Laboratorio

Realtà aumentata: Augmented Reality Player

Corso di Smart City e Tecnologie Mobili

Università di Bologna

Dipartimento di Informatica — Scienza e Ingegneria

Simone Buoncompagni, Dario Maio

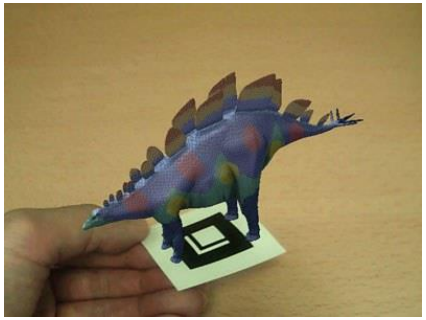
Panoramica dell'esercitazione

- Tool di realtà aumentata: Augmented Reality Browser
- Panoramica su SDK, framework e librerie
- Framework di lavoro: AForge.NET
- Attività di riconoscimento di un marker
- Calcolo della posa di un marker rispetto alla camera del device
- Visualizzazione delle informazioni in realtà aumentata

Creazione di applicazioni AR

• Indipendentemente dalla piattaforma utilizzata (Android, iOS, Windows, Linux, ecc.) e dalla tipologia di device (smartphone, tablet, pc, ecc.) una comune applicazione di Visual AR è il risultato della composizione di due task fondamentali:

- riconoscimento di punti fiduciali posti nell'ambiente e/o geolocalizzazione;
- creazione e visualizzazione di informazioni grafiche coerenti con posizione e orientamento del dispositivo rispetto all'ambiente circostante.



• Sviluppare un'applicazione AR è piuttosto complicato: numerosi strumenti di sviluppo sono stati proposti per facilitare la creazione e la fruizione di applicazioni AR anche da parte di utenti senza specifiche competenze informatiche.

Augmented Reality Browser

• Applicazioni di realtà aumentata per dispositivi mobili (smartphone, tablet, ecc.) dotati di opportuni sensori (fotocamera, accelerometro, bussola e rilevatore GPS).

• Combinazione sinergica dei dati acquisiti dai sensori con i contenuti reperibili sul web: un AR Browser consente di "navigare" la realtà circostante in analogia con quanto avviene con un classico browser internet.

• Gli strati informativi virtuali e interattivi (testi, immagini, video, animazioni 3D) sono sovrapposti su ciò che si sta osservando attraverso il device (es. informazioni sui ristoranti più vicini o sui luoghi dove si tengono eventi).

• Le case produttrici mettono a disposizione opportuni strumenti di sviluppo associati (solitamente a pagamento) per personalizzare il proprio strato informativo virtuale e per creare funzioni di rilevamento/riconoscimento di elementi custom (codici a barre, oggetti generici, ecc.).

AR Browser: Layar (https://www.layar.com/)

• Uno dei primi AR Browser ad essere lanciati sul mercato.

• Con oltre 1 milione di utenti attivi, Layar è stato costantemente espanso aggiungendo nuove feature e circa 2000 possibili layer.

• Mediante uso di tecniche di computer vision e acquisizione di opportuni dati di posizione, l'applicazione ricerca e riconosce gli oggetti su cui visualizzare graficamente le informazioni in realtà aumentata.

AR Browser: Junaio (http://www.junaio.com/)

• Prodotto dalla compagnia tedesca Metaio, dal punto di vista delle feature offerte è molto simile a Layar.

