POLITECNICO
MILANO 1863

Dipartimento elettronica, informazione e bioingegneria

Ingegneria dell'Automazione

Smistamento pacchi: un approccio basato su computer vision e machine learning

Andrea Tanzini, 863324

Relatore: Prof. Manuel Roveri

Correlatori: Alessandro Basso, Ph.D.

Prof. Giacomo Boracchi

Anno accademico 2017-2018

Abstract

Il presente lavoro trova fondamento nella necessità di rendere il sistema per il riconoscimento dei colli logistici e per la *double detection* il più possibile versatile, allontanando dunque il suo funzionamento dal *modus operandi* dei *sistemi esperti,* configurati manualmente.

Per raggiungere questo obiettivo e per poter affrontare il problema con strategie basate sul *machine learning*, è stato necessario svolgere un attento lavoro per la creazione di un *dataset* sufficientemente rappresentativo, attraverso la creazione di un applicativo *software* che permettesse di ampliare e modificare i dati secondo le necessità future. Inoltre, analizzando il *dataset* prodotto, sono state selezionate ed estratte le *features* per permettere ai classificatori considerati di avere più informazioni possibili su cui basare il loro lavoro di classificazione.

Inizialmente è stata testata una strategia, definita come *baseline*, attraverso cui la classificazione viene portata a termine da classificatori *SVM, K-nn, Random Forest,* posti in parallelo che analizzano in toto le *features* disponibili con le *posterior probabilities* aggregate grazie all'uso di un classificatore *Naive Bayes.*

Successivamente, dopo aver appurato lo scarso valore aggiunto portato da una analisi a frame successivi per via della varianza quasi nulla delle *features* tra i diversi frames, è stata strutturata una nuova strategia di classificazione che ottimizza i parametri dei classificatori e che permette di arrivare alla classificazione del caso *double* in modo più robusto e in minore tempo, selezionando per ogni fase i sottoinsiemi di *features* con maggiore potere discriminante per la specifica classe.

Infine, analizzando i problemi sorti in fase di analisi e dopo aver evidenziato le criticità che questi comportano, si propongono delle alternative e degli sviluppi futuri per migliorare ulteriormente la qualità dei risultati.

