Bosch Security Systems

Relazione tecnica
Linguaggio script delle attività IVA

BOSCH
Invented for life

20 Novembre 2015

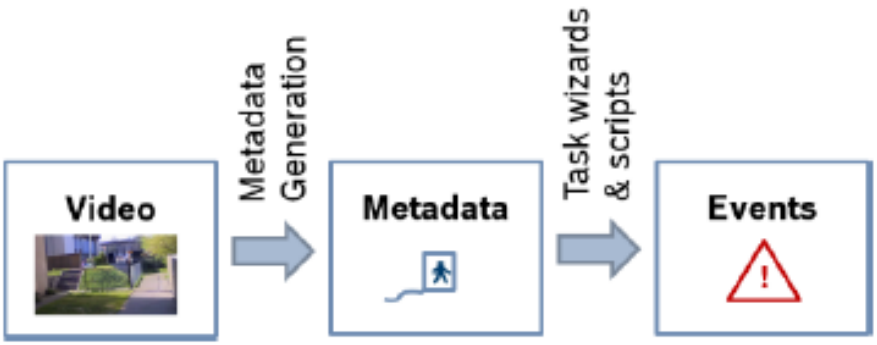
Linguaggio script delle attività IVA (Intelligent Video Analysis)

La seguente relazione tecnica introduce brevemente il linguaggio script delle attività IVA e fornisce esempi di script spiegati nel dettaglio.

Che cos'è il linguaggio script delle attività IVA?

Il linguaggio script delle attività IVA

- Descrive tutte le attività IVA predefinite e configurate ed il wizard delle attività (task wizard)
- Può combinare attività per crearne di più complesse
- NON può configurare la produzione di metadati



Quando utilizzare gli script delle attività IVA?

Gli script delle attività IVA sono utilizzati implicitamente ogniqualvolta un'attività IVA è configurata tramite l'interfaccia grafica utente. Tuttavia è possibile configurare manualmente gli script IVA, pratica suggerita quando le attività predefinite non siano sufficienti:

- Gli script possono essere copiati dall'editor ed incollati in esso per il backup e lo scambio delle attività configurate
- Quando sono necessari più di otto attività di allarme: sono configurabili 16 attività di allarme esterne tramite script di attività IVA
- Ottimizzazione delle linee e dei campi
- Combinazione di linee per il conteggio

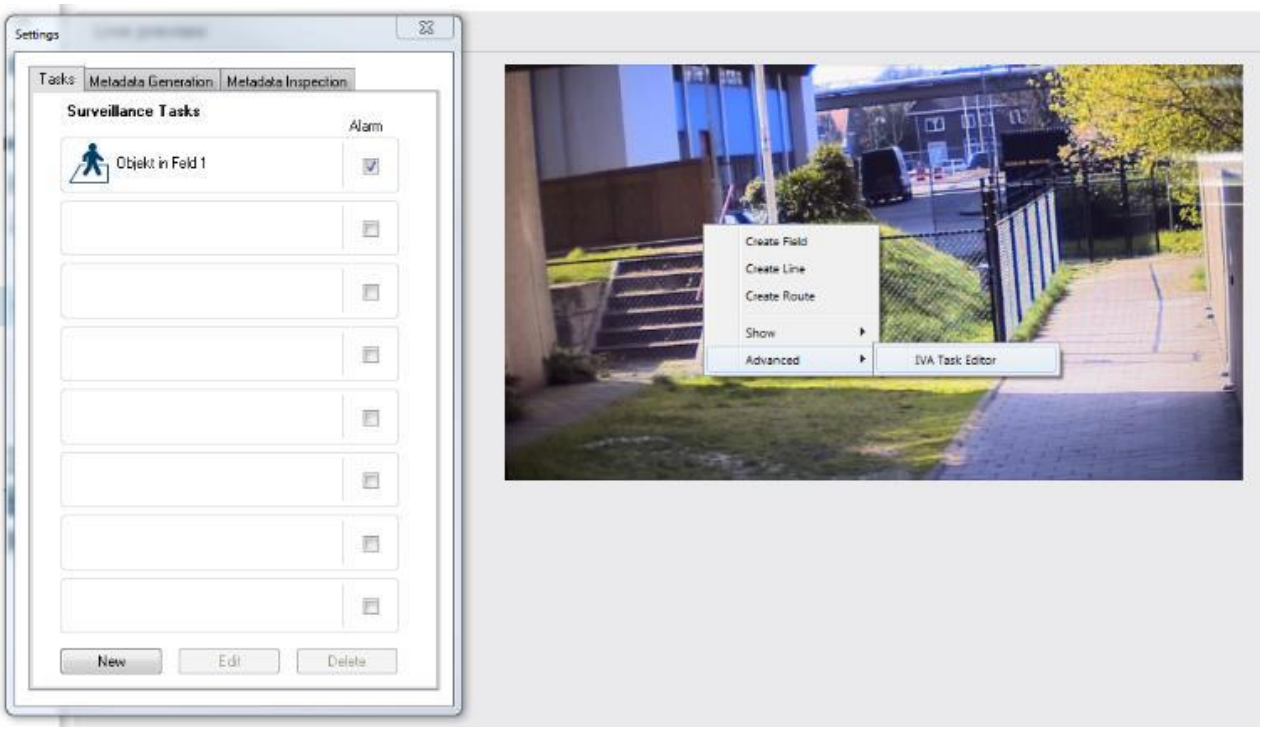
- “Non toccare” (modalità museo) con versione Firmware inferiore alla 6.10: l’allarme deve scattare da ogni parte del bounding box che delimita l’oggetto, non solo dal centro dello stesso
- Sono necessarie combinazione logiche di eventi predefiniti

