, ecc.) e viene inizialmente fatto funzionare OFFLINE, il secondo utilizza le informazioni del precedente e aggiorna ad ogni passo la posizione delle telecamere a seconda della policy scelta, questo, ovviamente viene fatto funzionare ONLINE.
Tali programmi sono stati, in primo luogo implementati in Matlab utilizzando il classico modello pinhole per le telecamere, e successivamente realizzati in C++ e testati sul Testbed.

6.1 Programma OFFLINE

Inizialmente, si sono impostati i limiti fisici delle telecamere, inserendo il massimo e il minimo angolo di pan, tilt, lo zoom e i dati delle telecamere (posizione rispetto coordinate mondo e matrice intrinseci, angolo di campionamento e area FOVs in metri quadrati).
Successivamente, si sono inseriti i limiti del terreno di ricerca, il quale deve risultare compatibile con i limiti fisici delle telecamere. La superficie è stata divisa in celle quadrate (lato 5cm). Le celle che risultano interamente dentro ad un FOV vengono chiamate **Celle Attive** e solamente in queste l'evasore può muoversi.
Si descrive, quindi, l'immagine (6.1).
In alto, nell'immagine di sinistra, viene scelto un terreno di ricerca (giallo) che non tiene conto dei limiti fisici delle delle telecamere (cerchi neri) e che risulta molto più grande rispetto a quello del Testbed (immagine sotto a sinistra).

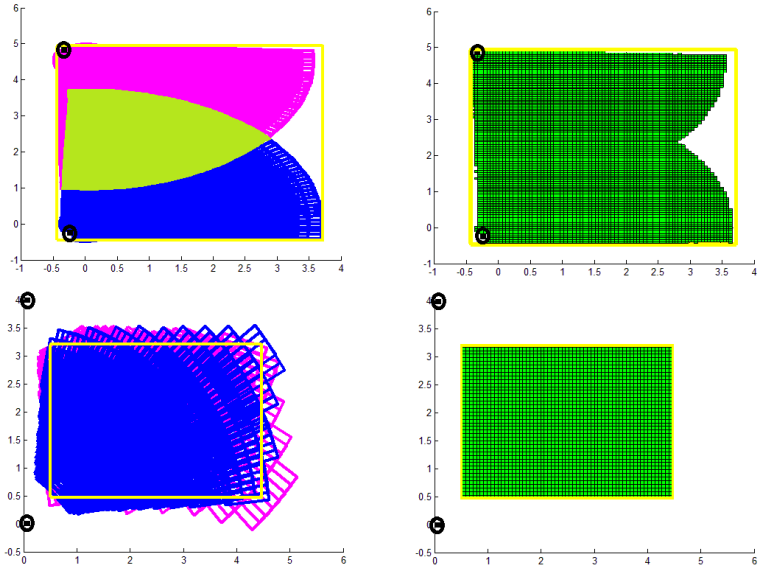

Figura 21: Analysis of Search Terrain

I FOVs (Magenta e Blu), infatti, non riescono a ricoprirlo interamente e generano come caselle attive solo quelle presenti nell'immagine di destra, limitando a tali celle i movimenti dell'evasore.

Nelle due immagini in basso, che rappresentano il Testbed, si sceglie un terreno di ricerca più piccolo, esplorato interamente da entrambe le telecamere. Per chiarezza non è stata colorata di verde la zona di overlap. In queste immagini si vede come le caselle attive combacino con l'intero terreno di ricerca e siano necessari meno FOVs per esplorarlo completamente.

Si fa notare che le caselle attive sono generate dall'**unione** dei FOVs di entrambe le telecamere. In seguito si capirà come, al fine di avere una policy persistente in media, non sia indispensabile che anche l'**intersezione** dei FOVs copra le stesse caselle attive.

Di seguito si analizza l'algoritmo offline in relazione al Testbed adottato.

Si creano, inizialmente, tante matrici quante sono le telecamere e ognuna di queste matrici raccoglie le informazioni di tutti i FOVs della singola telecamera, annotando per ciascuno gli angoli di Tilt, Pan, Zoom, le caselle attive completamente all'interno e le caselle attive non completamente all'interno.

Di queste ultime si scrive anche il valore dell'area interna al FOV.

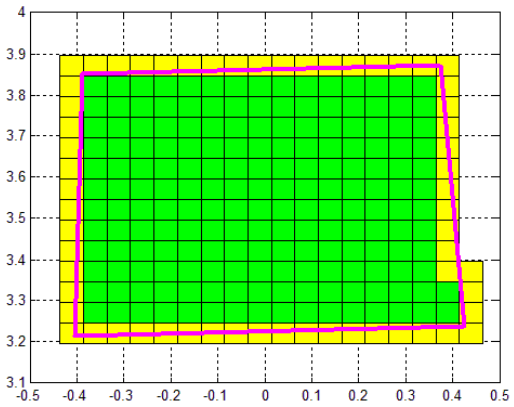

Figura 22: Esempio di celle contenute nel FOV

Le figura mostra il FOV di pan=-88.35° tilt=69.7° e λ=0.6131: in verde sono evidenziate le caselle completamente dentro, mentre in giallo quelle non completamente all'interno.

Successivamente si sono creati tanti grafi, non orientati e pesati, pari al numero delle telecamere.

Si genera, in primo luogo, la matrice delle adiacenze, dove i nodi sono rappresentati da tutti i FOVs della singola telecamera e si attribuisce il valore 1 al peso dell'arco solamente se due FOVs sono adiacenti (si ricorda che per ogni FOV in condizioni normali ci sono 8 FOVs adiacenti).

Ciò permette di costruire un grafo, trovare i cammini minimi (algoritmo di Dijkstra) e attribuirne il costo. Con tali costi si crea una mappa (Figura 24) che, per ogni FOVs, indica i FOVs distanti un passo (viola nell'immagine), due passi (nero), e così via, dal FOV di partenza (rosso).

Va da sè che in ogni cammino minimo si minimizza il numero di FOVs che la telecamera deve attraversare riducendo i consumi energetici di quest'ultima. Così facendo **non** si va a minimizzare la distanza euclidea all'interno del terreno di ricerca. Non c'è da stupirsi se si presentano casi come (Figura 23) i

seguenti:

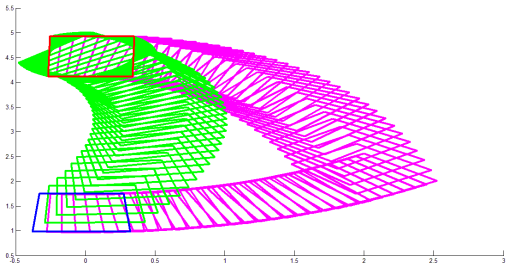

Figura 23: Più di un percorso minimo

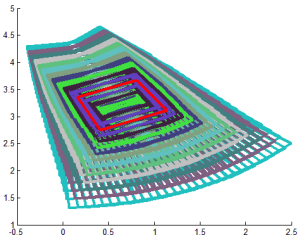

Figura 24: Mappa della distanza

Nell'immagine di sinistra, andando dal FOV rosso a quello blu, la strada dei FOVs magenta attraversa 55 FOVs, esattamente come la strada verde. Ciò dimostra che possono esserci divesi cammini minimi non banali tra due FOVs, tali cammini richiedono la stessa energia alla telecamera e sarà pertanto indifferente utilizzarne uno o l'altro nell'algoritmo.

Nell'immagine di destra si evidenzia che, avendo scelto un movimento quantizzato di solo 1,65°, i FOVs adiacenti ricoprono quello di partenza (rosso). Questo non è rilevante al fine dell'algoritmo, infatti, l'importante è che rispetto a tale FOV, i FOVs adiacenti introducano una minima percentuale di nuove Caselle Attive e ciò è sempre verificato con il passo di quantizzazione scelto.
Si vuole sottolineare che ogni telecamera può coprire la propria zona di ricerca in un numero **finito** di passi partendo da una qualsiasi posizione iniziale. Per il primo pursuer tale numero di passi è denotato con d_1, per il secondo con d_2 e così via, quindi al massimo in d=max$\{d_1, d_2, ..d_{N_p}\}$ passi si può arrivare in qualsiasi $\mathbf{v}_{final}$ a partire da qualsiasi $\mathbf{v}_{init} \in \mathcal{V}$, anche in presenza di zone di Overlap tra le telecamere.

6.2 Programma ONLINE

L'algoritmo ONLINE gestisce la posizione dei pursuers e aggiorna, su ogni casella attiva x, la probabilità $p_e(x, Y_t)$ ad ogni timestep. Il programma elabora i passi di **Misura**, di **Predizione** e **Decisionale**.
Si analizzano i primi due passi, creando le matrici chiamate rispettivamente 'Misure' e 'Predizione'.