Sommario

1. Il problema industriale 9

2. Formalizzazione del problema 13

2.1. Le classi 13

3. Il software per l'analisi e l'etichettamento 17

3.1. Creazione della reference 18

3.2. Correzione delle immagini 20

3.3. Sottrazione dello sfondo 21

3.4. Estrazione dei contorni 22

3.5. Etichettamento 26

4. Analisi del dataset 29

4.1. Rappresentazione classi 29

5. La selezione ed estrazione delle features 31

5.1. Contour: area e perimetro 31

5.2. Bounding box: altezza, larghezza e area 32

5.3. Aspect ratio 32

5.4. Extent ratio 32

5.5. Hull Area 33

5.6. Contour solidity 33

5.7. Ellisse equivalente: asse maggiore ed asse minore 33

5.8. Convexity defects 34

5.9. Minimum rectangle area 34

5.10. Extent minima second 35

5.11. Solidity minima 35

5.12. HU moments 35

5.13. Istogrammi 36

5.13.1. Istogramma 0-400mm: 20 steps 36

5.13.2. Istogramma 400mm-1000mm: 10 steps 37

5.14. Invarianti alla rotazione 38

6. Machine learning: strategie di classificazione e analisi risultati 40

6.1. Analisi classificatori e complessità computazionale 41

6.1.1. SVM 41

6.1.2. Random Forest 42

6.1.3. K-Nearest Neighbor 42

6.1.4. Naive Bayes 43

6.2. Strategia 1: strutturazione baseline 44

6.2.1. Random Tree 45

6.2.2. Random Forest 46

6.2.3. Support Vector Machine 48

6.2.4. K-Nearest Neighbor 49

6.2.5. Aggregazione posterior: Naive Bayes 50

6.2.6. Considerazioni aggiuntive 52

6.3. Test Multi frame: media VS aggregazione 53

6.3.1. Nuova struttura del dataset 53

6.3.2. Strategia di analisi 56

6.3.3. Conclusione 60

6.4. Strategia 2: classificazioni successive con classificatori specializzati 63

6.4.1. Prima fase: double detection 66

6.4.2. Seconda fase: rigid vs. envelope 67

6.4.3. Terza fase: evelope vs. env. Parcel e flyer vs. parcel 69

6.4.4. Confronto 70

7. Limitazioni in produzione 72

8. Conclusioni e sviluppi futuri 74

9. Bibliografia 76

Indice delle figure

Figura 1-Immagine RGB 10

Figura 2-Immagine IR 10

Figura 3-Immagine DEPTH 10

Figura 4-Esempio parcel 14

Figura 5-Esempio envelope 14

Figura 6-Esempio enveloped parcel 15

Figura 7-Esempio flyer 15

Figura 8-Esempio double 16

Figura 9-Esempio other 17

Figura 10- Sequenza operazioni software etichettamento 18

Figura 11- Sequenza operazioni software on-line 18

Figura 12-Rgb original 19

Figura 13-Rgb cropped 19

Figura 14-Rgb without zeros and spikes 19

Figura 15-Correzione RGB 21

Figura 16-Correzione IR 21

Figura 17-Correzione DEPTH 21

Figura 18- Output rimozione background 22

Figura 19-Maschere risultato RGB-IR-DEPTH 23

Figura 20-Maschera complessiva risultato 24

Figura 21-Contorni 24

Figura 22-Esempio contorni 25

Figura 23-Esempio contorni 25

Figura 24-Esempio contorni 25

Figura 25-Menù selezione classi....26
Figura 26-Ispezione 3D....27
Figura 27- Controllo etichettamento....27
Figura 28-Contorni....31
Figura 29-Bounding box....32
Figura 30-Hull....33
Figura 31-Ellisse equivalente....33
Figura 32-Convexity defects....34
Figura 33-Minimum area rectangle....34
Figura 34-Istogrammi 0-400 mm....37
Figura 35-Istogrammi 400-1000 mm....38
Figura 36-Esempio SVM....41
Figura 37-Matrice di confusione Random Tree....45
Figura 38-Matrice di confusione Random Forest....47
Figura 39-Matrice di confusione SVM....48
Figura 40-Matrice di confusione K-NN....49
Figura 41-Matrice di confusione Baseline....52
Figura 42-Posizione frames....55
Figura 43-Esempi sequenze di frames....56
Figura 44-Matrice di confusione test multiframe, singola classificazione....57
Figura 45-Schema test media....58
Figura 46-Matrice di confusione test multiframe, media frames....58
Figura 47-Schema test aggregazione....59
Figura 48-Matrice di confusione test multiframe, aggregazione....60
Figura 49-Varianze medie in triple di frames....61

Figura 50-Nuova divisione in macroclassi 64

Figura 51-Fasi classificazione sequenziale 65

Figura 52-Schema sequenziale, prima fase 66

Figura 53-Matrice di confusione classificazione sequenziale, prima fase 67

Figura 54-Schema sequenziale, seconda fase 67

Figura 55-Matrice di confusione classificazione sequenziale, seconda fase 68

Figura 56-Schema sequenziale, terza fase 69

Figura 57-Matrice di confusione classificazione sequenziale, terza fase 70

Figura 58-Schema sequenziale complessivo 71

Figura 59-Problema usura del nastro 73

Figura 60-Risultato usura del nastro in analisi 73

1. Il problema industriale

Il problema industriale in cui va a inserirsi il lavoro di seguito trattato è fortemente caratteristico del settore logistico: lo smistamento pacchi. Nei centri di smistamento vengono fatti confluire tutti i vari colli su un nastro trasportatore e, affinché lo smistamento avvenga correttamente, è necessario che i colli siano sufficientemente distanziati tra di loro. In questo modo il riconoscimento dei codici e le successive manipolazioni possono avvenire agevolmente senza danneggiare la merce trasportata e i sistemi di manipolazione. Risulta frequente che alcuni colli superino la fase di smistamento preliminare ancora a contatto gli uni con gli altri, parzialmente sovrapposti, oppure completamente sovrapposti.

Per superare questo problema, tra la stazione di singolarizzazione preliminare (realizzata mediante l'uso di rulli in azione a velocità diverse) e la stazione vera e propria di smistamento, si configura ottimale l'inserimento negli impianti di una stazione che permetta l'analisi dei colli per poter riconoscere la loro bontà (ovvero che ciascun collo sia debitamente distanziato dagli altri) e la sua natura (ovvero che si tratti di una busta, di una pacco, di un pacco imbustato, di un flyer, etc…) . Una volta individuato il caso critico *double* sarà necessario il suo allontanamento dal flusso principale, per far sì che un singolarizzatore più efficace provveda alla separazione e all'allontanamento tra loro dei colli in questione.

Questa fase di riconoscimento è di vitale importanza per l'efficacia degli impianti e risulta importante che l'analisi venga effettuata in modo robusto rispetto all'accuratezza di classificazione ed efficiente in ottica di tempo di computazione richiesto per le analisi. L'analisi viene effettuata attraverso un sistema di visione posto ortogonale alla direzione di scorrimento del nastro e permette l'acquisizione di tris di immagini relative allo stesso istante. Si possono dunque analizzare, dello stesso istante temporale, una immagine *RGB*, una *IR* e una relativa alle altezze dei colli che transitano sotto il sistema di visione, detta *DEPTH*. In Figura 1-

Immagine RGB, Figura 2-Immagine IR e Figura 3-Immagine DEPTH, ne è riportato un esempio.

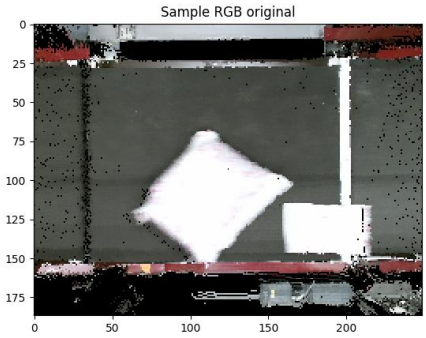


Figura 1-Immagine RGB

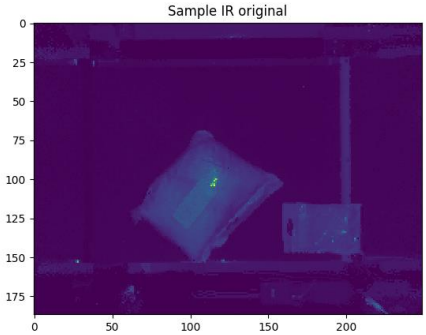


Figura 2-Immagine IR

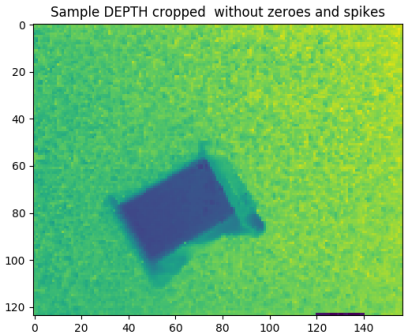


Figura 3-Immagine DEPTH

Lo stato dell'arte di questa sezione di impianto consiste nello sfruttare il sistema di visione per svolgere analisi di pura *computer-vision*. Vengono, attraverso una libreria proprietaria, sfruttate le immagini *RGB*, *IR* e *DEPTH* per effettuare misurazioni dello stato

sottostante il mezzo di rilevazione e, mediante un *sistema esperto* [1] tarato in base all'esperienza dell'operatore in fase di progettazione, prendere una decisione. Tale sistema tiene conto delle varie distanze che si riescono a rilevare e dello sbilanciamento di colli trattati in ogni specifico impianto. Ad esempio, se un impianto A tratta 80% di una classe B e il restante 20% diviso tra classi C e D, il sistema avrà un comportamento specifico che tenga conto di questo fattore (nei casi dubbi, in determinate condizioni, si classificherà il collo come B in modo tale da avere l'80% di probabilità di effettuare una classificazione corretta).