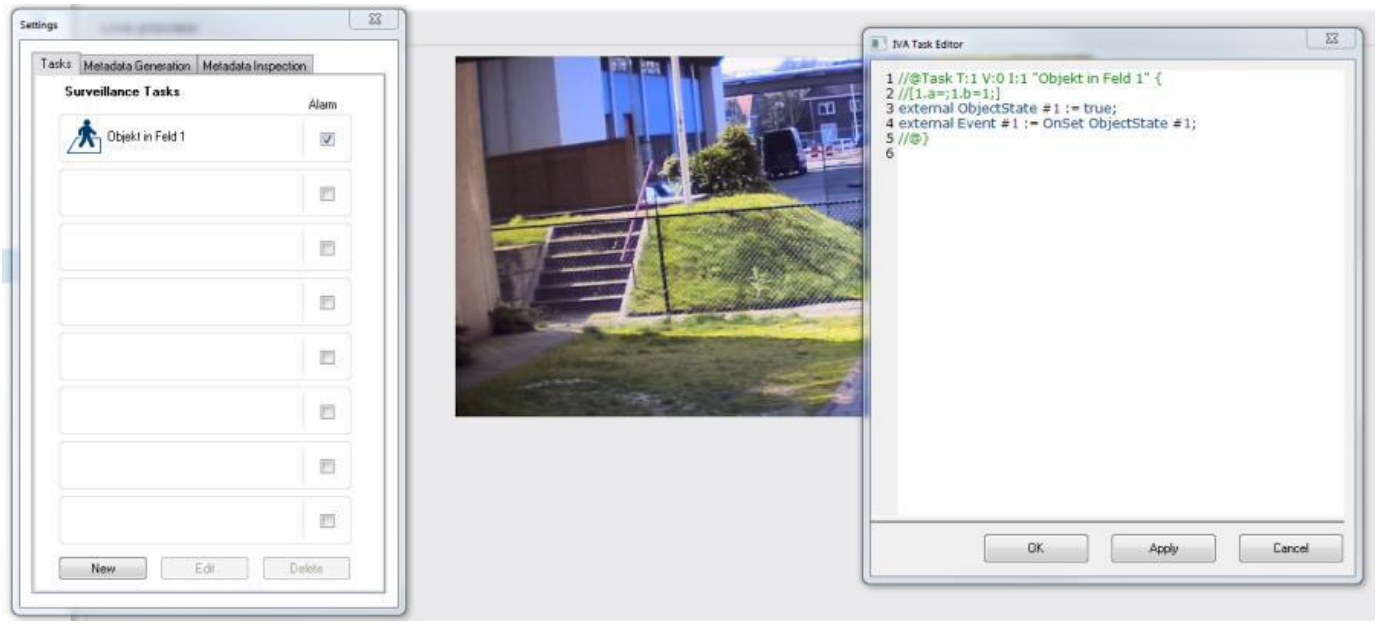
Come utilizzare gli script delle attività IVA?