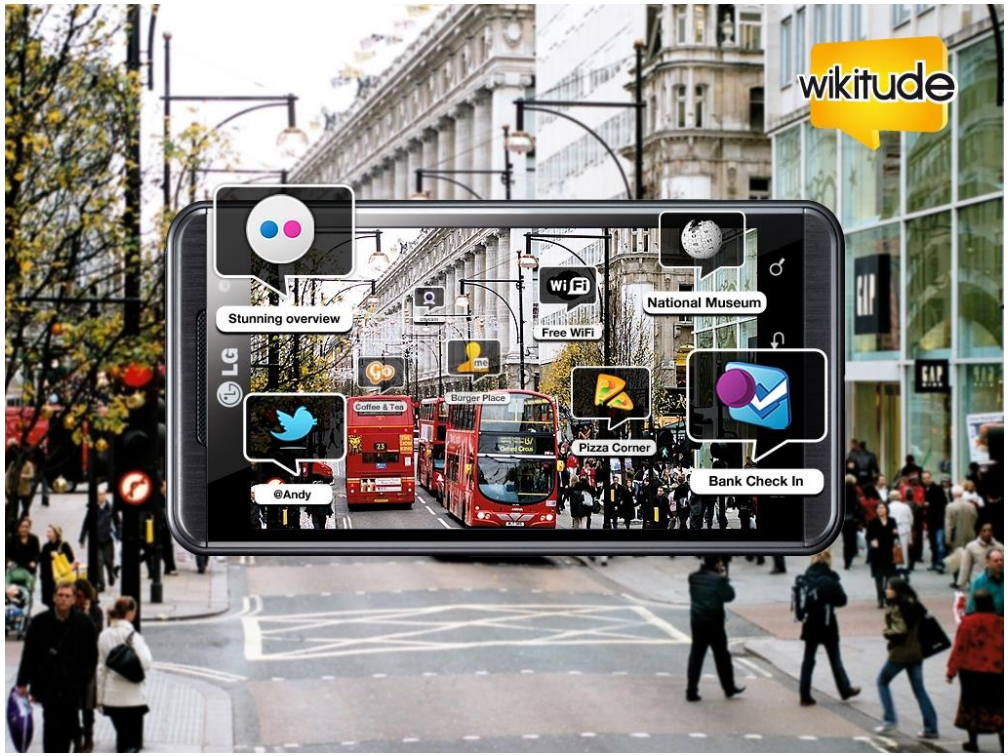
• È stato studiato per favorire una rapida integrazione nel browser dei più recenti sviluppi sul tema AR, al fine di creare una sorta di "AR Hub" (all – in – one solution).

AR Browser: Wikitude (http://www.wikitude.com/)

- È uno più importanti e famosi AR Browser ed è l'unico ad essere disponibile per i principali sistemi operativi su dispositivi mobili (iOS, Android, Symbian, Windows e Blackberry).

• Pur non incorporando i più recenti avanzamenti in tema di AR, esso si distingue favorevolmente da Layar e Junaio grazie alla sua funzione "social".

• Gli strati informativi possono essere condivisi e migliorati attraverso l'interazione con i più comuni social network (Facebook, Twitter, Instagram, Foursquare, ecc.).



Framework per lo sviluppo autonomo

• Gli AR Browser possono essere eseguiti solo su dispositivi mobili e offrono un ambiente che può essere utilizzato così com'è o che può essere arricchito mediante l'aggiunta di componenti custom.

• Gli AR Browser, inoltre, sono studiati per offrire certe tipologie di contenuti ma non sono la soluzione per ogni possibile esigenza di realtà aumentata.

• Per poter sviluppare "in proprio" un'applicazione di realtà aumentata eseguibile anche su un classico PC occorre utilizzare framework e librerie che offrano opportune funzionalità, consentendo al programmatore di non partire "da zero".

• Negli ultimi anni i tool di sviluppo proposti sono stati molteplici, differenziandosi per piattaforme supportate, linguaggi di programmazione impiegati, feature e tipologia di licenza (free o a pagamento).

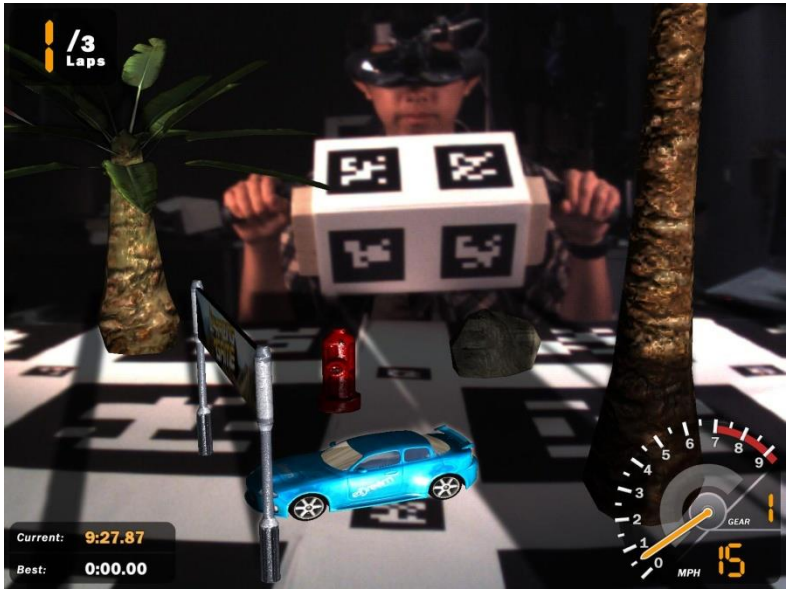
• Panoramica: http://socialcompare.com/en/comparison/augmented-reality-sdks

Esempio di AR framework: Goblin XNA

• Goblin XNA nasce come piattaforma per la ricerca in ambito di interfacce utente 3D, con applicazioni che includono la realtà aumentata, la realtà virtuale e una particolare enfasi verso i giochi.