Tale soluzione risulta efficace nel caso in cui i colli da analizzare abbiano nella maggior parte dei casi configurazioni tali da ricadere nei casi preselezionati dall'operatore, ma non presenta una sufficiente robustezza e versatilità per essere utilizzato *"off-the-shelf"* per ciascun impianto.

Risulta dunque fondamentale cercare di superare questo scoglio con un metodo che sia più flessibile e slegato dall'applicazione *ad hoc* dello specifico impianto. Per ottenere questo risultato è necessario il superamento del *sistema esperto* basato su varie soglie, sfruttando le ormai note capacità degli algoritmi di *machine learning* in ambito di analisi dei dati. Per non rendere inutilmente obsoleto l'hardware degli impianti già in funzionamento si sfrutterà lo stesso sistema di visione e si utilizzerà uno storico di dati di cui l'azienda partner dispone come base per lo studio e per l'analisi che verrà spiegata nei successivi capitoli.

Trattandosi di un problema di *machine learning*, sarà dunque necessario attraversare tutte le fasi di studio del problema: dall'organizzazione e dall'analisi del dataset, passando per l'estrazione delle features, fino ad arrivare alla vera e propria strutturazione di una strategia di

---

[1] Tan, Chee Fai & Wahidin, L.S. & Khalil, Siti Nurhaida & Tamaldin, Noreffendy & Hu, Jun & Rauterberg, Matthias. (2016). The application of expert system: A review of research and applications.

classificazione legata alle features estratte. Alle varie strategie proposte sarà dunque poi necessario l'affiancamento di uno studio e di una analisi che permetta di scegliere il giusto *trade-off* tra accuratezza di classificazione e rapidità di esecuzione. Relativamente alla specifica parte temporale, la specifica richiesta è di effettuare la classificazione in un $\Delta t_{class} \ll 50ms$.

2. Formalizzazione del problema

Come in ogni problema di classificazione che sfrutti le tecniche di *machine learning,* è necessario disporre di una serie di dati etichettati che permettano di avere una rappresentazione delle classi da riconoscere durante l'esecuzione dei classificatori. Dallo storico della azienda si dispone di una sostanziosa serie di triple di immagini (*RGB, IR e DEPTH)* raccolti dai vari impianti e, inoltre, si ha la possibilità di creare nuovi campioni attraverso l'impianto demo disponibile presso la sede. Essendo i dati stati raccolti su tutti gli impianti con il sistema di visione che si intende utilizzare, possiamo ritenerli tutti significativi in ottica del problema da risolvere. Non essendo i dati organizzati, però, è necessaria la realizzazione di un applicativo che permetta di segmentare l'immagine e selezionare la regione che corrisponde a una relativa classe, assegnando a questa l'etichetta di pertinenza. I dati a disposizione sono raccolti in cartelle che racchiudono sia le triple di immagini dei vari campioni, sia una tripla particolare di *reference,* che rappresenta la situazione a nastro vuoto. Quest'ultima sarà sfruttata per eliminare il *background* delle immagini e limitare l'analisi alle sole zone significative.

2.1. Le classi

Prima di inoltrarci nella descrizione del programma e delle sue varie parti, è utile fornire una descrizione di massima delle classi. Di norma, nell'attività logistica, si distinguono tra loro sei sottofamiglie di colli, descritte di seguito:

- **Parcel**

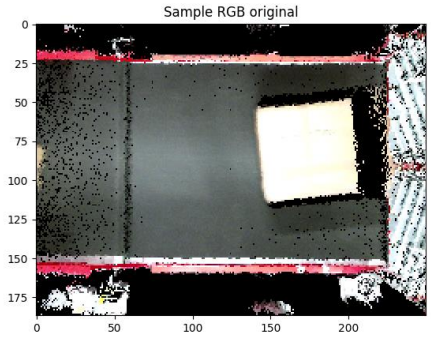


Figura 4-Esempio parcel

Rappresenta il pacco classico rigido, dalle misure più disparate che vanno da pochi centimetri fino a quasi il metro di altezza. La superficie risulta normalmente uniforme e la forma squadrata è sempre ben definita.

- **Envelope**

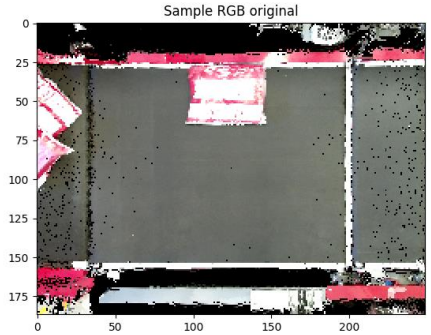


Figura 5-Esempio envelope

Rappresenta i colli senza una protezione rigida inseriti in una busta (solitamente di plastica). Questa classe è caratteristica dei colli spediti ai clienti dalle attività di *e-commerce* contenenti vestiti, presenta superficie e morfologie raramente uniformi.