- Misure: tale matrice è riempita con numeri $\in$ [0,1], ogni suo elemento corrisponde ad una casella attiva. Il valore dato ad ogni singola cella genera una maschera di misure che, in base alla posizione dei pursuer al tempo **t**, tiene conto se una casella è o meno osservata da uno dei due pursuer. Un casella attiva, che al tempo **t** è completamente coperta da un FOV, assume valore pari a 0, se invece la cella è completamente esterna assume valore uguale a 1. Alle celle non completamente coperte viene assegnata la percentuale di area esterna, in formula:

$$\frac{\text{(Area Casella-Area interna)}}{\text{Area Casella}} \tag{41}$$

- Predizione: Ogni elemento della matrice al tempo **t** corrisponde alla probabilità $p_e(x, Y_t)$ di trovare l'evasore all'istante successivo **t+1** in una determinata Casella Attiva x. Inizialmente tale probabilità è uniforme su tutte le caselle e risulta pari a $1/N_c$, con N_c numero di Caselle Attive.

Il passo di **Misura** si realizza analizzando le intersezioni tra FOV e caselle attive, aggiornando di conseguenza la matrice 'Misure' ad ogni timestep.
Il passo di **Predizione** è, invece, concettualmente più elaborato e la matrice di 'Predizione' viene aggiornata in maniera ricorsiva ad ogni timestep. Per

poter calcolare la matrice di 'Predizione' al tempo **t** è, infatti, necessario avere le 'Misure' al tempo **t** e la matrice di 'Predizione' al tempo **t-1**. Moltiplicando tali matrici e normalizzando il tutto (ciò viene automaticamente fatto solo sulle caselle attive) si riesce ad avere in ogni casella x la probabilità che l'evasore, **al tempo t**, le occupi $P(x_e(t) = x|\mathbf{Y}_t = Y_t)$. In formule:

$$P(x_e(t) = x|\mathbf{Y}_t = Y_t) = \alpha p_e(x, Y_{t-1}) \begin{cases} 0, & \text{se x} \cap \text{FOVs=x} \\ Formula(41), & \text{se x} \cap \text{FOVs} \neq \oslash \\ 1, & \text{se x} \cap \text{FOVs} = \oslash \end{cases} \tag{42}$$

Dove α è il coefficiente di normalizzazione tale che $\sum_x$ P(x_e(t)=x | $\mathbf{Y}_t$=Y_t)=1.
Partendo da queste ultime probabilità, nel caso di **evasore non intelligente**, conoscendo semplicemente il parametro ρ del modello Markoviano si riesce a calcolare per ogni casella la probabilità che l'evasore, **al tempo t+1**, le occupi.
Per fare ciò basta utilizzare la seguente formula per ogni casella attiva x:

$$p_e(x, Y_t) = (1 - |\mathbf{A}(x)|\,\rho)P(x_e(t) = x|\mathbf{Y}_t = Y_t) + \rho \sum_{\bar{x} \in \mathbf{A}(x)} P(x_e(t) = \bar{x}|\mathbf{Y}_t = Y_t) \tag{43}$$

Dove $\mathbf{A}(x)$ rappresenta le caselle attive adiacenti alla casella x e $|\mathbf{A}(x)|$ la loro cardinalità.
Analizzando tale formula si capisce che, per ogni casella attiva, la probabilità che l'evasore al tempo **t+1** sia in x è data dalla probabilità che l'evasore resti fermo in x al tempo **t** o che che da caselle adiacenti ad x, salti in x, il tutto pesato con la probabilità di trovarsi in tali caselle al tempo **t**.
Agendo iterativamente si riesce a costruire la matrice 'Predizione' ad ogni timestep.
Si apre volutamente una parentesi riguardo il passo di **Predizione** mostran-

done di seguito una importante proprietà.
Dalla (42) e (43) per qualsiasi sequenza $Y_t \in \mathcal{Y}_t^{\neg fnd}$ di t misure per le quali l'evasore non è stato trovato, e per qualsiasi x $\in \mathcal{X}$ per il quale x non è nella posizione occupata dai pursuer al tempo t si ha:

$$p_e(x, Y_t) \geq (1 - |\mathbf{A}(x)|\,\rho) P(x_e(t) = x | \mathbf{Y}_t = Y_t) \geq \alpha(1 - |\mathbf{A}(x)|\,\rho) p_e(x, Y_{t-1}) \tag{44}$$

Si calcola ora α al tempo **t**:

$$\alpha = \frac{1}{\sum_{\mathrm{X}} p_e(x, Y_{t-1}) \cdot Misure(t)} \tag{45}$$

E' facile vedere che $\alpha \geq 1$ in quanto per la matrice 'Predizione' si ha per ogni **t**:

$$\sum_x p_e(x, Y_{t-1}) = 1 \tag{46}$$

Moltiplicando tale matrice per 'Misure' all'istante **t**, le caselle dove sono contenuti i FOV andranno a zero, pertanto $\sum_x$ p_e(x | Y_{t-1})·Misure(t)≤1, quindi $\alpha \geq 1$. Si può scrivere che:

$$p_e(x, Y_t) \geq (1 - |\mathbf{A}(x)|\,\rho) p_e(x, Y_{t-1}) = \gamma p_e(x, Y_{t-1}) \tag{47}$$

Dove $\gamma = (1 - |\mathbf{A}(x)|\,\rho) \leq 1$, che nel caso peggiore, se l'evasore si può muovere di 3 caselle e si hanno tutti gli adiacenti a disposizione $\gamma = (1 - 48\frac{1}{49}) \approx 0.98 \leq 1$. Questa proprietà sarà utilizzata in (7.2) per dimostrare l'**Assunzione 2**.
Elaborata la predizione non resta che spostare le telecamere al fine di catturare l'evasore, mediamente, in un tempo finito attraverso diverse leggi di controllo: le policy, che sono il cuore del passo **Decisionale**.

7 Design Policy

Si descriverà di seguito una tecnica per ottenere policy almeno persistenti in media, che permettono di avere $E[\mathbf{T}^*]$ finito.
Come spiegato nella sezione (3) le policy che si andranno a creare avranno limiti nei movimenti ad ogni timestep, per questo si introduce la definizione di:

Ammissibilità
Una policy $\bar{g}$:$\mathcal{Y}^* \rightarrow \mathcal{U}$ è detta ammissibile se per ogni sequenza di misure $Y \in \mathcal{Y}^*$, $\bar{g}$(Y) è raggiungibile in un singolo timestep dalle posizioni **v** dei pursuers, specificate nelle ultime misure in Y.

Quindi al fine di avere una policy ammissibile è sufficiente muovere tutti i pursuers nelle loro rispettive 8 posizioni adiacenti, in ogni timestep. Le policy utilizzate sono tutte ammissibili.
Per garantire la persistenza in media della policy si prende un'arbitraria sequenza di misure $Y \in \mathcal{Y}^*$ e si denota con $\Re(Y, m) \subset \mathcal{X}$ l'insieme di celle attive raggiungibile in m passi dai pursuer quando questi partono da FOVs specificati dall'ultima misura di Y. Successivamente per ogni pursuer, k, si definisce:

$$\Re(Y, m) = \{x \in \mathcal{X} : \exists\ \mathbf{v} \in \mathcal{V}\ tale\ che\ x \cap \mathbf{v} = \mathrm{x}\ \mathrm{e}\ \min_k dist(\mathbf{v}, k, Y) \leq m\} \quad (48)$$

Dove $\Re(Y, d) = \mathcal{X}$, con d=max$\{d_1, d_2, ..d_{N_p}\}$.
Si definisce m(Y) come il più piccolo intero positivo per il quale:

$$\exists x^* \in \Re(Y, m(Y)) : p_e(x^*, Y) \geq \frac{\gamma^{d-m(Y)}}{Nc} \quad (49)$$

Con γ come in (47).

L'intero m(Y) è al più uguale a d, questo perchè la (49) vale anche quando m(Y)=d, infatti, almeno una cella x $\in \mathcal{X}$ deve avere $p_e(x, Y) \geq \frac{1}{Nc}$. Questo si dimostra facilmente per assurdo perchè, se così non fosse, si ha che $\forall$ x $p_e(x, Y) < \frac{1}{Nc}$ e la (46) non sarebbe verificata.
In un timestep, partendo dalla posizione dei pursuer (verde e nero) e utilizzando la matrice di 'Predizione', è quindi possibile trovare più di una cella x^* (rosso) guardando a m(Y) passi, come mostra la seguente immagine.