• È stato sviluppato in linguaggio C# ed è basato su Microsoft XNA Game Studio.

• Disponibilità di funzioni predefinite per il riconoscimento di marker, proiezione di contenuti, ecc.



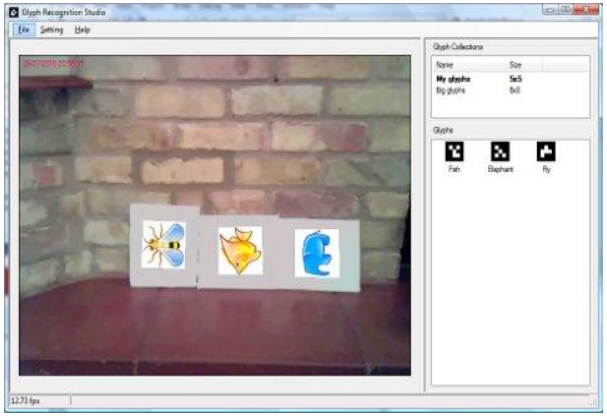
Il framework AForge.NET (http://www.aforgenet.com/framework/)

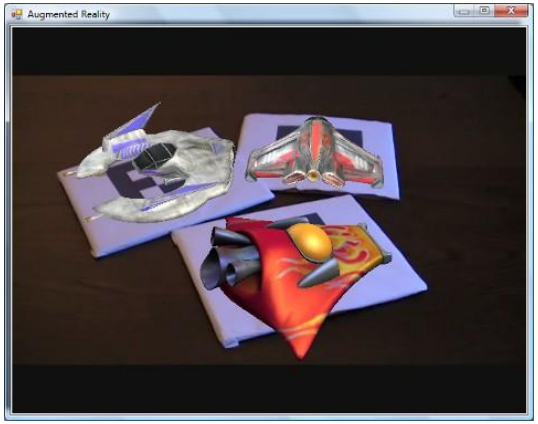
• AForge.NET è un framework open source in linguaggio C# progettato per sviluppatori e ricercatori che operano nel campo della Computer Vision e dell'Intelligenza Artificiale.

• Esso comprende 7 librerie dedicate a svariate tematiche di sviluppo (elaborazione di immagini, reti neurali, algoritmi genetici, logica fuzzy, machine learning, robotica, ecc.).

• Grazie alla sua natura open source, ogni funzione di libreria è completamente accessibile da parte dello sviluppatore.

• Progetto AR open – source: GRATF (*Glyph Recognition And Tracking Framework*)





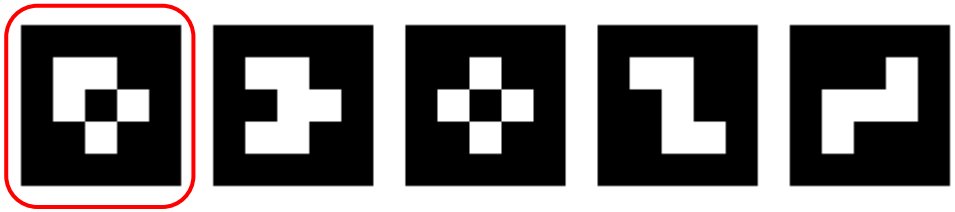
Esercitazione: Augmented Reality Player

• Per la nostra esercitazione faremo ricorso a una versione notevolmente semplificata di GRATF: progetto Augmented Reality Player.

• Evoluzione dell'applicativo Player che riproduce frame provenienti da una sorgente (file video, camera, ecc): http://www.aforgenet.com/framework/samples/video.html .

• Suddivisione in classi di funzionalità per analizzare le principali fasi di elaborazione e utilizzare in modo appropriato la libreria AForge.NET.

• Impiego di particolari marker contenenti un pattern interno: *optical glyph*.



• Obiettivo dell'esercitazione: riconoscere un particolare tipo di optical glyph (nel nostro caso il primo a sinistra) e proiettare su esso informazioni in realtà aumentata.