- **Enveloped parcel**

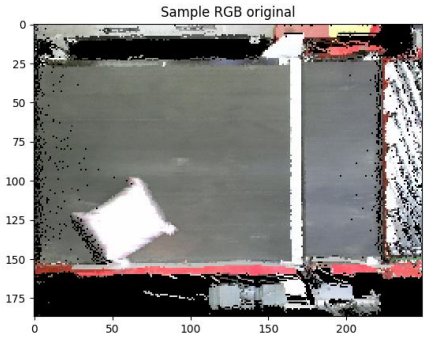


Figura 6-Esempio enveloped parcel

Rappresenta una sottofamiglia della classe *envelope*, che acquistando sempre più importanza in quanto a numeri, si è deciso di trattare separatamente. Risulta di fatto un *parcel* inserito in un *envelope.*

La sua distinzione dalla classe *envelope* si può presupporre essere più ostica delle altre per la somiglianza delle caratteristiche morfologiche che le contraddistingue.

- **Flyer**

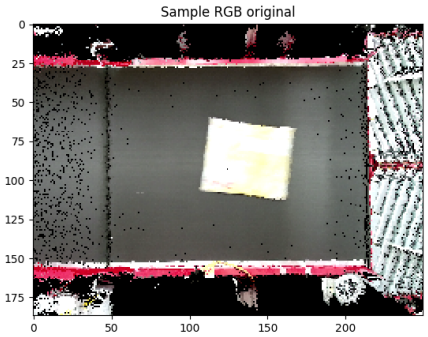


Figura 7-Esempio flyer

Rappresenta di fatto le buste rigide dello spessore di pochi centimetri. Le sue caratteristiche principali sono dunque forme regolari, superfici uniformi e una altezza non superiore ai 1 o 2 cm.

- **Double**

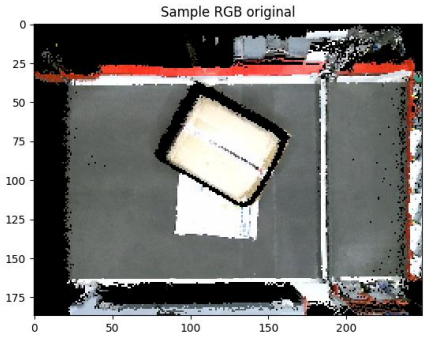


Figura 8-Esempio double

Rappresenta la situazione più critica da trattare, verso cui vanno indirizzati tutti gli sforzi. Se, per esempio, una errata classificazione di un *envelope* con un *enveloped parcel* non creerebbe grossi problemi nel seguito dell'impianto, non riconoscere un caso *double* potrebbe causare errori di manipolazione e di riconoscimento delle etichette. Sebbene difficilmente si possa raggiungere una accuratezza pari al 100% dei casi, il sistema di classificazione è preferibile abbia la massima accuratezza ottenibile, anche se questo volesse dire ridurre l'accuratezza verso la classificazione delle altre classi. La difficoltà insita in questa classe è data dal fatto che "*double*" non indica un corpo fisico ben preciso, come potrebbero essere le altre classi; indica invece una situazione che si verifica con infinite possibilità di combinazioni delle altre famiglie di colli.

- **Other**

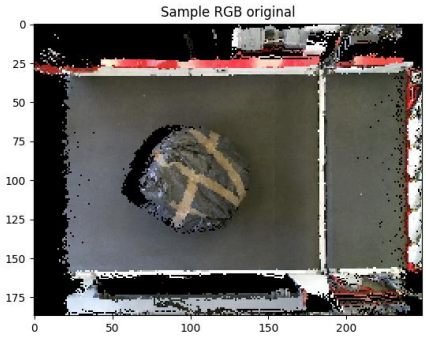


Figura 9-Esempio other

Rappresenta tutto ciò che non ricade nelle precedenti classi.

3. Il software per l'analisi e l'etichettamento

Nonostante sia possibile scegliere tra numerosi programmi per l'etichettamento del dataset, nessuno di questi permette di realizzare questo processo in maniera efficace per il problema trattato. Si decide dunque di procedere con una creazione di un programma che rispecchi anche in sequenza le operazioni che poi verranno svolte dalla macchina durante l'analisi. Utilizzando le stesse librerie *open source*, come *open CV*[2], è possibile comprendere la robustezza della stessa procedura che sarà utilizzata anche in macchina per estrarre i contorni, fondamentali sia per la segmentazione durante l'etichettamento, sia per l'estrazione delle feature (fase successiva). Avremo dunque la presenza di fasi di:

1. Creazione di una *reference*
2. Correzione digitale dei campioni raccolti

---

[2] I. Culjak, D. Abram, T. Pribanic, H. Dzapo and M. Cifrek, "A brief introduction to OpenCV," *2012 Proceedings of the 35th International Convention MIPRO*, Opatija, 2012

3. Sottrazione della reference ai nuovi campioni
4. Selezione dei contorni e segmentazione
5. Etichettamento delle sotto parti di immagini

Figura 10- Sequenza operazioni software etichettamento

Queste fasi, come in Figura 10- Sequenza operazioni, sono successive l'una all'altra e sono le stesse fasi che si susseguiranno una volta che l'impianto sarà avviato e utilizzerà i classificatori realizzati, in modo tale che le features in *input* siano quanto più possibile coerenti con quelle utilizzate per il *training* e il *testing* qui riportati.

Dunque, una volta implementati i classificatori sul sistema reale, la procedura sarà la stessa, sostituendo all'etichettamento manuale la fase di estrazione delle *features* dei contorni [3]e la classificazione, come da Figura 11.