1) Si definiscano tutte le attività tramite il task wizard per quanto possibile
2) Si passi all’editor degli script di attività IVA. Le attività predefinite si possono trovare lì con tutti i dettagli
3) Si effettuino le proprie modifiche
4) Si modifichino le attività predefinite in attività definite via script, così che un utilizzo accidentale dei task wizard non sovrascriva i propri cambiamenti

Come accedere ali linguaggio script delle attività IVA?

E’ necessario andare nella configurazione IVA ed aprire la pagine delle attività. Si clicchi con il tasto destro del mouse sul video e si selezioni Advanced -> IVA Task Editor. Apparirà un’altra finestra con lo script delle attività IVA attuali:





Panoramica delle attività e dei filtri applicabili agli oggetti

Un'attività è formata da:

- Una primitiva dell'attività
- Uno stato dell'oggetto o un'interazione con una primitiva dell'attività
- Un filtro da applicare alle proprietà dell'oggetto

Si noti bene che non tutte le attività o i filtri indicati sono disponibili in ciascuna versione del firmware

Attività IVA
Rilevazione di oggetti
Attraversamento linee
Oggetto in campo
Ingresso in campo
Uscita dal campo
Seguire percorsi
Indugiare in modo sospetto
Oggetto rimosso
Oggetto inattivo
Modifica delle condizioni
Ricerca di somiglianza
Conteggio
Conteggio delle persone con BEV
Rilevamento in presenza di folle
Manomissione

Attività di flusso IVA
Flusso nella scena
Flusso opposto nella scena
Rilevamento in presenza di folle
Manomissione

Primitive delle attività IVA
Linea
Campo
Percorso
Immagine

Filtri
Area dell'oggetto
Proporzioni
Velocità
Direzione
Colore
Sagoma
Classe dell'oggetto

Panoramica degli stati degli oggetti e degli eventi

Gli eventi sono relazioni temporali:

- Possono essere mostrati sull'interfaccia grafica utente fino ad 8 eventi esterni
- Possono essere definiti in totale fino a 16 eventi esterni
- Possono essere definiti in totale fino a 32 eventi

Gli stati sono relazioni spaziali, proprietà degli oggetti e stati di manomissione:

- Possono essere definiti fino a 16 stati esterni
- Possono essere definiti in totale fino a 32 stati

Gli eventi e gli stati possono essere interni od esterni. Solo gli eventi esterni e gli stati sono mostrati in output dall'IVA.

SimpleStates dovrebbero essere utilizzati ogniqualvolta una proprietà non ha oggetti corrispondenti. Possibili esempi sono Motion+, attività di flusso IVA, valori di conteggio e stati di manomissione.

Eventi degli oggetti		
CrossedLine	→	Linea attraversata
EnteredField	→	Ingresso in campo
LeftField	→	Uscita dal campo
FollowedRoute	→	Seguito percorso
Appeared	→	Comparso
Disappeared	→	Scomparso
Idled	→	Diventato inattivo
Removed	→	Rimosso

Stati degli oggetti	
InsideField	→ Se dentro il campo
ObjectsInField	→ Numero di oggetti in campo
IsLoitering	→ Se si indugia in modo sospetto
SimilarToColor	→ Se l'istogramma dei colori è simile a quello di soglia
DetectedFlow	→ Se presente flusso significativo
EstimatedCrowdDensity	→ Se densità di folla supera la soglia o meno
HasDirection	→(true) Se la direzione dell'oggetto specificato è disponibile nel frame attuale
HasVelocity	→(true) Se la velocità dell'oggetto specificato è disponibile nel frame attuale
HasAspectRatio	→(true) Se il box virtuale dell'oggetto specificato è disponibile nel frame attuale
HasObjectSize	→(true) Se il box virtuale dell'oggetto specificato è disponibile nel frame attuale
HasColor	→ (true) Se l'istogramma dei colori dell'oggetto specificato è disponibile
HasFace	→(true) Se una sagoma è rilevata nell'oggetto specificato nel frame attuale
HadFace	→ (true) Se una sagoma è stata rilevata nell'oggetto specificato dalla sua ultima apparizione

