Università degli Studi di Trento

Tecniche di elaborazione video per la rilevazione di eventi

Matteo Liotta

8 gennaio 2014

Moduli Implementati

0.1 Accesso vietato

Il modulo ha come obbiettivo la detection di persone in aree ad accesso limitato, con preciso riferimento a due tipi di situazioni: superamento di determinati punti all'interno della scena (es: line gialla della metropolitana) e attraversamento di porte (es: door surveilance).
Per fare ciò sono necessarie alcune operazioni:

1. Estrazione dello sfondo (implementato nella classe mmCodeBookBgSubstraction): necessario per trovare gli oggetti in movimento all'interno della scena. I parametri della classe porre l'attenzione sono:
   - *cbMemory*: durata della fase di training dell'algoritmo. In generale il modello di background è migliore più è lunga e più è statica questa fase.
   - *alphaValue*: parametro che regola la gestione del modello di background. E' sempre minore di 1.
   - *betaValue*: parametro che regola la gestione del modello di background. E' sempre maggiore di 1.
2. Tracking di oggetti (implementato nella classe mmBlobDetector): effettua il tracciamento delle persone che attraversano la scena. Per gestire le occlusioni, è stato implementato un sistema di discriminazione dell'oggetto occluso, basato su modello colore (implementato nella classe appModel). Le variabili interne alla classe più importanti sono:
   - *increasedBRCornerParam* e *increasedTLCornerParam*: un'espansione del blob generato in fase di tracking risulta utile al momento della gestione delle occlusioni.
   - *minDetHeigthParam* e *minDetWidthParam*: minima altezza e grandezza che un blob deve avere per essere considerato valido.
   - *splitThreshParam*: parametro utilizzato nella gestione della fase di split di un blob. Se la distanza tra i blob splittati è minore di questo parametro allora i due blob vengono uniti e lo split non ha effetto.
   - *borderThreshParam*: delimita l'area intorno ai bordi del frame, in cui non viene effettuata la gestione dei blob.
3. Gestione dei blob all'interno delle aree ad accesso limitato (implementato nella classe blobStructure): aggiornamento della lista dei blob all'interno e all'esterno delle aree proibite, necessaria per la gestione degli allarmi.
4. Gestione dell'apertura e della chiusura della porta (implementato nella classe harrisDetector). Ci sono alcune variabili su cui porre l'attenzione:
   - *maskParam*: il corner Harris detector non lavora su tutta l'immagine, ma solamente su una porzione di essa, la cui intensità varia al momento dell'apertura e della chiusura della porta (in generale queste parti sono gli stipiti e gli angoli della porta). Per questo si costruisce un'immagine binaria, che viene usata per isolare tali punti (la matrice avrà valore 1 in corrispondenza dello stipite o dell'angolo della porta selezionato, 0 in tutti gli altri punti).
   - *threshParam*: regola la gestione dei punti di corner al variare dell'intensità luminosa.

0.2 Crowd analysis

Il modulo prevede la rilevazione di comportamenti anomali all'interno di una folla, dove per anomalo si intende tutto ciò che si discosta dal normale comportamento della folla stessa. Per fare ciò si modella il moto delle persone tramite una distribuzione gaussiana. In questo modo, basandosi sul valor medio e sulla varianza di ogni gaussiana nella scena, si può capire se essa può essere associata al moto normale della folla, o se rappresenta un comportamento anomalo.
All'interno della classe mmGaussianModel sono presenti tutte le funzioni che gestiscono il funzionamanto del modulo, che si compone di alcune fasi:

1. Inizializzazione: viene creata una griglia di particelle all'interno della scena.
2. Estrazione feature di moto: all'interno di una finestra di osservazione si trovano le particelle legate al moto della folla
3. Fase di studio: alla fine della finestra di osservazione si cerca se ciascuna particella corrisponde ad un moto anomalo o normale.
4. Gestione allarme: viene alzato l'allarme se si trovano N particelle anomale in uun area ravvicinata per più frame successivi.

Ci sono alcune variabili da tenere bene in considerazione in fase di test del modulo su altri date set:

- *numGaussianParam*: numero di gaussiane nel modello. Più il sistema è multimodale, più riesce a modellare vari comportamenti normali della folla.
- *heightParam* e *widthParam*: attorno ad ogni singola particella anomala viene costruito un rettangolo, le cui dimensioni vengono gestite da questi parametri e serve per trovare se nelle vicinanze ci sono altre particelle anormali. Questi due valori vanno variati a seconda della scena in esame.