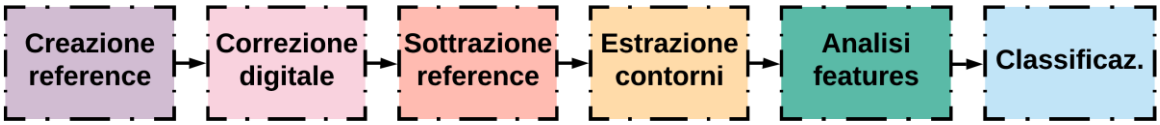


Figura 11- Sequenza operazioni software on-line

3.1. Creazione della reference

Si predispongono le immagini *RGB*, *IR* e *DEPTH* perché vengano poi utilizzate per essere sottratte ai vari campioni raccolti. Non essendo necessario l'utilizzo di tutta l'immagine per l'analisi, poiché si ha a disposizione per ogni impianto anche una *Region Of Interest (ROI*),

[3] Papari, Giuseppe & Petkov, Nicolai. Edge and line-oriented contour detection: State of the art, 2011.

si provvederà al ritaglio delle immagini per far sì che l'etichettamento avvenga nelle sezioni di misura delineate dalla *ROI.* Non influendo i disturbi in maniera significativa sulla *reference*, si rimanda la spiegazione delle tecniche di correzione digitale delle immagini nel seguente paragrafo dove i risultati saranno più apprezzabili e significativi, presentando, da Figura 12-Rgb original a Figura 14-Rgb without zeros and spikes, solo i risultati relative alla reference.

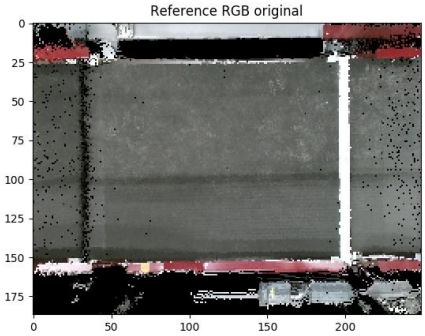


Figura 12-Rgb original

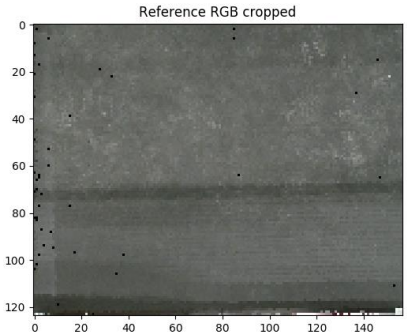


Figura 13-Rgb cropped

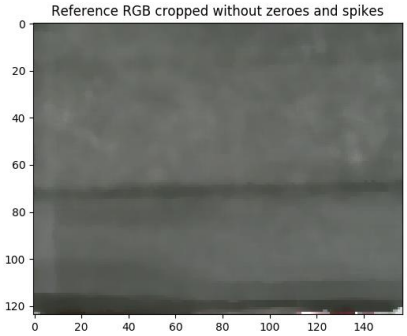


Figura 14-Rgb without zeros and spikes

3.2. Correzione delle immagini

È da notare la presenza di disturbi riguardo la *DEPTH e RGB.* I disturbi sono maggiori nelle aree laterali delle immagini e si configurano nelle stesse posizioni nella matrice dei pixel in entrambe le immagini. Questo si verifica in quanto, dal sistema di misura, la matrice *RGB* viene riempita solamente nei punti dove il sensore di profondità identifica la presenza di un corpo.

I tipi di disturbi sono principalmente di tre tipi:

- Zeri: l'immagine non presenta il suo valore originale che avrebbe avuto nella realtà e si trova un punto isolato a con valore nullo.
- Picchi: l'immagine presenta un valore eccessivamente alto rispetto i punti intorno ad esso.
- Ombre: l'immagine presenta un prolungamento della forma del corpo che porta dunque in errore rispetto alle reali dimensioni dei colli.

Una volta tagliata l'immagine in entrata secondo le misure indicate dalla *ROI* e avendone dunque individuato la regione di analisi, è necessario rimuovere gli zeri, i picchi e le ombre. Questa operazione è fondamentale per permettere la buona riuscita dell'estrazione dei contorni successiva, perché rimuovendo le ombre sarà possibile avere contorni che morfologicamente rappresentino con biunivocità la forma del collo in transito.

Dunque, presentando le fasi intermedie di questa operazione, si presentano in Figura 15, Figura 16 e Figura 17, le triple di immagini con i principali risultati intermedi, così da permettere al lettore di apprezzare l'efficacia della correzione dei dati in ingresso.