HasClass	→ (true) Se è rilevato un oggetto di una delle classi specificate nel frame attuale
HadClass	→ (true) Se è stato rilevato un oggetto di una delle classi specificate dalla sua ultima apparizione

Proprietà degli oggetti	
Direction	→ In output la direzione attuale dell'oggetto specificato, se disponibile
Velocity	→ In output la velocità attuale in [m/s] dell'oggetto specificato, se disponibile
AspectRatio	→ In output le proporzioni attuali dell'oggetto specificato, se disponibile
ObjectSize	→ In output le dimensioni attuali in [m²] dell'oggetto specificato, se disponibile
FaceWidth	→ In output la larghezza attuale della sagoma rilevata assegnata all'oggetto specificato, se disponibile
MaxFaceWidth	→ In output la larghezza massima tra tutte le sagome rilevate assegnate all'oggetto specificato, se disponibile

Stati di manomissione	
SignalTooNoisy	→ Se il segnale video presenta troppo rumore
SignalTooDark	→ Se il segnale video diventa troppo scuro
SignalTooBright	→ Se il segnale video diventa troppo luminoso
SignalLoss	→ Se il segnale video è perso
GlobalChange	→ Se la maggior parte del contenuto dell'immagine è cambiato
RefImageCheckFailed	→ Se sono rilevate differenze significative tra il segnale video attuale e l'immagine di riferimento, nel caso sia stata stabilita

Combinare stati degli oggetti ed eventi

Sono possibili le seguenti combinazioni tra stati degli oggetti ed eventi:

Combinazione logica di stati/condizioni:

- and
- or

Condizioni sugli eventi:

- where

Combinazione di eventi tramite relazioni temporali:

- before
- before (<from>, <to>)
- not before

Cambi di stato come eventi:

- OnChange
- OnSet
- OnClear

Comprendere il linguaggio script delle attività IVA: Attraversamento Linee, GUI

```
// Definizione delle primitive dell'attività

Line #1 := { Point(103,  30) Point(159,  77)};
Line #2 := { Point(26,  65) Point(82,  122)
             DebounceTime(0.50)  Direction(1) };

// Definizione di un'attività di allarme mostrata nella GUI
//@Task T:2 V:0 I:1 "Crossing line 1" {
//[1.a=s1:1;1.b=1;1.c=32;1.d=31;4.a=i:1;]
external Event#1  :={CrossedLine#1};
//@}

//Definizione di un'attività definita ad hoc mostrata nella GUI
//@Task T:0 V:0 I:2 "Crossing line 2" {
external Event#2  :={CrossedLine#2};
//@}

//Definizione di una attività di allarme non mostrata nella GUI
external Event#3  :={CrossedLine#2};
```

Line: 2 punti di delimitazione

Direction di una Line(linea):
//qualsiasi
Direction(1) //Avanti
Direction(2) //Indietro

DebounceTime di una linea o di un campo è opzionale

CrossedLine #x è un evento che scatta quando un oggetto attraversa la Line #x nella maniera specificata

external è la parola chiave per gli allarmi e le statistiche

Definizione della task wizard:
//@Task T:x V:y I:z
T:x indica il # dell'attività (Oggetto in campo, attraversamento linee… guardare l'icona per quella corretta!)
T:0 indica una task creata ad hoc. Utilizzarlo per i propri script per evitare che i task wizard la sovrascrivano
V:0 è il numero della versione, attualmente sempre 0; I:z è lo slot nella pagina delle attività
I:1 indica lo slot occupato nella lista delle attività sull'interfaccia utente. Affinché esso diventi rosso in caso di allarmi, l'evento/stato esterno definito nell'attività deve avere lo stesso numero dell'attività stessa
[1.a=s1:1;…] indica i valori del task wizard