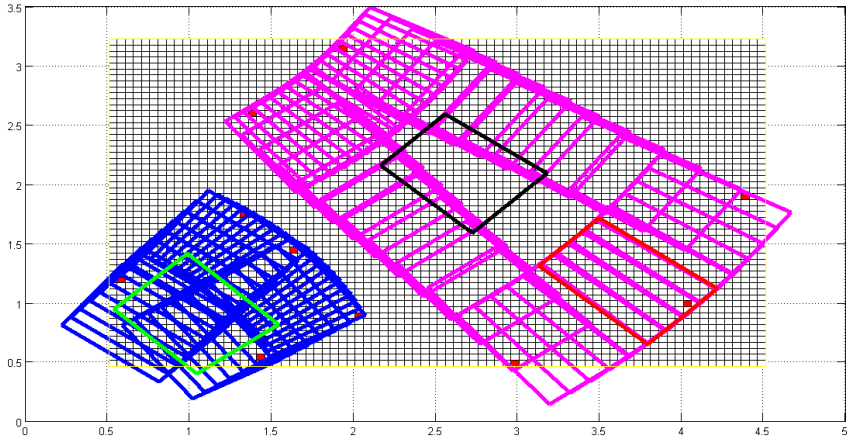

Figura 25: Rappresentazione v_{final} con $m(Y) = 5$

Tra tutti i FOVs distanti m(Y) passi dalle posizioni dei pursuers e contenenti x^* si sceglie quello avente la probabilità più elevata di trovare l'evasore: v_{final} (FOV rosso) è raggiungibile in m(Y) passi da una sola delle telecamere (le altre nemmeno contegono quel FOV), chiamata **pursuer prescelto** ed evidenziato in nero nel disegno.

Per come si definisce m(Y) e v_{final} è evidente che:

$$P_e(v_{final}(Y), Y) \geq \frac{\gamma^{d-m(Y)}}{Nc} \tag{50}$$

in quanto

$$P_e(v_{final}(Y), Y) = \sum_{j=1}^{N_i} p_e(x_j, Y) \tag{51}$$

Dove N_i sono le caselle interne a v_{final} e tra queste c'è almeno x^*.
Utilizzando questo metodo per scegliere v_{final} si è sicuri di avere una policy persistente in media (la dimostrazione dettagliata verrà fatta nella sezione (7.2)).
Ora non resta che costruire policy ammissibili che contengano tra i FOVs di destinazione $\mathbf{u}_{final}(Y)$, in m(Y) passi, il FOV v_{final}.

7.1 Caratterizzazione singole policy

L'unico vincolo da soddisfare per garantire la persistenza in media di una determinata policy è che, in almeno m(Y) passi, il **pursuer prescelto** arrivi in $v_{final}(Y)$.
Si presentano di seguito quattro policy ammissibili.

- LOCALMAX
- TOTALMAX
- LOCK
- RANDOM

7.1.1 Policy LOCALMAX

All'istante **t**, date le 'Predizioni' e la posizione dei pursuer, si parte da m(Y)=1 e si verifica se sono presenti caselle che soddisfano la (49), altrimenti si passa a m(Y)=2 continuando fino a che non è soddisfatta la (49). Si nota come la funzione di probabilità sia crescente ,quindi, non si devono ricontrollare i passi

già visti. A questo punto si calcola v_{final} trovando automaticamente il **pursuer prescelto**. Quest'ultimo percorrerà il cammino minimo con l'algoritmo di Dijkstra al fine di arrivare in v_{final} in m(Y) passi, mentre gli altri pursuers, ad ogni passo, cercheranno di massimizzare la seguente:

$$\mathbf{v}_{final}(Y) = \arg \max_{[v_1, v_2, .. v_{N_p}] \in \mathbf{N}} \sum_{k=1}^{N_p} P_e(v_k, Y) \tag{52}$$

Dove **N**= {**v** :v$_{final}$ è un elemento di **v** distante m(Y) passi}.

In questo modo, guardando sempre localmente, si evita di sovrapporre i FOVs.

Chiaramente ad ogni timestep la matrice di 'Predizione' si autoaggiorna e dopo m(Y) passi si dovrà ripetere l'algoritmo cercando altre celle x^* procecendo come prima.

Si sottolinea che se m(Y)=1 **per ogni istante di tempo**, ovvero se $\forall$ t:

$$\exists x^* \in \Re(Y, 1) : p_e(x^*, Y) \geq \frac{\gamma^{d-1}}{Nc} \tag{53}$$

allora v_{final} è sempre distante un passo dal **pursuer prescelto**, il quale diventa l'adiacente con probabilità massima di trovare l'evasore.

Il pursuer prescelto, ad ogni passo, si comporterebbe come gli altri pursuer, cioè massimizzerebbe la probabilità di trovare l'evasore solamente guardando i suoi 8 adiacenti, generando in questo modo un semplice algoritmo molto più scalabile del precedente e che non necessita dell'algoritmo di Dijkstra.

Se si riuscisse a dimostrare che, per ogni istante di tempo, m(Y)=1 si avrebbe una policy persistente: ciò si dimostra solo in maniera empirica attraverso le simulazioni.

7.1.2 Policy TOTALMAX

Si procede esattamente come la policy LOCALMAX, ma in questo caso, x^* è la cella che ha probabilità più alta nella matrice 'Predizione' fra tutte le caselle attive. Per quanto visto tale probabilità è sicuramente $p_e(x^*, Y) \geq \frac{1}{Nc}$, in quanto esiste sempre una cella con questa probabilità e quindi, ci si può arrivare sicuramente in al massimo d passi garantendo la persistenza in media. Utilizzando questo approccio si nota che mediamente occorrono più passi per raggiungere v_{final} rispetto alla policy LOCALMAX.

7.1.3 Policy LOCK

Tenendo fermi i pursuer in ogni timestep e facendo muovere l'evasore sicuramente al passo successivo la probabilità di trovarlo sui FOV correnti è diversa da zero. Quindi è verificata la (27).

7.1.4 Policy RANDOM

In questo caso non si riesce a dimostrare la persistenza in media, in quanto le telecamere si muovono in maniera randomica. Tale policy verrà utilizzata solo per portare un confronto con policy più intelligenti.

7.2 Verifica della persistenza in media

Si vuole dimostrare che le policy LOCALMAX e TOTALMAX sono persistenti in media con periodo di persistenza T=d.
Affinchè ciò accada è necessario che siano soddisfatte le seguenti due assunzioni:

Assunzione 1

La policy scelta deve essere ammissibile e in d passi, partendo da qualsiasi FOVs $\mathbf{v}_{init}$, è possibile raggiungere qualsiasi posizione $\mathbf{v}_{final}$. Inoltre, in un singolo passo il pursuer prescelto deve essere un passo più vicino a v_{final}.
Tale assunzione è chiaramente verificata per ogni policy analizzata LOCALMAX e TOTALMAX.

Assunzione 2

Esiste una costante positiva $\gamma \leq 1$, tale che, per ogni sequenza $Y_t \in \mathcal{Y}_t^{\neg fnd}$ di t $\in \mathcal{T}$ misure per le quali l'evasore non è stato trovato, si ha:

$$p_e(x, Y_t) \geq \gamma p_e(x, Y_{t-1}) \tag{54}$$

Ciò deve valere per ogni x $\in \mathcal{X}$, a meno delle caselle che appartengono ai FOVs dei pursuers, specificati nell'ultima misura Y_t. Dove Y_{t-1} denota la sequenza formata dai primi t-1 elementi di Y_t.

In pratica l'Assunzione 2 richiede che la probabilità di trovare l'evasore nella cella x $\in \mathcal{X}$, date le misure prese fino a quel momento, non decada più di un certo ammontare al passo successivo, a meno che tale passo x non sia all'interno di un FOV.
Si è già visto come, nel caso di modello Markoviano per l'evasore, tale assunzione sia sempre verificata (47) per tutte le policy.
Soddisfando tali assunzioni nelle policy LOCALMAX e TOTALMAX è facile dimostrare che il Lemma 3 è verificato.
In particolare si mostrerà che, data una qualsiasi sequenza Y $\in \mathcal{Y}_{t+T-2}^{\neg fnd}$ di misure, per le quali l'evasore non è stato trovato e compatibili con la pursuit policy g_c esiste $\tau \in \{t-1, t, ..., t+T-2\}$ per il quale $h_{g_c}(Y_\tau) \geq \delta = \frac{\gamma^{d-1}}{N_c}$.
Per dimostrare ciò si prende un s $\in \{t-1, t, ..., t+T-3\}$ per il quale $m(Y_s)$>1.