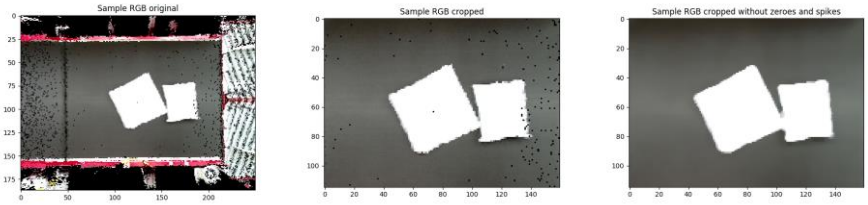


Figura 15-Correzione RGB

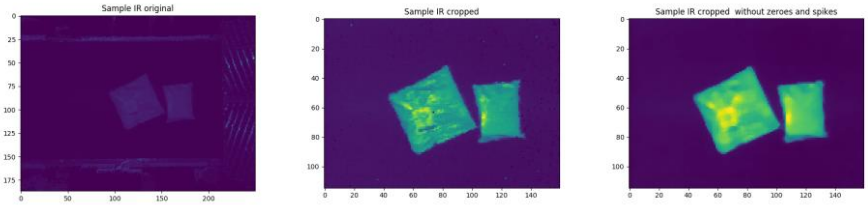


Figura 16-Correzione IR

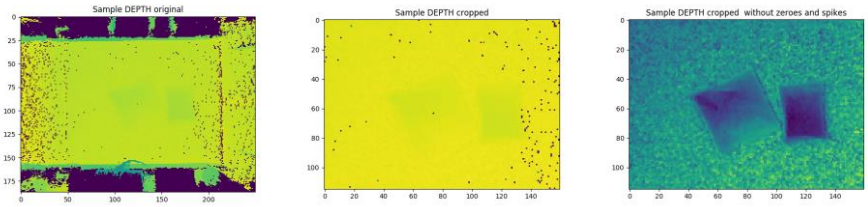


Figura 17-Correzione DEPTH

È immediato notare come la selezione di un'area più ridotta e la rimozione degli zeri permetta di far risaltare nella *DEPTH* i colli anche all'occhio umano, grazie al cambiamento del fondo scala di visualizzazione.

3.3. Sottrazione dello sfondo

Il passo successivo riguarda la rimozione del background per ottenere una forma degli oggetti risulti più delineata e decisa. Per far ciò, alla matrice risultato dell'operazione di correzione sì sottraggono le rispettive sezioni della reference, ottenendo perciò tre immagini distinte *RGB, IR* e *DEPTH,* rappresentate in Figura 18, che rappresentano le misure del corpo più marcate.

È necessario notare, per quanto riguarda la *DEPTH,* che il risultato della sottrazione risulta essere il valore effettivo dell'altezza del corpo. Con la sottrazione si va infatti a sottrarre al valore inizialmente misurato, comprendente l'altezza del collo e del nastro, l'altezza del

nastro stesso. È importante tenere a mente questa caratteristica, in quanto di fondamentale importanza nella fase di *estrazione delle features*[4].

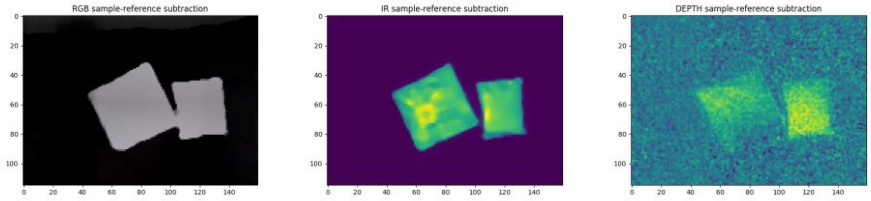


Figura 18- Output rimozione background

3.4. Estrazione dei contorni

Una volta ottenute tre immagini con una ridotta presenza di sfondo (nella *Depth* a causa di disturbi avremo uno sfondo con valori di media intorno allo zero) si ha la possibilità di estrarre tre differenti maschere. Per maschera si intende una immagine binaria con valori 0-1, dove 1 indica la presenza di valori sopra una certa soglia nell'immagine da cui la maschera viene estratta[5]. Lavorando sulle tre immagini separatamente e unendole in una fase finale in modo "intelligente" possiamo irrobustire l'estrazione successiva dei contorni.

Come si nota dai risultati delle fasi precedenti, le tre immagini presentano caratteristiche diverse. L'immagine *RGB* è affetta dal disturbo di tipo "ombra" di cui è stato parlato nel paragrafo precedente, mentre la maschera della immagine *IR* ne è immune.

Estraendo le maschere dalle rispettive immagini, otteniamo i seguenti risultati:

---

[4] Mahamkali, Naveenkumar & Vadivel. OpenCV for Computer Vision Applications, 2015.

[5] S. Cho, R.M. Haralick, and S. Yi, Improvement of Kittler and Illingworth's Minimum Error Thresholding," Pattern Recognition, 1989.

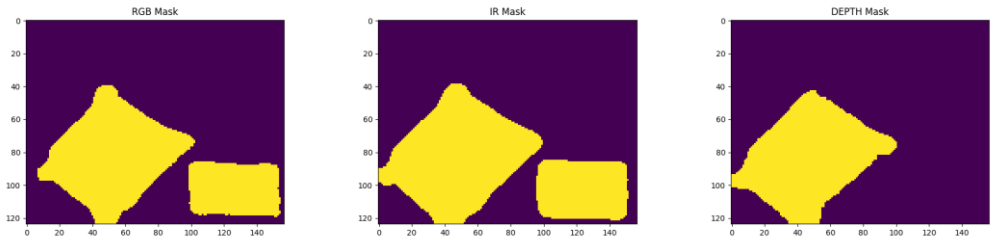


Figura 19-Maschere risultato RGB-IR-DEPTH

Si può notare come nella *DEPTH* il collo di destra non sia stato incluso nella maschera, questo si deve al fatto che il valore del collo in tale immagine è inferiore alla varianza del disturbo e viene dunque nascosto. Per superare questo problema ritorna, come già sostenuto, sfruttare la tripla informazione per irrobustire la selezione.

Chiamate $Mask_{RGB}$, $Mask_{DEPTH}$ *e* $Mask_{IR}$ [6]le maschere estratte rispettivamente da *RGB*, *DEPTH* e *IR*, la maschera complessiva sarà dunque il risultato delle seguenti operazioni*:*

$$Mask_{temporary} = Mask_{RGB} \ \textbf{BitwiseAND} \ Mask_{IR}$$

$$Mask_{def} = Mask_{temporary} \ \textbf{BitwiseOR} \ Mask_{DEPTH}$$

Dunque, la una maschera temporanea andrà a costituire una matrice binaria che avrà valori 1 dove sono 1 RGB e IR. Consecutivamente la maschera finale sarà 1 dove la maschera temporanea o la *DEPTH* avranno valori 1. In questo modo, combinando tutte le informazioni a nostra disposizione, possiamo ottenere una maschera complessiva che rappresenta robustamente la situazione dei colli sottostanti il sistema di visione, di cui è riportato un esempio in Figura 20[7].