Augmented Reality Player: classi principali

• **MainForm.cs**: contiene il metodo *videoSourcePlayer_NewFrame(object sender, ref Bitmap image)*, invocato ogni qualvolta viene ricevuto un nuovo frame dalla sorgente (oggetto *Bitmap image* passato per riferimento).

• **MarkerDetection.cs**: classe per il rilevamento delle coordinate di corner dei potenziali marker di interesse.

• **MarkerRecognition.cs**: classe per decodificare i pattern dei marker individuati sulla scena e verificare la corrispondenza con la tipologia di pattern di nostro interesse.

• **PoseEstimation.cs:** classe per la stima della posa del marker rispetto alla camera (calcolo della matrice di trasformazione).

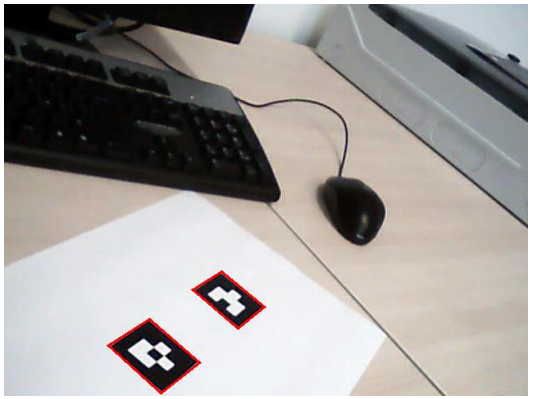
• **ImageAugmentation.cs:** classe per il rendering grafico delle informazioni aumentanti (proiezione di un'immagine e disegno del sistema di riferimento 3D solidale al marker).

Augmented Reality Player: schema

Frame di input

MarkerDetection.cs

Rilevamento presenza marker



MarkerRecognition.cs

Decodifica marker e verifica di congruenza

PoseEstimation.cs
ImageAugmentation.cs

Calcolo posa del marker e proiezione informazioni aumentanti

Frame di output

Augmented Reality Player: codice “main”

```
// Rilevamento di potenziali marker
markerDetectionOperator = new MarkerDetection(image, 30, 48, 48, 40);
markerDetectionOperator.Execute();

if (markerDetectionOperator.AllSquareShapesCorners.Count > 0) // Se sono stati trovati potenziali marker
{   // Decodifica del contenuto del marker
    markerRecognitionOperator = new MarkerRecognition(markerDetectionOperator.GrayscaleImage,
                                                      markerDetectionOperator.AllSquareShapesCorners, 0.80f);
    markerRecognitionOperator.Execute();

     if (markerRecognitionOperator.IsMarkerFound) // Trovato il pattern del marker di nostro interesse
    {   // Calcolo della posa del marker rispetto alla camera
        poseEstimationOperator = new PoseEstimation(markerRecognitionOperator.MarkerCorners,
                                                    image.Width, image.Height);
        poseEstimationOperator.Execute();
        // Proiezione delle informazioni aumentanti
        imageAugmentationOperator = new ImageAugmentation(image, Resources.Globe,
                                                          poseEstimationOperator.MarkerCorners,
                                                          poseEstimationOperator.TransformationMatrix);
        imageAugmentationOperator.Execute();
        image = imageAugmentationOperator.BitmapOutputImage;
    }
}
```

videoSourcePlayer_NewFrame

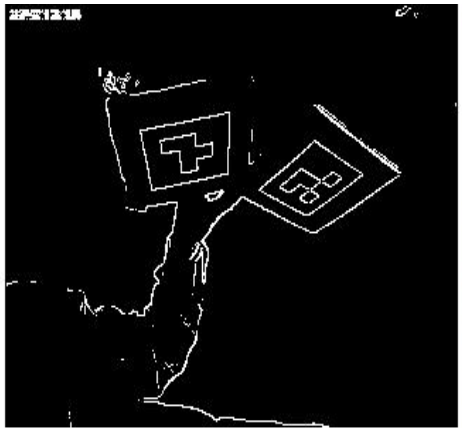
Rilevamento di un marker

• Input: frame video acquisito dalla sorgente.

• Output: insieme di quadruple di punti cartesiani. Ognuna contiene le coordinate dei 4 punti di corner di un marker di potenziale interesse.