Per tale motivo:

$$p_e(x, Y_s) < \frac{\gamma^{d-1}}{N_c} \quad \forall x \in \Re(Y_s, 1) \tag{55}$$

Per $\gamma \leq 1$ e per la (49) si ha:

$$\exists x^* \in \Re(Y_s, m(Y_s)) \ : p_e(x^*, Y_s) \geq \frac{\gamma^{d-m(Y_s)}}{N_c} \geq \frac{\gamma^{d-1}}{N_c} \tag{56}$$

Da (56) e da (55) si può concludere che $x^* \notin \Re(Y_s, 1)$, quindi, v_{final} non è certamente distante un passo dalle posizioni dei pursuer al tempo s, ma è distante al più $1 < m(Y_s) \leq d$ passi.

Per questo motivo e per l' Assunzione 2 si ottiene che, al passo successivo (s+1), le celle denotate con x^* diminuiscono la loro probabilità al massimo di:

$$p_e(x^*, Y_{s+1}) \geq \gamma p_e(x^*, Y_s) \geq \frac{\gamma^{d-(m(Y_s)-1)}}{N_c} \quad \forall x^* \in \Re(Y_s, m(Y_s)) \tag{57}$$

Visto che l'assunzione 1 è verificata e si usano policy g_c (ammissibili) per muovere i pursuers, al tempo s il **pursuer prescelto** si muove verso il FOV $v_{final}(Y_s)$ accorciando la distanza di un passo, passando attraverso un cammino minimo; pertanto al tempo s+1 la distanza da v_{final} è di $m(Y_s)$-1 passi. Inoltre, la casella x^* in v_{final} appartiene a $\Re(Y_{s+1}, m(Y_s) - 1)$, come è evidente da (57). Questo unito a (49) e (57) implica che $m(Y_{s+1}) \leq m(Y_s) - 1$. Iterando il ragionamento per i successivi istanti di tempo è chiaro che $m(Y_s)$ decresce fino a diventare al minimo 1.

Ricapitolando:

Dati $m(Y) \leq$d passi e le policy g_c adottate (ad ogni timestep ci si avvicina a v_{final} con il pursuer prescelto) si ottiene che, per almeno un $\tau \in \{$t-1,t,..., t+T-2 con T=d, $m(Y)$ diminuisce ad ogni passo fino a valere al più $m(Y_\tau)$=1 al tempo τ.

Partendo dall'istante di tempo τ avendo $m(Y_\tau)$=1 si vuole far evolvere di un passo il pursuit-evasion game ricordando che:

- $\mathbf{u}(\tau + 1) = [v_1, v_2, .., v_{N_p}]^T = g_c(Y_\tau)$

- $u_k(\tau+1) = v_k$, con $k \in \{1, 2, ...N_p\}$
- La probabilità di trovare l'evasore per la prima volta al tempo $\tau+1$ date le misure fino τ è:

$$h_{g_c}(Y_\tau) = \sum_{k=1}^{N_p} P_e(v_k, Y_\tau) \tag{58}$$

Dove $P_e(v_k, Y_\tau)$ è la probabilità che l'evasore al passo $\tau+1$ si trovi nel FOV del kappesimo pursuer, dato che non è mai stato trovato.

Visto che $m(Y_\tau)$=1, la distanza dalla casella x^* contenuta in v_{final} è pari solo a un passo e per la (49) si ha $p_e(x^*, Y_\tau) \geq \frac{\gamma^{d-1}}{N_c}$, perciò il pursuer prescelto può raggiungere v_{final} in un singolo passo. Si conclude che, per la (51), esiste un τ tale che:

$$h_{g_c}(Y_\tau) \geq P_e(v_{final}(Y_\tau), Y_\tau) \geq \delta = \frac{\gamma^{d-1}}{N_c} > 0 \tag{59}$$

Una semplice applicazione del lemma 3 dimostra che le policy LOCALMAX e TOTALMAX sono persistenti in media con periodo T=d e che il bound sulla media è:

$$E[\mathbf{T}^*] \leq \begin{cases} d^2(1-\frac{1}{d})^{-d+1}, & \text{se } \frac{\gamma^{d-1}}{N_c} \geq \frac{1}{d} \\ \frac{dN_c}{\gamma^{d-1}}(1-\frac{\gamma^{d-1}}{N_c})^{-d+1}, & \text{se } \frac{\gamma^{d-1}}{N_c} < \frac{1}{d} \end{cases} \tag{60}$$

E' evidente che tale bound non dipende dalla policy utilizzata (a patto che sia ammissibile e si possa arrivare in qualsiasi punto del terreno di ricerca in al più d passi), ma bensì da:

- il tipo di aggiornamento sulle predizioni, il quale deve soddisfare l'Assunzione 2. In questo caso vale per un modello Markoviano di evasore (evasore non intelligente)
- il numero di FOV necessari per coprire l'intero terreno di ricerca
- il numero di Caselle attive N_c.

Va da se che gli ultimi 2 punti sono influenzati dall'angolo della telecamera utilizzato ad ogni passo (1,65°) e dalla risoluzione dei FOV.
Se si aumenta l'angolo è ovvio che per coprire l'intero terreno di ricerca occorrono meno FOV, quindi d diminuisce. Risulta anche vero, però, che esagerando troppo con lo spostamento minimo delle telecamere non si è più in grado di controllare alcune caselle, diminuendo in questo modo il numero di caselle attive e conseguentemente aumentando il bound.
E' altresì ovvio che a parità di caselle attive, diminuendo l'angolo, aumenta d e quindi il bound su E[$\mathbf{T}^*$].
Dopo svariate prove si è notato che non si riuscirà mai a raggiungere il primo bound, in quanto:

$$\frac{\gamma^{d-1}}{N_c} \geq \frac{1}{d} \Rightarrow \frac{d\gamma^{d-1}}{N_c} \geq 1 \tag{61}$$

Tale disuguaglianza risulta impossibile visto che $\gamma \in [0,1]$ e $\frac{d}{N_c}<1$: per questo motivo si ottiene un bound molto conservativo. Infatti, verificata l'ipotesi $\frac{\gamma^{d-1}}{N_c} \leq \frac{1}{d}$ e, tenendo conto che si è all'interno del foglio, si ha che:

- d=40
- $\gamma = 0.0204$ nel caso peggiore (massimo numero adiacenti)
- Nc=4400

allora il valore del bound è E[$\mathbf{T}^*$]$\leq 1.4560 \cdot 10^{71}$.
Se si diminuisce il numero di FOVs per esplorare l'intera area non si ottiene un miglioramento significativo del bound che, risulta essere uguale per tutte le policy. Il risultato ottenuto è quindi del tutto inutile ai fini pratici, sebbene dimostri la capacità di catturare l'evasore mediamente in un tempo finito.
Consapevoli di tale risultato si è deciso di calcolare il valore eurstico di E[$\mathbf{T}^*$] per ogni policy adottando, **simulazioni di MONTE CARLO** per descrivere la variabile aleatoria $\mathbf{T}^*$.

8 Simulazioni di Monte Carlo

Il metodo di Monte Carlo è usato per trarre stime attraverso simulazioni. Si basa su un algoritmo che genera una serie di prove tra loro incorrelate, le quali seguono la distribuzione di probabilità che si suppone abbia il fenomeno da indagare.
Si sono eseguite 1000 simulazioni al variare della policy e della velocità raggiungibile dall'evasore. In ogni simulazione si registra il numero di passi necessari per catturare l'evasore ricominciando il gioco quando questo viene catturato. Tutte le volte che il gioco riprende si cambiano le posizioni iniziali dei pursuer e dell'intruso. I dati salvati permettono di dare una descrizione euristica della variabile aleatoria $\mathbf{T}^*$.
Le simulazioni utilizzano il modello Markoviano per l'evasore, variandone i parametri secondo:

1. $|A(x)| = \{8, 24, 48, 80\}$
2. e di conseguenza $\rho = \frac{1}{|A(x)|+1}$

Ciò influenza la velocità dell'evasore e l'aggiornamento della matrice 'Predizione'. Si suppone, infatti, che i pursuer siano a conoscenza di tali parametri. Le velocità tra telecamera e pursuer risultano compatibili solo quando $|A(x)| = 48$, come spiegato nella sezione (3.1.1), infatti, al variare di $|A(x)|$ si raggiungono velocità massime rispetto al punto 1. di $v_e = \{0.2, 0.4, 0.6, 0.8\}$ $[\frac{m}{s}]$.
Prima di iniziare un analisi approfondita delle simulazioni si mostra come la densità di probabilità della funzione $\mathbf{T}^*$, utilizzando la policy LOCALMAX, sia influenzata dal variare della velocità dell'intruso. Di seguito verrà disegnato in blu, $|A(x)| = 80$, in verde $|A(x)| = 48$, in rosso $|A(x)| = 24$ e in magenta $|A(x)| = 8$.