---

[6] Bora, Dibya. Importance of Image Enhancement Techniques in Color Image Segmentation: A Comprehensive and Comparative Study. Indian Journal of Scientific Research, 2017.

[7] Cruz, Leandro & Lucio, Djalma & Velho, Luiz. Kinect and RGBD Images: Challenges and Applications, 2012.

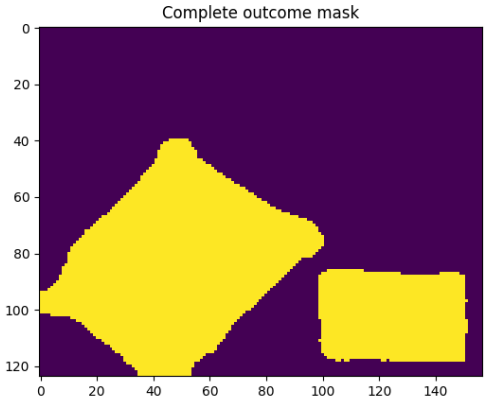


Figura 20-Maschera complessiva risultato

Estraendo da questa immagine i contorni e riportandoli sulla immagine originale *RGB* è possibile notare come specialmente il corpo di destra presenti l'effetto ombra e come questo sia stato con successo escluso nell'estrazione del contorno[8], raffigurato in verde in Figura 21.

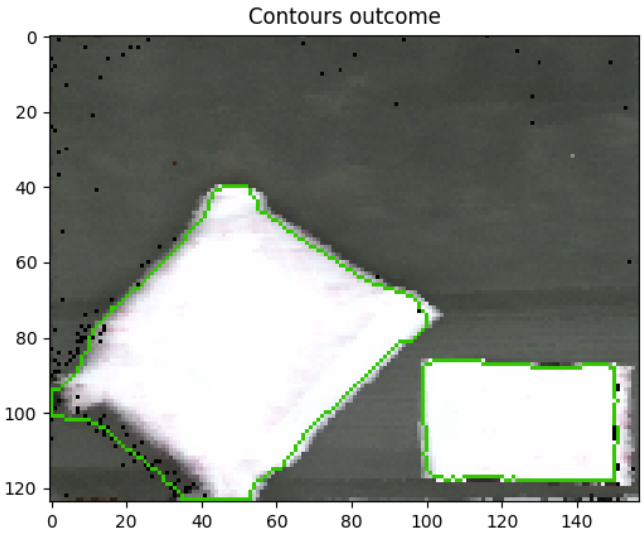


Figura 21-Contorni

[8] A, Dziech & Besbas, Wajdi & Nabout, Adnan & N, EldinFast Algorithm for Closed Contour Extraction, 1997.

Si presentano di seguito, da Figura 22 a Figura 24 , per completezza di trattazione altri esempi di contorni calcolati combinando le varie maschere, sempre raffigurandoli sull'immagine originale *RGB*.

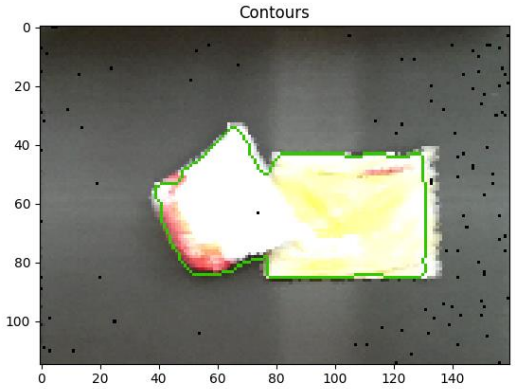


Figura 22-Esempio contorni

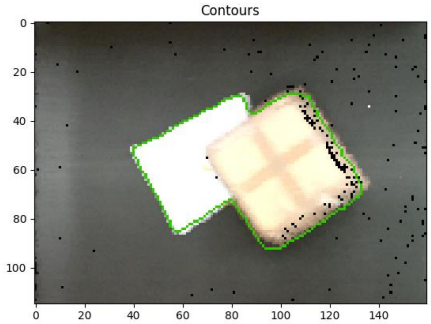


Figura 23-Esempio contorni

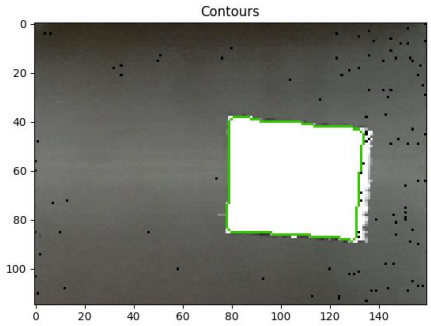


Figura 24-Esempio contorni

3.5. Etichettamento

Con la costruzione di una semplice, ma utile, interfaccia grafica di Figura 25 è possibile dunque rappresentare tutte e tre le informazioni (*RGB, IR e DEPTH)* contemporaneamente per aiutare l'operatore a meglio comprendere la natura del collo da etichettare. Sebbene le defferenti nature tra le diverse famiglie siano facilmente distinguibili ad occhio nudo nel momento in cui il collo logistico fosse realmente di fronte alla persona che deve effettuare la decisione, risulta sovente complesso comprendere la classe dal solo caso *RGB.*