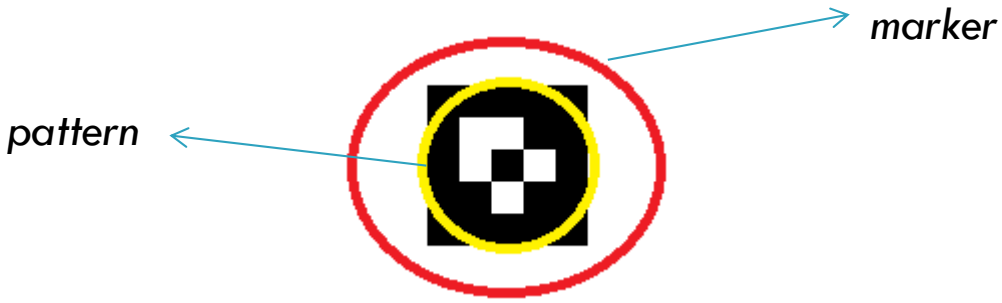
• Principali task di image processing (in ordine): conversione grayscale, rilevamento degli edge, rilevamento di componenti connesse (blob), filtraggio dei blob non significativi (non quadrati e/o con luminosità non coerente a quella dei marker che vogliamo rilevare).






Riconoscimento (decodifica) di un marker

• L'attività di rilevamento restituisce le coordinate di ognuno dei marker rilevati, ma non fornisce informazioni sulla tipologia (pattern) del marker rilevato.



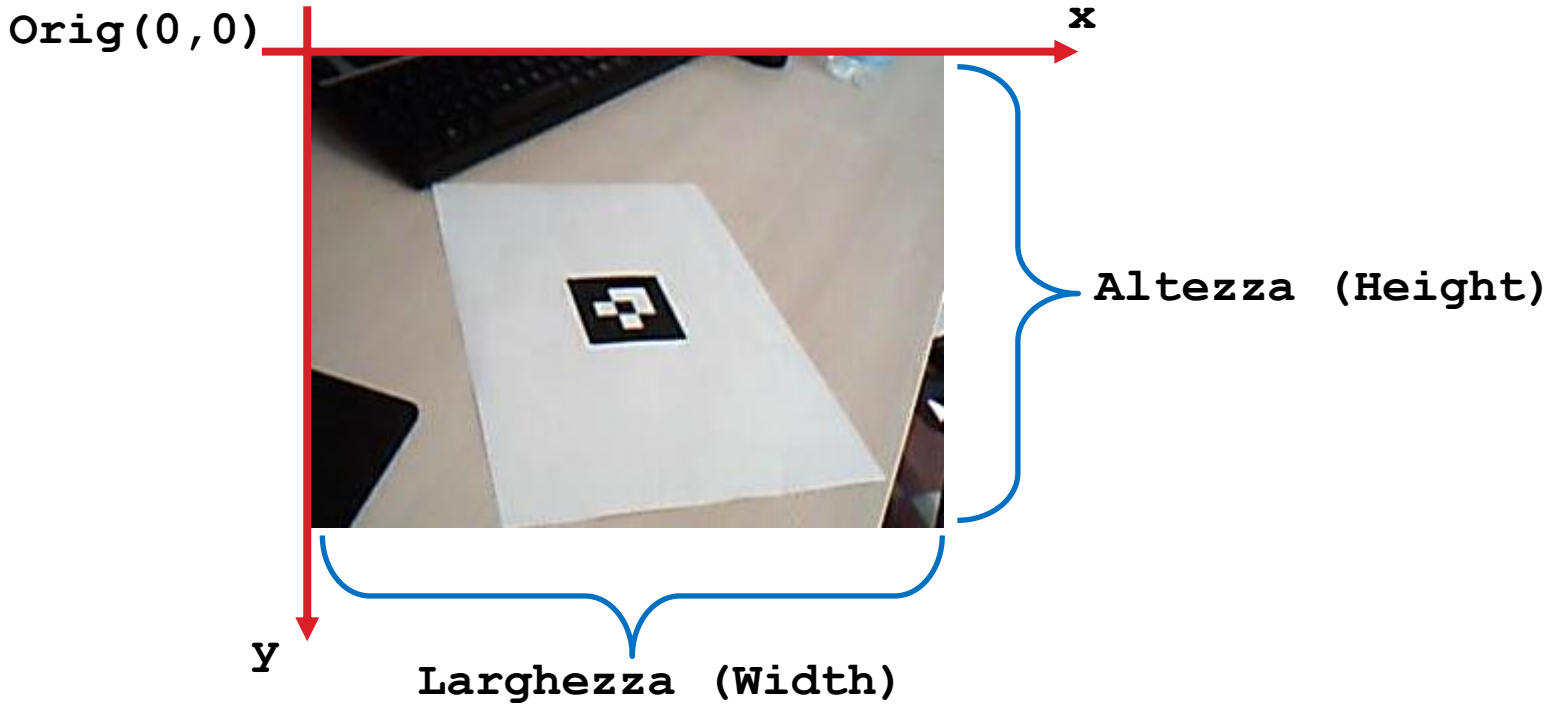
• Il nostro obiettivo consiste nel verificare la presenza sulla scena di uno specifico pattern e proiettare le informazioni in realtà aumentata sull'area del marker corrispondente.