Comprendere il linguaggio script delle attività IVA: Attraversamento linea con filtri applicati agli oggetti

```
//Definizione delle primitive dell'attività
Line #1  :=  { Point(103,  30) Point(159,  77)};

//@Task T:2 V:0 I:1 "Crossing line 1"  {
//[1.a=s1:1;1.b=1;1.c=32;1.d=31;3.o=(b1:1,
b2:0.1,b3:500,  c1:1,c2:0.02,c3:34.00,d1:1,
d2:0.d3:27.78,e1:1,e2:315,  e3:45,f1:1,f2:135,
f3:225);4.a=(c:2b19,i:1,pr:2);5.a=(c:1,p:1);
6.a=(d1:8,d2:33,r:2,u:1);]
ColorHistogram  #1:= { HSV(60,38,100,25)
        Similarity(75)  Outlier(75) };
external Event#1:={CrossedLine#1
        where ObjectSize within(0.1,500)
        and AspectRatio within(0.02,34.00)
        and Velocity within(0,27.78)
        and (Direction within(315,405)  or  Direction within(135,225))
        and SimilarToColor #1
        and HadFace and MaxFaceWidth within(8,33)};
//@}
```

Le proprietà degli oggetti possono essere modellate tramite gli stati degli oggetti o tramite limiti agli eventi

where pone dei limiti agli eventi per quanto concerne le proprietà degli oggetti

and / or servono a combinare le proprietà

within limita entro un intervallo di valori

Il carattere * indica valori non definiti

Esempio di creazione di codice tramite gli stati degli oggetti:

```
ObjectState#1  := Velocity  within (30.0,*);
```

Comprendere il linguaggio script delle attività IVA: Campi

```
//Definizione delle primitive dell'attività
Field  #1 := { Point(56,  24) Point(106,  24)
               Point(106,  74)};
Field  #2 := { Point(56,  24) Point(106,  24)
               Point(106,  74) Point(56, 74)
               DebounceTime(0.50)
               ObjectSet(BoundingBox)
               SetRelation(Covering)};

//@Task T:1 V:0 I:1 "Object in field 1"  {
//[1.a=1;1.b=1;a=1:1;]
external ObjectState  #1 := InsideField  #1;
//@}
```

Field: da 3 a 16 vertici

DebounceTime del campo è opzionale

ObjectSet del campo:
ObjectSet(BaryCenter) #default
ObjectSet(BoundingBox)

SetRelation per il BoundingBox:
SetRelation(Intersection) #default
SetRelation(Covering)

States del campo:
InsideField#x
ObjectsinFIeld#x

Events del campo:
EnteredField#x
LeftField#x

Comprendere il linguaggio script delle attività IVA: Percorsi

```
//Definizione delle primitive dell'attività
Route  #1 :=  { Point(34,  111) Distance(5)
                Point(92,  125) Distance(5)
                Point(140,  104) Distance(5)
                Point(143,  72) Distance(5)
                MinPercentage(80) MaxGap(10) };

//@Task T:6 V:0 I:1 "Follow route 1" {
//[1.a=id:1;1.b=1;3.a=i:1;]
external Event  #1 := {FollowedRoute  #1};
//@}
```

Route: da 2 a 8 vertici più la distanza

Direction della linea/del percorso:
#qualsiasi
Direction(1) #Avanti
Direction(2) #Indietro

FollowedRoute #x è un evento che scatta quando un oggetto segue una Route #x nella maniera specificata

Esempio: Combinare linee per il conteggio

```
//Definizione delle primitive dell'attività
Line  #1 := { Point(0, 90) Point(60, 90)
              DebounceTime(0.10)  Direction(1) };
Line #2 := { Point(60, 90) Point(90,60)
             DebounceTime(0.10) Direction(1) };
Line #3 := { Point(90, 60) Point(90, 0)
             DebounceTime(0.10) Direction(1) };

//@Task T:0 V:0 I:1 "Counter 1"  {
Event#1 :=CrossedLine#1 or CrossedLine#2
            or CrossedLine#3;
external Counter#1  := { Event#1  Text("Corner Count1:")
                         TopLeft(4,4) Mode(Wraparound)
                           within(0,999999999)  };
//@}
```

Bisogna assicurarsi che il punto che chiude una linea costituisca l'inizio di quella adiacente, in modo da non lasciare spazi