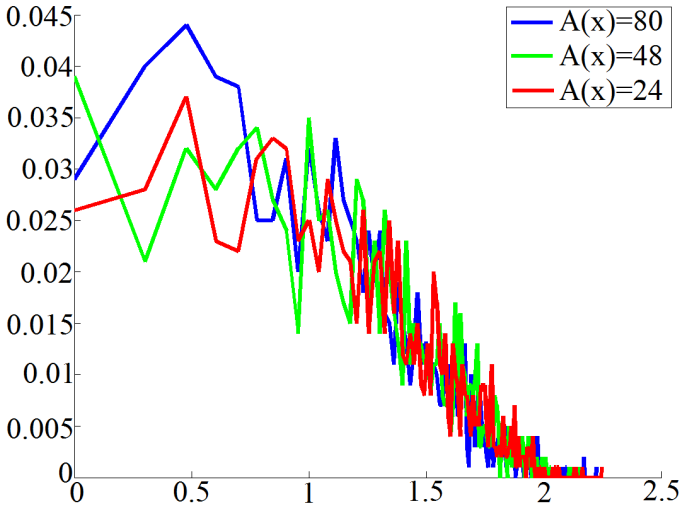

Figura 26: densità di probabilità $\mathbf{T}^*$ in scala logaritmica

$\lvert A(x)\rvert$	$E[\mathbf{T}^*]$	$\sqrt{VAR[\mathbf{T}^*]}$	$max[\mathbf{T}^*]$
2	27.562	23.85	177
3	26.874	22.94	145
4	24.325	22.46	168

Tabella 3: Tabella riassuntiva policy LOCALMAX

Si nota come la media E[$\mathbf{T}^*$] e la VAR[$\mathbf{T}^*$] diminuiscano all'aumentare della velocità dell'evasore, in pieno accordo con il paper [2]. Questo è dovuto al fatto che più aumenta la velocità e più è probabile che l'intruso capiti all'interno di un FOV.

Si evidenzia, in via del tutto euristica, che se si eseguissero infinite simulazioni la densità di probabilità si accosterebbe a quella di un V.A geometrica, la quale presenta un asintoto orizzontale in 0, garantendo di prendere l'evasore in un tempo infinito con probabilità nulla.

Per poter confrontare le diverse policy $\bar{g}$ viene utilizzata la funzione di distribuzione $F_{\bar{g}}(t) = P_{\bar{g}}(\mathbf{T}^* \leq t)$, fissando un tempo di $t = T_{max} = 100$ passi al fine di verificare quale è la probabilità di catturare l'evasore entro 25 secondi,

al variare di $|A(x)|$.

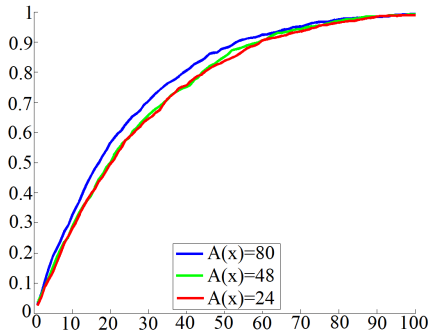

Figura 27: Policy LOCALMAX

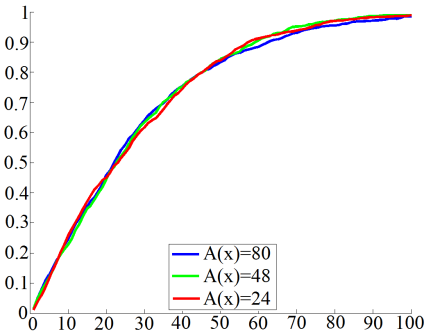

Figura 28: Policy TOTALMAX

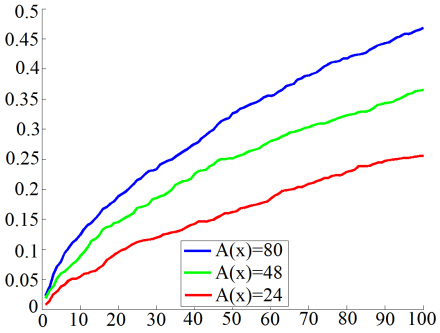

Figura 29: Policy LOCK

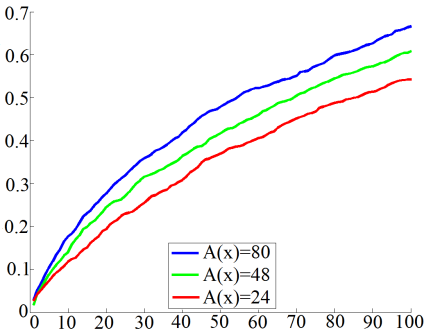

Figura 30: Policy RANDOM

$\|A(x)\|$	$E[\mathbf{T}^*]$	$\sqrt{VAR[\mathbf{T}^*]}$	$max[\mathbf{T}^*]$	$P(\mathbf{T}^* \leq 100)$
2	27.562	23.85	177	0.989
3	26.874	22.94	145	0.991
4	24.325	22.46	168	0.992

Tabella 4: Tabella riassuntiva policy LOCALMAX

$\|A(x)\|$	$E[\mathbf{T}^*]$	$\sqrt{VAR[\mathbf{T}^*]}$	$max[\mathbf{T}^*]$	$P(\mathbf{T}^* \leq 100)$
2	28.8250	24.19	177	0.989
3	28.4790	22.61	167	0.989
4	29.0270	24.95	168	0.984

Tabella 5: Tabella riassuntiva policy TOTALMAX

$\|A(x)\|$	$E[\mathbf{T}^*]$	$\sqrt{VAR[\mathbf{T}^*]}$	$max[\mathbf{T}^*]$	$P(\mathbf{T}^* \leq 100)$
2	788.4470	1235.55	12934	0.225
3	390.5290	551.08	5439	0.365
4	225.9480	300.78	2036	0.468

Tabella 6: Tabella riassuntiva policy LOCK

$\|A(x)\|$	$E[\mathbf{T}^*]$	$\sqrt{VAR[\mathbf{T}^*]}$	$max[\mathbf{T}^*]$	$P(\mathbf{T}^* \leq 100)$
2	243.0490	404.12	3368	0.542
3	156.5090	248.37	2324	0.608
4	115	164.73	1286	0.666

Tabella 7: Tabella riassuntiva policy RANDOM

Dalle precedenti immagini e dalle rispettive tabelle si nota che, nel caso di evasore non intelligente, la policy migliore è la policy LOCALMAX con $|A(x)| = 4$, in quanto permette la cattura dell'evasore con una probabilità di 0.992 entro 25 secondi. Le policy LOCALMAX e TOTALMAX sono, inoltre, poco influenzate da minime variazioni del parametro $|A(x)|$. Probabilmente si aumenterebbero le prestazioni di quest'ultima policy se si pesasse la cella con la probabilità più alta di trovare l'evasore con il tempo necessario per raggiungerla.

Policy meno intelligenti riflettono scarse prestazioni: si evince, infatti, che la peggiore tra le policy è LOCK con evasore lento $|A(x)| = 2$. Questa policy, tuttavia ha una grande variabilità nelle prestazioni all'aumentare della velocità dell'intruso: è ,infatti, più probabile che questo ricada in un FOV.

La policy RANDOM non presenta grandi prestazioni sebbene sia migliore della policy LOCK, ma nettamente peggiore rispetto alla policy TOTALMAX.

Avendo indentificato la migliore tra tutte le policy, per evasore non intelligente, si è voluto utilizzarla anche nel caso di evasore intelligente.

Per evasore intelligente si intende un evasore che è a conoscenza della posizione, 'ad un passo', che assumeranno i pursuer e, se possibile, cercherà di evitare la cattura.