• L'attività di decodifica prende in input le coordinate di tutti i marker rilevati e restituisce in output le coordinate dei 4 corner relativi al marker di nostro interesse (in caso di presenza multipla, si considera il marker rilevato con la miglior confidenza).

Sistema di riferimento immagine

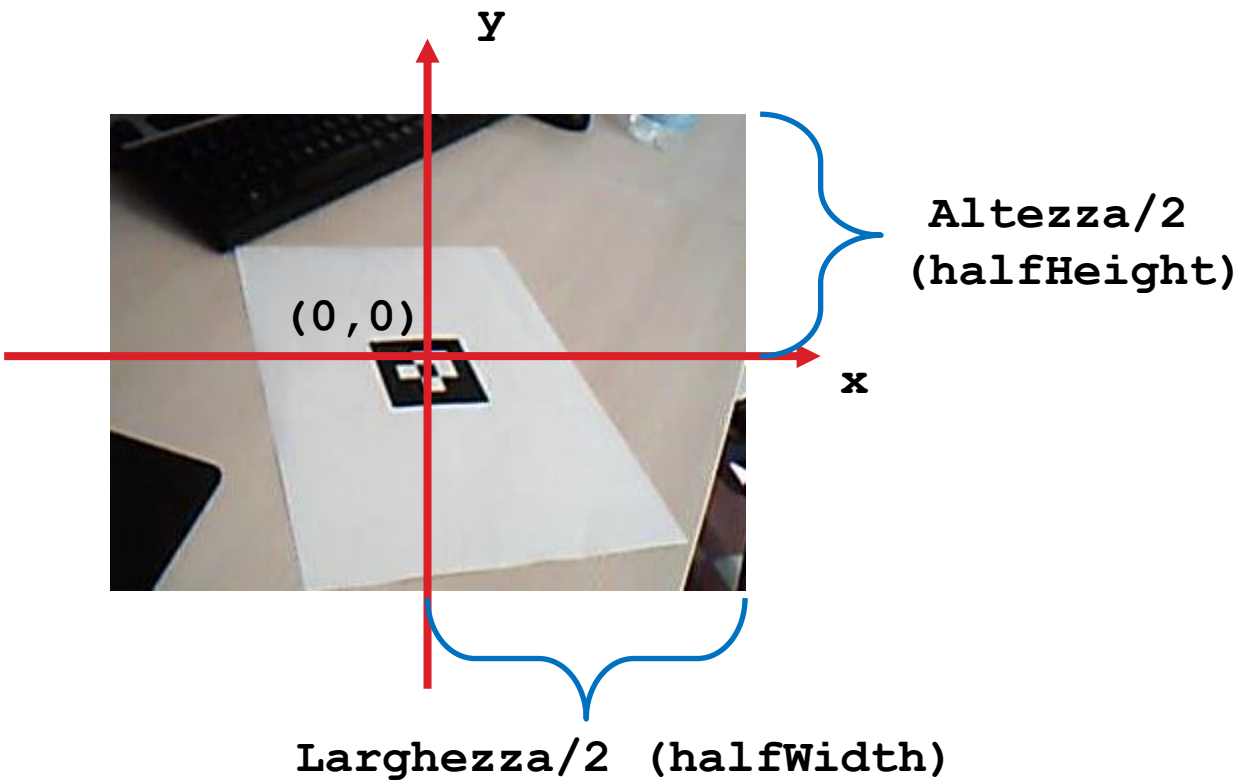
• Ogni punto (pixel) all'interno di un'immagine è identificato da un valore di ascissa e da un valore di ordinata.

• Le comuni librerie di image processing considerano il sistema di riferimento immagine con l'origine situata in alto a sinistra.



Sistema di riferimento immagine: variazione

• L'algoritmo di stima della posa del marker richiede in input (e restituisce in output) punti (pixel) con coordinate definite nel sistema cartesiano classico (con origine che coincide per convezione con il centro dell'immagine).