Assicurarsi di fissare il medesimo DebounceTime per tutte le linee

Combinare gli eventi di attraversamento linee tramite or

Counter indica l'attività di conteggio. L'evento per cui inizia il conteggio è configurato, così come la posizione e la stringa di testo per il conteggio. La modalità(Mode) selezionata nell' esempio indica che, quando si raggiunge il numero massimo, il contatore riparte da 0

Esempio: attraversamento linea1 a 60 km/h e linea2 a 20 km/h

```
//Definizione delle primitive dell'attività
Line #1  := {Point(50,0)  Point(50,144)};
Line #2  := {Point(120,0) Point(120,144)};

//Definizione degli eventi di attraversamento linea
Event#11  :={CrossedLine#1  where
             Velocity  within(13.89, 19.44) };
Event#12  :={CrossedLine#2  where
            Velocity  within(2.778,5.556)  };

//Combinazione di eventi di attraversamento linea
//@Task T:0 V:0 I:1 "Two Line Speed Check" {
external Event#1  :={Event#11 before Event#12
                     where first.oid==second.oid};
//@}
```

In realtà l'allarme scatta quando la velocità è compresa per la linea1 tra 50-70 km/h e per la seconda tra 10-30 km/h

L'intervallo di velocità non è in [km/h] ma in [m/s], pertanto le condizioni sono state generate con le attività di attraversamento linee di default dell'interfaccia utente

Procedura:

1) Definiti due attraversamenti linea con filtri di velocità tramite l'interfaccia utente
2) Rimossi tutti i comandi e definizioni di attività external, mantenendo le primitive dell'attività e le definizioni degli eventi
3) Aggiunta la combinazione di entrambi gli eventi di attraversamento linea come attività ad hoc

Lo stesso oggetto deve far scattare entrambi gli attraversamenti linea. Di conseguenza l'utilizzo della scrittura where first.oid==second.oid

Esempio: fermarsi in un'area dopo aver attraversato una linea

```
//Definizione delle primitive dell'attività
Line #1 := { Point(130, 0) Point(130, 144) };
Field #2 := { Point(50, 0) Point(115, 0)
              Point(115, 144) Point(50, 144)};

//@Task T:0 V:0 I:1 "Loitering after Line Crossing" {
Event#11 :={CrossedLine  #1};
Loitering #13 := {Radius (15) Time (10) };
ObjectState #14 := InsideField #2 and IsLoitering #13;
external Event #1 := {Event #11 before OnSet ObjectState #14
                      where first.oid==second.oid};
//@}
```

Si sta utilizzando "loitering" invece che "idle" a fini dimostrativi

Si noti che solo Event #1 è external e quindi tale da generare allarmi!

Lo stesso oggetto deve far scattare sia l'attraversamento linee che i movimenti che il loitering. Di conseguenza l'utilizzo di where first.oid==second.oid

Esempio: allarme quando un oggetto tocca un'area

```
//Definizione di primitive dell'attività
Field  #1 := { Point(100, 0) Point(175, 0)
               Point(175, 144) Point(100, 144)
               ObjectSet(BoundingBox)};

//@Task T:0 V:0 I:1 "Object Touches Field" {
ObjectState  #1 := InsideField   #1;
external Event #1 := OnSet ObjectState #1;
//@}

//In alternativa, gli oggetti devono arrivare da fuori il campo
//@Task T:0 V:0 I:2 "Object Touches Field"  {
external Event #2 := EnteredField #1;
//@}
```

L'allarme scatta al tocco dell'oggetto invece che al suo centro di gravità tramite ObjectSet(BoundingBox)

Gli eventi e gli stati hanno una numerazione separata, di conseguenza sia ObjectState sia Event possono avere #1

L'utilizzo dello stato InsideField permette all'allarme di scattare quando gli oggetti appaiono in campo

L'utilizzo dell'evento EnteredField fa scattare l'allarme solo con oggetti che sono stati fuori dal campo in precedenza