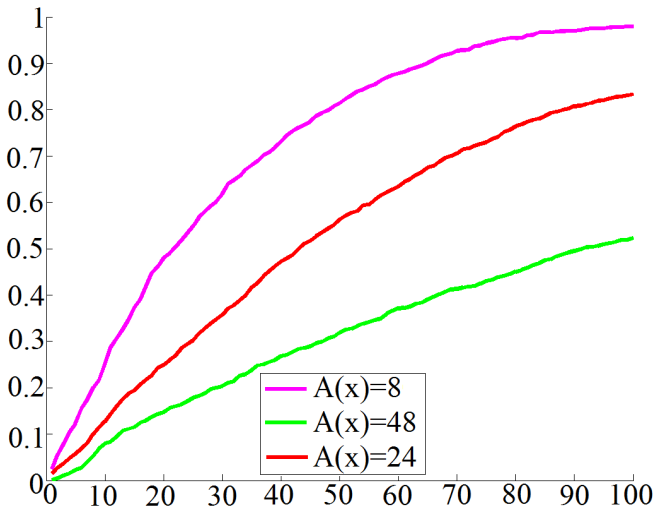

Figura 31: Policy LOCALMAX con EVASORE INTELLIGENTE

$\|A(x)\|$	$E[\mathbf{T}^*]$	$\sqrt{VAR[\mathbf{T}^*]}$	$max[\mathbf{T}^*]$	$P(\mathbf{T}^* \leq 100)$
1	29.7770	26.46	208	0.979
2	58.1780	54.29	522	0.833
3	146.6030	159.54	1242	0.523

Tabella 8: Tabella riassuntiva policy LOCALMAX INTELLIGENTE

Si nota fin da subito la grande variabilità di prestazioni relazionata alla velocità dell'evasore. Questo è dovuto al fatto che se l'intruso è intelligente e molto più veloce dei pursuer, questi lo prendono con molta difficoltà. Con $|A(x)| >> 3$ l'evasore non viene mai catturato vedi sezione (3.1.1).
Rendendo compatibili le velocità l'unica chance che hanno i pursuers per prendere l'evasore è quella di chiuderlo in un angolo del terreno di ricerca, ecco perchè con $|A(x)| = 3$ le probabilità di catturare l'evasore sono così basse (0.523). Tuttavia, mano a mano che la velocità massima dell'evasore diminuisce, questo può essere preso molto più facilmente dalla telecamera, già con $|A(x)| = 2$ la probabilità subisce un sostanziale incremento.

Per tutte le policy vale la pena osservare anche il comportamento di E[$\mathbf{T}^*$], la quale fornisce l'informazione riguardo il tempo medio di cattura all'interno del Testbed. Visto che la miglior policy è LOCALMAX, questa presenta una media più bassa rispetto alle altre, infatti, la cattura avviene mediamente in 24 passi pari ad un tempo di 6 secondi. La peggiore tra le policy, invece, ha un tempo medio di cattura di 197 secondi.
Stando a quanto spiegato conviene utilizzare la policy LOCALMAX nel caso in cui non si conosca il livello di intelligenza dell'evasore utilizzando una telecamera più veloce di quest'ultimo.
Nella realtà si sono fatte delle prove a capione per vedere se, con la policy LOCALMAX, il tempo medio di cattura era paragonabile a quello delle simulazioni. La risposta ha dato esito positivo dal momento in cui le telecamere, utilizzando il modello della simulazione, si muovono allo stesso modo dell'algoritmo.

9 Conclusioni e Sviluppi futuri

Grazie allo studio effettuato si è arrivati ad una possibile soluzione del problema di videosorvegliare un terreno bidimensionale. Tale soluzione non garantisce di catturare sicuramente l'evasore, ma la probabilità che ciò non avvenga in un tempo finito è nulla.
Si è visto come la miglior policy da utilizzare sia la LOCALMAX, soprattutto se non si conosce il livello di intelligenza dell'evasore. Tale policy, essendo in pratica persistente, risulta facilmente scalabile, in quanto i pursuer massimizzano semplicemente la probabilità di trovare l'evasore al passo successivo.

I possibili sviluppi futuri possono comprendere:

- l'unione di tale soluzione con un algoritmo di Tracking al fine di non perdere l'evasore e ottenerne un modello più dettagliato
- la possibilità di videosorvegliare un area tridimensionale attraverso telecamere montate su elicotteri radiocomandati da PC, aggiornando contemporaneamente il terreno di ricerca in caso di ostacoli (edifici, alberi, lampioni ecc.)
- l'implementazione dell'algoritmo in maniera decentralizzata tramite lo sviluppo di diagrammi di Voronoi adattati alla dinamica delle telecamere analizzato in seguito.

9.1 Soluzione decentralizzata del problema

La soluzione trovata per il problema di videosorvegliare un ambiente bidimensionale è chiaramente centralizzata, in quanto per aggiornare la matrice di 'Predizione' si deve conoscere la posizione di tutte le telecamere. Inoltre, una volta ottenuto l'aggiornamento di tale matrice, questa deve essere passata alle

telecamere al fine di calcolare la nuova posizione dei pursuer.
Si vorrebbe decentralizzare il problema in modo da renderlo facilmente scalabile e ridurre al minimo il passaggio di informazioni tra le telecamere.
L'idea è quella di dividere il terreno di ricerca in N_p zone, tutte con la stessa area, utilizzando un diagramma di Voronoi [7] che sia compatibile con le posizioni delle telecamere. In questo modo ogni pursuer controlla solamente un area designata e per ognuna di queste zone si ha che:

$$\sum_{x \in Z_i} p_e(x, Y_{t-1}) = \frac{1}{N_p} \quad \forall t \tag{62}$$

Dove in Z_i vi sono tutte le caselle appartenenti alla i-esima zona, in modo che sull'intero terreno di ricerca sia ancora valida la (46).
In questo modo è possibile aggiornare la singola zona e rendere l'algoritmo decentralizzato. Predividendo l'intero terreno di ricerca si risolve, inoltre, automaticamente anche il problema dell'intersezione dei FOV.

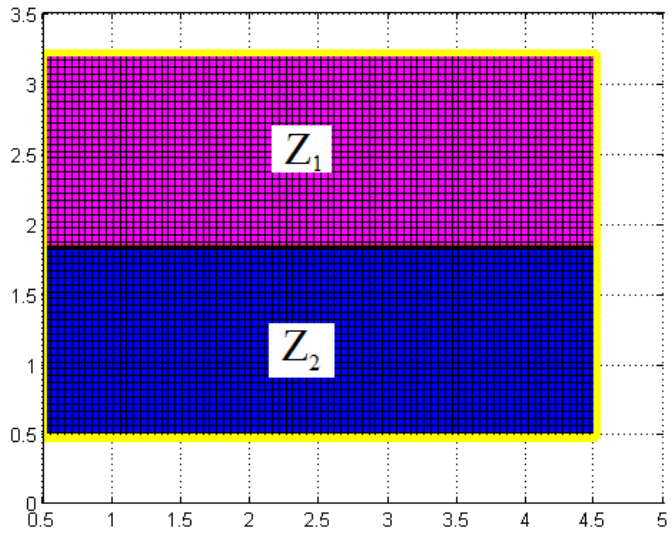

Figura 32: divisione terreno tramite algoritmo di Voronoy

L'algoritmo è sicuramente molto più scalabile, rispetto alla soluzione centralizzata, in quanto se si dovesse rompere una telecamera basterebbe utilizzare l'algoritmo di Voronoi per dividere il terreno in $N_p - 1$ zone e ricominciare la videosorveglianza.
Il tempo a disposizione non è purtroppo stato sufficiente per implementare ques'ultima soluzione.

10 Appendice

10.0.1 Lemma 4

Lemma 4

Dato un insieme a tempo discreto $\mathcal{T}= \{t_0, t_1, ...\}$, con $t_{k+1} \geq t_k$, $k \geq 0$, sia $f : [0,1]^* \rightarrow [0,1]$ definita da $\{u_0, u_1, ...\} \rightarrow \sum_{n=1}^{\infty} t_n u_n(\prod_{k=0,k\neq m}^{n-1}(1-u_k))$. Allora per ogni intero m per il quale $\prod_{k=0,k\neq m}^{\infty}(1-u_k)=0$ si ha:

$$\frac{\partial f}{\partial u_m} = -\sum_{n=m+1}^{\infty}(t_n - t_m)u_n \prod_{k=0,k\neq m}^{n-1}(1-u_k) \leq 0 \tag{63}$$

Proof Lemma 4.

Sia m un intero per il quale $\prod_{k=0,k\neq m}^{\infty}(1-u_k)=0$. Prendendo la derivata parziale di f rispetto a u_m si ottiene:

$$\begin{aligned}\frac{\partial f}{\partial u_m} =& t_m \prod_{k=0}^{m-1}(1-u_k) - \sum_{n=m+1}^{\infty} t_n u_n(\prod_{k=0,k\neq m}^{n-1}(1-u_k)) \\ =& (\prod_{k=0}^{m-1}(1-u_k))(t_m - \sum_{n=m+1}^{\infty} t_n u_n(\prod_{k=m+1}^{n-1}(1-u_k)))\end{aligned} \tag{64}$$

Ma si dimostra facilmente che $\sum_{n=m+1}^{\infty} u_n(\prod_{k=m+1}^{n-1}(1-u_k)) = 1-\prod_{k=m+1}^{\infty}(1-u_k)$, quindi:

$$\begin{aligned}\frac{\partial f}{\partial u_m} =& (\prod_{k=0}^{m-1}(1-u_k))(t_m \prod_{k=m+1}^{\infty}(1-u_k) - \sum_{n=m+1}^{\infty}(t_n - t_m)u_n(\prod_{k=m+1}^{n-1}(1-u_k))) \\ =& -\sum_{n=m+1}^{\infty}(t_n - t_m)u_n(\prod_{k=0,k\neq m}^{n-1}(1-u_k)) + t_m \prod_{k=0,k\neq m}^{\infty}(1-u_k)\end{aligned} \tag{65}$$

dalla quale segue la(63).