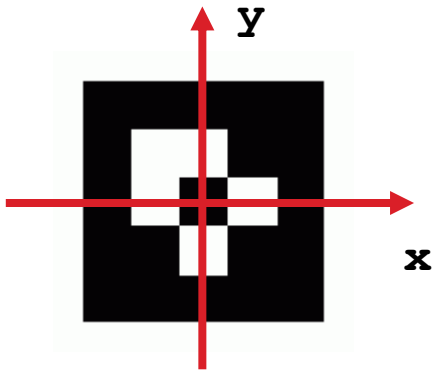


Sistema di riferimento “mondo”

• Per stimare la posa del marker è necessario conoscere quali siano le sue caratteristiche originali, cioè le coordinate dei 4 corner nel “mondo reale”.

• Per semplicità, si può supporre il sistema di riferimento “mondo” centrato sul marker e definire in questo modo le sue coordinate (considerando il marker come giacente sul piano e quindi con coordinata z pari a zero).

• Supponendo che il marker sia di dimensioni *n x n*, le coordinate dei suoi corner sono banalmente determinabili.



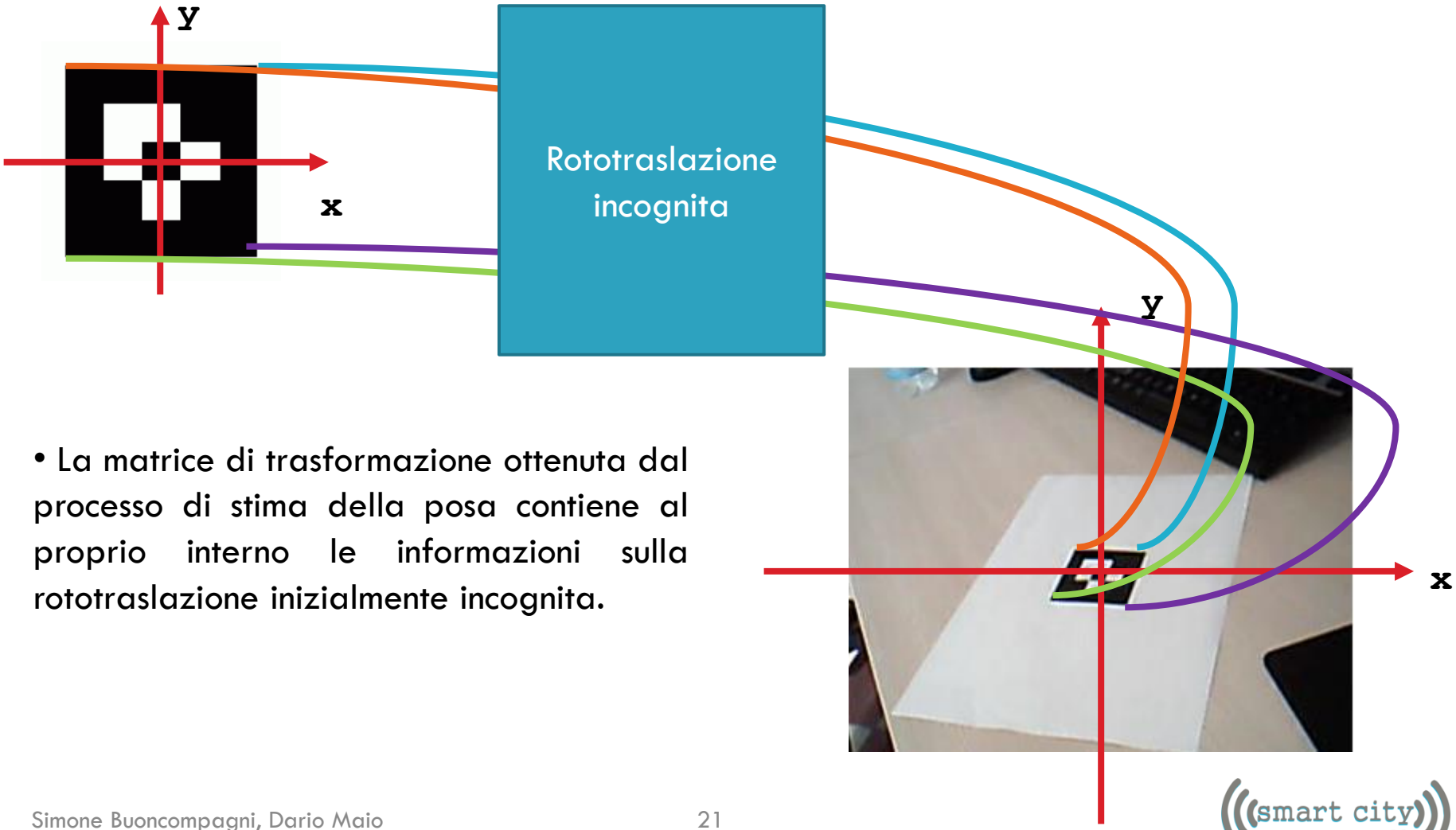
Calcolo della posa della camera

• Per proiettare informazioni in realtà aumentata coerenti con l'orientazione e la posizione del marker nello spazio è necessario procedere alla determinazione della posa del marker stesso calcolando la matrice di trasformazione.

• Le informazioni necessarie per calcolare la posa di un marker sono:
  - coordinate dei punti di corner del marker rilevato nell'immagine;
  - coordinate dei corrispondenti punti di corner del marker nel mondo reale.

• La matrice di trasformazione è ottenuta dalla combinazione del vettore di traslazione e della matrice di rotazione calcolati in maniera stimata attraverso l'algoritmo *Coplanar POSIT* di AForge.NET (Oberkampf, Daniel DeMenthon and Larry Davis, "*Iterative Pose Estimation Using Coplanar Feature Points*").