Esempio: allarme quando un oggetto entra in un'area almeno 4 volte

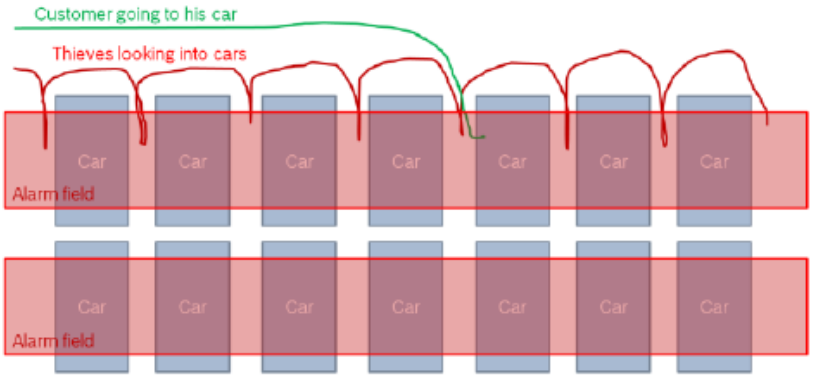
```
//Definizione delle primitive dell'attività
Resolution  := { Min(-1, -1) Max(1, 1) };
Field  #1 := { Point(- 0.244, -0.922) Point(0.325, -0.944)
               Point(0.350, 0.944) Point(-0.231, 0.944)
               DebounceTime(0.50) };

//@Task T:0 V:0 I:1 "Thieve detection" {
Event  #11  :={EnteredField  #1 before EnteredField  #1
                                  where first.oid==second.oid};
Event  #12  :={Event #11 before EnteredField  #1
                                 where first.oid==second.oid};
external Event#1  :={Event #12 before EnteredField  #1
                                  where first.oid==second.oid};
//@}
```

Applicazione: proteggere macchine parcheggiate

Utilizzo di before per concatenare eventi

Utilizzo di un modo alternativo per definire le coordinate utilizzando un range di risoluzione



Esempio: allarme quando almeno due oggetti sono all'interno di un'area

```
//Definizione di primitive dell'attività
Resolution := { Min(-1, -1) Max(1,1) };
Field  #1 := { Point(-0.656, -0.700) Point(0.350, -0.767)
               Point(0.469, 0.644) Point(-0.694, 0.644)
               DebounceTime(0.10)  };

//@Task T:0 V:0 I:1 "Alarm on more than two objects " {
external ObjectState  #1:= ObjectsInField#1  within (2,*);
//@}
```

ObjectsInField#1 fornisce in output il numero di oggetti nel campo 1. In questo esempio l'intervallo per l'allarme è configurato da 2 ad infinito.

Esempio: contare il numero di oggetti in un'area

```
//Definizione delle primitive dell'attività
Resolution := { Min(-1, -1) Max(1,1) };
Field   #1 := { Point(-0.656, -0.700) Point(0.350, -0.767)
                Point(0.469, 0.644) Point(-0.694, 0.644)
                DebounceTime(0.10) };

//@Task T:0 V:0 I:1 "Count of objects in field " {
external Counter#1 := { ObjectsInField#1 Text("Counter:")
                        TopLeft(-0.975,-0.844) };
//@}
```

ObjectsInField può anche essere utilizzato come input per un contatore

Esempio: Allarme se una fila è vuota e l'altra è composta da almeno 3 persone

```
//Definizione delle primitive dell'attività
Resolution:= { Min(-1, -1) Max(1, 1) };
Field #1:= { Point(-0.150, -0.700) Point(0.250, -0.700)
             Point(0.250, 0.600) Point(-0.150, 0.600)
             DebounceTime(0.10)  };
Field #2:= { Point(-0.600, -0.700) Point(-0.200, -0.700)
             Point(-0.200, 0.600) Point(-0.600, 0.600)
             DebounceTime(0.10) };

//@Task T:0 V:0 I:1 "Only one queue" {
ObjectState  #1:= ObjectsInField#1  within (3,*);
external Event  #1:= {OnSet ObjectState  #1
                      where ObjectsInField#2 within (0,0)};
//@}

//Per vedere i risultati del controllo sull'assenza di oggetti, utilizzare la seguente:
//@Task T:0 V:0 I:2 "No object"}
external SimpleState  #2:= ObjectsInField#2  within (0,0);
//@}
```

ObjectsInField#2 within (0,0) è un simple state perché non è coinvolto nessun oggetto

Bosch Sicherheitssyteme GmbH
Robert-Bosch-Ring 5
856303 Grasbrunn
Germany
www.boschsecurity.com