10.0.2 Dimostrazione Lemma 3

Per un dato $t \in \mathcal{T}$, $\bar{f}_{\bar{g}}$(t) può essere scritto come:

$$\begin{aligned}\bar{f}_{\bar{g}}(t) &= \sum_{k=0}^{T-1} P_{\bar{g}}(\mathbf{T}^* = t+k | \mathbf{T}^* \geq t) \\ &= \sum_{k=0}^{T-1} P_{\bar{g}}(\mathbf{Y}_{t+k} \notin \mathcal{Y}_{t+k}^{\neg fnd}, \mathbf{Y}_{t+k-1} \in \mathcal{Y}_{t+k-1}^{\neg fnd} | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd})\end{aligned} \tag{66}$$

Prendiamo adesso $\bar{\delta} \in$ [0,δ]. Un lower bound per $\bar{f}_{\bar{g}}$(t) può essere allora costruito vincolando gli eventi sulla probabilità delle misure come segue:

$$\begin{aligned}\bar{f}_{\bar{g}}(t) \geq \sum_{k=0}^{T-1} P_{\bar{g}}(&\mathbf{Y}_{t+k} \notin \mathcal{Y}_{t+k}^{\neg fnd}, \mathbf{Y}_{t+k-1} \in \mathcal{Y}_{t+k-1}^{\neg fnd}, h(\mathbf{Y}_{t+k-1}) \geq \bar{\delta}, \\ &h(\mathbf{Y}_s) < \bar{\delta}, s = t-1, t, ..., t+k-2 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd})\end{aligned} \tag{67}$$

Per ogni n $\in \{0, 1, ..., T-1\}$ si definisce ora:

$$\begin{aligned}u_n = \sum_{k=0}^{T-n-2} & P_{\bar{g}}(\mathbf{Y}_{t+k-1} \in \mathcal{Y}_{t+k-1}^{\neg fnd}, h(\mathbf{Y}_{t+k-1}) \geq \bar{\delta}, h(\mathbf{Y}_s) < \bar{\delta}, \\ & s = t-1, .., t+k-2 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd}) \\ & + P_{\bar{g}}(\mathbf{Y}_{t+T-n-2} \in \mathcal{Y}_{t+T-n-2}^{\neg fnd}, h(\mathbf{Y}_s) < \bar{\delta}, \\ & s = t-1, t, ..., t+T-n-3 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd})\end{aligned} \tag{68}$$

Si mostra in seguito che $\bar{f}_{\bar{g}}$(t)$\geq \bar{\delta} u_0$. Per questo motivo si nota che per ogni τ $\in \{t, t+1, ..., t+T-1\}$,

$$
\begin{aligned}
&P_{\bar{g}}(\mathbf{Y}_\tau \notin \mathcal{Y}_\tau^{\neg fnd}, \mathbf{Y}_{\tau-1} \in \mathcal{Y}_{\tau-1}^{\neg fnd}, h(\mathbf{Y}_{\tau-1}) \geq \bar{\delta}, h(\mathbf{Y}_s) < \bar{\delta},\\
&s = t-1, t, ..., \tau-2 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd}) =\\
&\sum_{\substack{\mathbf{Y}_{\tau-1} \in \mathcal{Y}_{\tau-1}^{\neg fnd} \\ h(\mathbf{Y}_{\tau-1}) \geq \bar{\delta} \\ h(\mathbf{Y}_s) < \bar{\delta}, s = t-1, t, ..., \tau-2}} \Big(\sum_{\substack{Y \in \mathcal{Y} \\ Y_\tau \notin \mathcal{Y}_\tau^{\neg fnd}}} P_{\bar{g}}(\mathbf{Y}_\tau = Y_\tau | \mathbf{Y}_{\tau-1} = Y_{\tau-1}) \Big)\\
&P_{\bar{g}}(\mathbf{Y}_{\tau-1} = Y_{\tau-1} | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd})
\end{aligned}
\tag{69}
$$

Dove Y_τ denota la sequenza che consiste elle misurazioni in $Y_{\tau-1}$, seguite da Y, e Y_s, $s < \tau - 1$, la sequenza che consiste delle prime misure in Y_τ. Ma

$$
\sum_{\substack{Y \in \mathcal{Y} \\ Y_\tau \notin \mathcal{Y}_\tau^{\neg fnd}}} P_{\bar{g}}(\mathbf{Y}_\tau = Y_\tau | \mathbf{Y}_{\tau-1} = Y_{\tau-1}) \tag{70}
$$

é precisamente la probabilità $h(Y_{\tau-1})$ di trovare l'evasore per la prima volta al tempo τ, date le misure $\mathbf{Y}_{\tau-1}$=$Y_{\tau-1} \in \mathcal{Y}_{\tau-1}^{\neg fnd}$. Quindi

$$
\begin{aligned}
&P_{\bar{g}}(\mathbf{Y}_\tau \notin \mathcal{Y}_\tau^{\neg fnd}, \mathbf{Y}_{\tau-1} \in \mathcal{Y}_{\tau-1}^{\neg fnd}, h(\mathbf{Y}_{\tau-1}) \geq \bar{\delta}, h(\mathbf{Y}_s) < \bar{\delta}, s = t-1, t, ..., \tau-2\\
&| \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd})\\
&= \sum_{\substack{\mathbf{Y}_{\tau-1} \in \mathcal{Y}_{\tau-1}^{\neg fnd} \\ h(\mathbf{Y}_{\tau-1}) \geq \bar{\delta} \\ h(\mathbf{Y}_s) < \bar{\delta}, s = t-1, t, ..., \tau-2}} \Big(\sum_{\substack{Y \in \mathcal{Y} \\ Y_\tau \notin \mathcal{Y}_\tau^{\neg fnd}}} h(Y_{\tau-1}) \Big) P_{\bar{g}}(\mathbf{Y}_{\tau-1} = Y_{\tau-1} | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd})\\
&\geq \bar{\delta} P_{\bar{g}}(\mathbf{Y}_{\tau-1} \in \mathcal{Y}_{\tau-1}^{\neg fnd}, h(\mathbf{Y}_{\tau-1}) \geq \bar{\delta}, h(\mathbf{Y}_s) < \bar{\delta},\\
&s = t-1, t, ..., \tau-2 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd})
\end{aligned}
\tag{71}
$$

Usando questo in (67) si conclude che:

$$\begin{aligned}
&\bar{f}_{\bar{g}}(t) \geq \bar{\delta} \sum_{k=0}^{T-1} P_{\bar{g}}(\mathbf{Y}_{t+k-1} \in \mathcal{Y}_{t+k-1}^{\neg fnd}, h(\mathbf{Y}_{t+k-1}) \geq \bar{\delta}, h(\mathbf{Y}_s) < \bar{\delta}, \\
&s = t-1, .., t+k-2 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd}) \\
&= \bar{\delta} \sum_{k=0}^{T-2} P_{\bar{g}}(\mathbf{Y}_{t+k-1} \in \mathcal{Y}_{t+k-1}^{\neg fnd}, h(\mathbf{Y}_{t+k-1}) \geq \bar{\delta}, h(\mathbf{Y}_s) < \bar{\delta}, \\
&s = t-1, .., t+k-2 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd}) \\
&+ \bar{\delta} P_{\bar{g}}(\mathbf{Y}_{t+T-2} \in \mathcal{Y}_{t+T-2}^{\neg fnd}, h(\mathbf{Y}_{t+T-2}) \geq \bar{\delta}, h(\mathbf{Y}_s) < \bar{\delta}, \\
&s = t-1, .., t+T-3 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd})
\end{aligned} \tag{72}$$