0.3 Heat map

Lo scopo di questo modulo è andare a osservare le anomalie nel flusso della folla, andando a modellare il suo comportamento su base energetica. Si costruisce un modello standard del moto della folla in determinate aree. Tale comportamento viene preso come riferimento per considerare se l'energia calcolate nei successivi frame è sotto soglia (meno persone di quanto ci si aspetta) o sopra soglia (più persone del normale).
All'interno delle classi mmParticleAccumulation e mmParticleEnergy vengono create le funzioni che gestiscono l'intero modulo e prevedono le fasi di:

1. Inizializzazione: all'interno della scena si vanno a individuare alcune aree di interesse, nelle quali verrà calcolata la mappa energetica del moto della folla.
2. Selezione della griglia: nelle aree scelte viene creata una griglia uniforme di particelle.
3. Fase di accumulo: si definisce un periodo di osservazione, all'interno del quale si effettua l'accumulo delle particelle, basandosi sul moto presente nella scena.
4. Calcolo dell'energia: alla fine del periodo di osservazione si mantengono le particelle con moto non trascurabile. Per ognuna di esse viene poi calcolato un potenziale energetico, che è alto, più le particelle sono vicine le une alle altre.
   Il modulo si divide in due fase: una parte di training, in cui si costruisce il comportamento di riferimento della folla e una di test, dove si valuta se c'è maggiore o minore concentrazione di persone rispetto al comportamento registrato nella fase precedente.

0.4 Fight detection

L'obbiettivo è di riconoscere la lotta di due persone. Per fare ciò è stato implementato un metodo che prevede l'utilizzo di un classificatore svm, che permette di discriminare i comportamenti anomali (situazione di fight) da quelli non anomali (altre interazioni tra due soggetti diverse dal fight). Per addestrare correttamente il classificatore, sono necessari alcuni esempi di entrambe le situazioni che si vogliono andare a classificare. Una volta addestrato il classificatore, il modulo può essere testato su intere sequenze, per riconoscere i comportamenti anomali. Per un corretto funzionamento del modulo è consigliabile utilizzare per le fasi di training e di test, riprese della stessa telecamere.
Il modulo è stato diviso in 3 parti:

1. Creazione dei file di training: Seleziona le particelle di interesse (tutte quelle interne ai blob nel periodo di interazione fra i due soggetti) per la fase di training del classificatore. Per questo parte sono necessari gli algoritmi di sottrazione dello sfondo e blob tracking (classi mmCodeBookBgSubstraction e mmBlobDetector).

2. Training: addestramento del classificatore. Tutte le funzioni che gestiscono questa parte di modulo sono implementate nella classe mmBehaviourDetector. Si possono fare alcune considerazioni sui parametri da settare:

   - *svmParameters*: il tipo di classificatore svm va scelto in base alle caratteristiche del data set. In generale scegliere come tipo radial basis function come kernel e un multi-class svm come tipo di classificatore , porta buoni risultati, mentre i valori gamma e C vanno modificati a seconda del dato in esame.
   - *clusterBins*: numero di bins dell'istogramma. Più alto è tale valore più la complessità computazionale dell'algoritmo è elevata.

3. Test: test del modulo. Utilizza le funzioni di sottrazione dello sfondo, blob tracking e predizione del classificatore implementate rispettivamente in mmCodeBookBgSubstraction, mmBlobDetector e mmBehaviourDetector.
   Il parametro *windowLengthParam* regola il numero di frame su cui effettuare il testing una volta che due individui sono venuti a contatto. Tale valore non deve essere mai inferiore a 100, per permettere una corretta classificazione delle interazioni veloci.

Configurazione del sistema

Nel seguito verrà riportata la configurazione ottimale del progetto su piattaforma linux a 64-bit, utilizzando come ide eclipse.

0.5 Librerie

Per poter compilare e eseguire correttamente il progetto è necessaria l'installazione di una serie di librerie:

1. opencv: la versione richiesta è la 2.3. In linea di massima si possono usare anche le 2.4, ma facendo ben attenzione a non utilizzare funzioni introdotte nelle 2.4.
2. ffmpeg: versioni non successive alla 0.8.5.
3. boost
4. GStreamer

0.6 Configurazione

Nel progetto sono compresi:

- File main.cpp: in questo file sono contenuti tutti i moduli, decommentare quello che si vuole utilizzare.
- Cartella mmLib: contiene tutte le classi implementate.
- Cartelle Test e Training: contengono i file necessari per gestire il 4 modulo.

Una volta creato un nuovo progetto in eclipse, inserire il file main.cpp contenente i quattro moduli e le tre cartelle date. Nelle preferenze del progetto selezionare il percorso *C/C++ Build–>Settings–>GCC C++ Compiler–>Includes* e inserire la path corretta della delle seguenti cartelle:

- opencv
- opencv2
- c++/4.6.
- libavformat
- libavcodec
- libavutil
- libswscale
- boost

Ad esempio il path corretto della cartella opencv potrebbe essere /usr/local/include/opencv/. Come ultima cosa selezionare il percorso *C/C++ Build–>Settings–>GCC C++ Linker–>Libraries* e inserire le librerie da linkare, che in questo caso sono:

- opencv_core
- opencv_imgproc
- opencv_highgui
- opencv_ml
- opencv_video
- opencv_features2d
- opencv_calib3d
- opencv_contrib
- opencv_legacy
- opencv_flann
- avformat
- avcodec
- avutil
- swscale
- boost_program_options
- boost_regex
- boost_system
- boost_filesystem

Il percorso corretto delle librerie in *Library search path* potrebbe essere /usr/local/lib/ ad esempio.
n.b. a seconda della versione di eclipse utilizzata, i percorsi potrebbero essere diversi. La cosa importante è andare a inserire correttamente i percorsi delle varie librerie.