Matrice di trasformazione



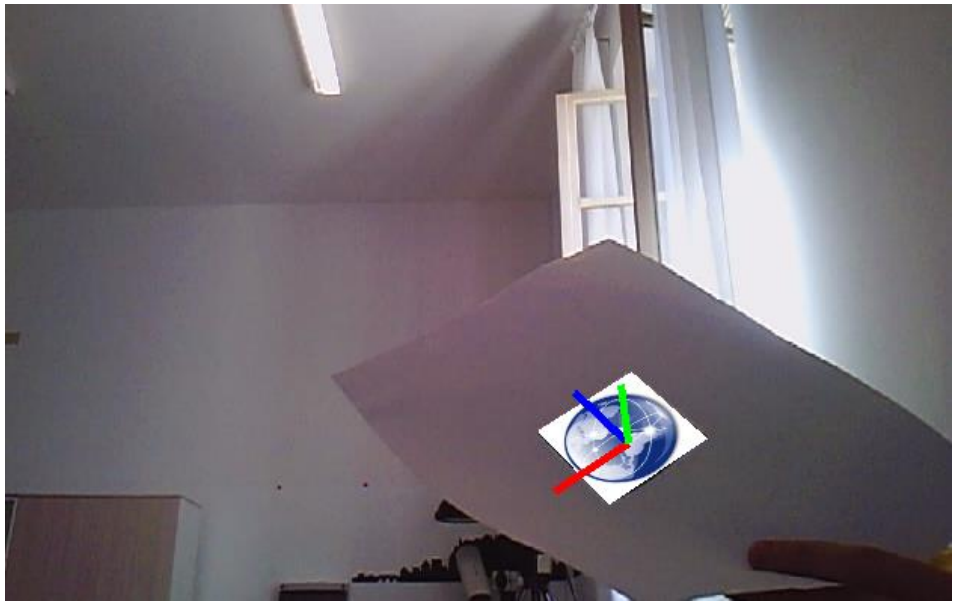
• La matrice di trasformazione ottenuta dal processo di stima della posa contiene al proprio interno le informazioni sulla rototraslazione inizialmente incognita.

Proiezione delle informazioni aumentanti

• Ottenuta la matrice di trasformazione è possibile convertire ogni punto definito nel sistema di riferimento mondo nel corrispondente punto definito in coordinate immagine.

• Nel nostro caso, le informazioni aumentanti da inserire nel frame sono di due tipi:

- immagine 2D mediante operazione di proiezione *backward;*
- assi cartesiani del sistema di riferimento solidale al marker mediante l'impiego della matrice di trasformazione.



Link utili e riferimenti

AForge.Net Source Code: http://code.google.com/p/aforge/

Simple Player: http://www.aforgenet.com/framework/samples/video.html

Progetto GRATF: http://www.aforgenet.com/projects/gratf/

Marker detection/recognition: http://www.aforgenet.com/articles/glyph_recognition/

3D Pose Estimation: http://www.aforgenet.com/articles/posit/

Oberkampf, Daniel DeMenthon and Larry Davis, "*Iterative Pose Estimation using Coplanar Feature Points*"