Ora dovuto alle ipotesi del lemma, per qualsiasi realizzazione delle variabili aleatorie $\mathbf{Y}_{t-1}$, $\mathbf{Y}_t$,..., $\mathbf{Y}_{t+T-2}$, se $h(\mathbf{Y}_s) < \bar{\delta} \leq \delta$, s=t-1,t,...,t+T-3, si deve avere $h(\mathbf{Y}_{t+T-2}) \geq \delta \geq \bar{\delta}$. Si può quindi rimuovere questo vincolo in (72) e ottenere

$$\bar{f}_{\bar{g}}(t) \geq \bar{\delta} u_0 \tag{73}$$

Dalla definizione di u_n risulta chiaro che $u_{T-1} = 1$.. Si procede mostrando che $u_n \geq (1-\bar{\delta}) u_{n+1}$, $n \in \{0, 1, ..., T-2\}$, dal quale si conclude che $\bar{f}_{\bar{g}}(t) \geq \bar{\delta}(1-\bar{\delta})^{T-1}$. Per un dato $\tau \in \{t, t+1, ..., t+T-1\}$,

$$\begin{aligned}
&P_{\bar{g}}(\mathbf{Y}_\tau \in \mathcal{Y}_\tau^{\neg fnd}, h(\mathbf{Y}_s) < \bar{\delta}, s = t-1, t, ..., \tau-1 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd}) \\
&= \sum_{\substack{\mathbf{Y}_{\tau-1} \in \mathcal{Y}_{\tau-1}^{\neg fnd} \\ h(\mathbf{Y}_s) < \bar{\delta}, s = t-1, t, ..., \tau-1}} \Big(\sum_{\substack{Y \in \mathcal{Y} \\ Y_\tau \in \mathcal{Y}_\tau^{\neg fnd}}} P_{\bar{g}}(\mathbf{Y}_\tau = Y_\tau | \mathbf{Y}_{\tau-1} = Y_{\tau-1}) \Big) \\
&P_{\bar{g}}(\mathbf{Y}_{\tau-1} = Y_{\tau-1} | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd})
\end{aligned} \tag{74}$$

Dove Y_τ denota la sequenza che consiste delle misure in $Y_{\tau-1}$ seguite da y, e Y_s, $s < \tau - 1$, la sequenza consiste delle prime s misure in Y_τ. Ma

$$\sum_{\substack{Y \in \mathcal{Y} \\ Y_\tau \in \mathcal{Y}_\tau^{\neg fnd}}} P_{\bar{g}}(\mathbf{Y}_\tau = Y_\tau | \mathbf{Y}_{\tau-1} = Y_{\tau-1}) \tag{75}$$

è precisamente la probabilità 1-$h(Y_{\tau-1})$ di non trovare l'evasore per la prima volta al tempo τ, date la misurazioni $Y_{\tau-1} = Y_{\tau-1} \in \mathcal{Y}_{\tau-1}^{\neg fnd}$. Quindi

$$\begin{aligned}
&P_{\bar{g}}(\mathbf{Y}_\tau \in \mathcal{Y}_\tau^{\neg fnd}, h(\mathbf{Y}_s) < \bar{\delta}, s = t-1, t, ..., \tau-1 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd}) \\
&= \sum_{\substack{\mathbf{Y}_{\tau-1} \in \mathcal{Y}_{\tau-1}^{\neg fnd} \\ h(\mathbf{Y}_s) < \bar{\delta}, s = t-1, t, ..., \tau-1}} (1 - h(Y_{\tau-1})) P_{\bar{g}}(\mathbf{Y}_{\tau-1} = Y_{\tau-1} | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd}) \\
&\geq (1-\bar{\delta}) P_{\bar{g}}(\mathbf{Y}_{\tau-1} \in \mathcal{Y}_{\tau-1}^{\neg fnd}, h(\mathbf{Y}_s) < \bar{\delta}, s = t-1, t, ..., \tau-1 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd})
\end{aligned} \tag{76}$$

Applicando la formula sopra per $\tau = t + T - n - 2$, si conclude che

$$\begin{aligned}
u_n \geq & \sum_{k=0}^{T-n-2} P_{\bar{g}}(\mathbf{Y}_{t+k-1} \in \mathcal{Y}_{t+k-1}^{\neg fnd}, h(\mathbf{Y}_{t+k-1}) \geq \bar{\delta}, h(\mathbf{Y}_s) < \bar{\delta}, \\
& s = t-1, .., t+k-2 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd}) \\
& + (1-\bar{\delta}) P_{\bar{g}}(\mathbf{Y}_{t+T-n-3} \in \mathcal{Y}_{t+T-n-3}^{\neg fnd}, h(\mathbf{Y}_s) < \bar{\delta}, \\
& s = t-1, t, ..., t+T-n-3 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd}) \\
& \geq (1-\bar{\delta}) (\sum_{k=0}^{T-n-3} P_{\bar{g}}(\mathbf{Y}_{t+k-1} \in \mathcal{Y}_{t+k-1}^{\neg fnd}, h(\mathbf{Y}_{t+k-1}) \geq \bar{\delta} \\
& , h(\mathbf{Y}_s) < \bar{\delta}, s = t-1, .., t+k-2 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd}) \\
& + P_{\bar{g}}(\mathbf{Y}_{t+T-n-3} \in \mathcal{Y}_{t+T-n-3}^{\neg fnd}, h(\mathbf{Y}_{t+T-n-3}) \geq \bar{\delta} \\
& , h(\mathbf{Y}_s) < \bar{\delta}, s = t-1, .., t+T-n-3 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd}) \\
& + P_{\bar{g}}(\mathbf{Y}_{t+T-n-3} \in \mathcal{Y}_{t+T-n-3}^{\neg fnd}, h(\mathbf{Y}_s) < \bar{\delta}, \\
& s = t-1, .., t+T-n-3 | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd})) = (1-\bar{\delta}) u_{n+1}
\end{aligned} \tag{77}$$

Da questa e (73) si conclude che

$$\begin{aligned} &\bar{f}_{\bar{g}}(t) \geq \bar{\delta}(1-\bar{\delta})^{T-1} u_{T-1} \\ &= \bar{\delta}(1-\bar{\delta})^{T-1} P_{\bar{g}}(\mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd} | \mathbf{Y}_{t-1} \in \mathcal{Y}_{t-1}^{\neg fnd}) = \bar{\delta}(1-\bar{\delta})^{T-1} \end{aligned} \tag{78}$$

Fino a che quella sopra è verificata da qualsiasi si $\bar{\delta} \in [0,\delta]$, ha anche $\bar{f}_{\bar{g}}(t) \geq \max_{\bar{\delta} \in [0,\delta]} \bar{\delta}(1-\bar{\delta})^{T-1} = \epsilon$ con ϵ come in (40)

Riferimenti bibliografici

[1] João P. Hespanha, Hyoun Jin Kim, Shankar Sastry (1999) *MULTIPLE-AGENT PROBABILISTIC PURSUIT-EVASION GAMES*, University of California, Berkeley. *Conferece*: Decision and Control, 1999. Proceedings of the 38th IEEE, Phoenix, AZ , USA. pp 2432 - 2437 vol.3. DOI: 10.1109/CDC.1999.831290

[2] René Vidal, Omid Shakernia, Jin Kim, David Hyunchul Shim, Shankar Sastry *Probabilistic PursuitEvasion Games: Theory,Implementation, and Experimental Evaluation* **Transaction on:** Robotics and Automation, IEEE. pp 662 - 669. DOI:10.1109/TRA.2002.804040

[3] David Sturzenegger (2010) *Multi Target Tracking Using Multiple Pan-Tilt-Zoom Cameras*, University ETH, Zurich. Master Thesis pp.57

[4] Baseggio Mauro, Merlo Pierangelo, Pozzi Mauro (2010) *Coordinazione distribuita di telecamere per patrolling e tracking perimetrale*, Università degli studi di Padova, Padova. pp 78

[5] Smith A.R. (1978) *Color gamut transform pairs. International Confer- ence on Computer Graphics and Interactive Techniques. Conference:* SIGGRAPH '78 Computer graphics and interactive techniques pp 20-25, New York, NY, USA ©1978. DOI:10.1145/800248.807361

[6] http://en.wikipedia.org/wiki/HSL and HSV.

[7] http://it.wikipedia.org/wiki/DiagrammadiVoronoi

11 Ringraziamenti

Vorrei, innanzittutto, ringraziare il professor Luca Schenato per avermi dato l'opportunità di sviluppare tale progetto presso l'ETH di Zurigo e per avermi aiutato nella stesura della tesi.
Ringrazio il professor John Lygeros, il quale che ha saputo darmi ottimi spunti e consigli preziosi.
Infine, un sentito ringraziamento va al Dr.Davide Raimondo e al Dr.Riccardo Porreca, non solo per avermi aiutato a risolvere il problema, ma anche per avermi accolto come un amico nella mia breve permanenza in Svizzera.