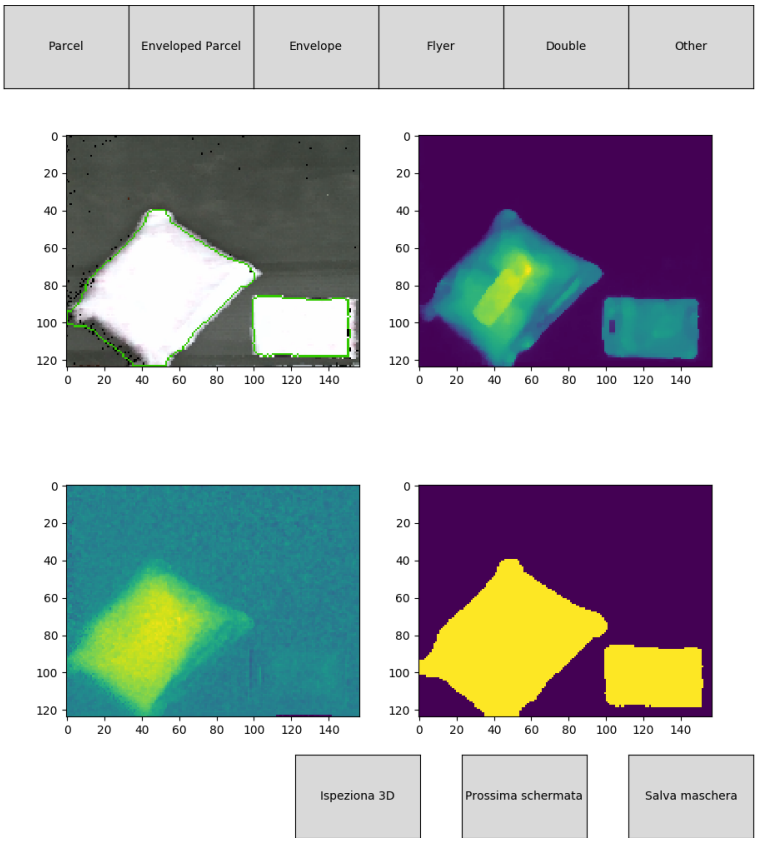


Figura 25-Menù selezione classi

Tale interfaccia permette di selezionare ciascun corpo contornato ed assegnargli la *label* di pertinenza. Per i casi più ostici è stata implementata la possibilità di una "*ispezione 3D"* che riporta la *DEPTH* in tre dimensioni per capire con più facilità l'etichetta da assegnare. Un esempio è riportato in Figura 26.

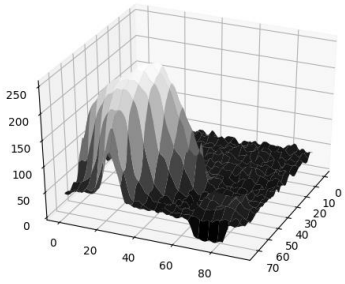


Figura 26-Ispezione 3D

Una volta finito l'etichettamento relativo alla tripla di immagini, viene mostrato il risultato e si offre la possibilità di rieffettuare l'etichettamento, o di procedere all' etichettamento tripla successivo.

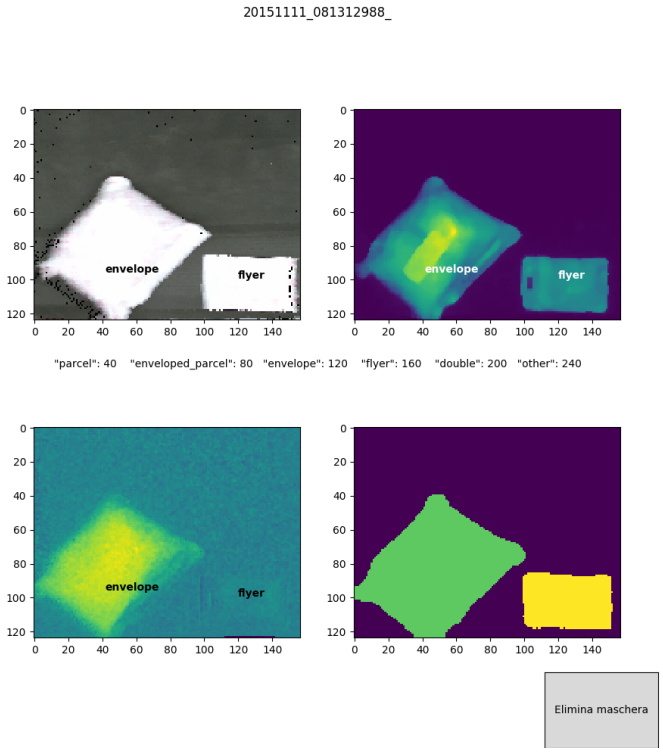


Figura 27- Controllo etichettamento

I differenti colori della maschera rappresentata in Figura 27 fanno comprendere che ad ogni assegnazione di classe viene sostituito alla maschera un valore dei pixel che corrisponde

univocamente alla classe selezionata. In tal modo il dataset etichettato conterrà le etichette in una nuova immagine salvata nelle cartelle analizzate con valori tabellati di seguito:

CLASSE	VALORE PIXEL
Parcel	40
Enveloped Parcel	80
Envelope	120
Flyer	160
Double	200
Other	240

A questo punto si dispongono di tutti gli strumenti per effettuare un etichettamento efficace e versatile e, qualora siano presenti più oggetti distinti in un singolo frame, sarà possibile etichettarli separatamente.

4. Analisi del dataset

Le immagini a disposizione per la creazione del dataset corrispondono a oltre 7000 triple *RGB*, *DEPTH* e *IR*, raccolte su cinque impianti diverse in sessioni di campionamento diverse. A seconda dell'impianto sono stati raccolti dati in modalità diverse: frame singoli, sequenze di frame, solo casi speciali, oppure solamente un determinato tipo di famiglia di colli (i più problematici).

4.1. Rappresentazione classi

Una volta scartati i casi di ripetizione del pacco, scartate le immagini a nastro vuoto e altri casi che non avrebbero giovato lo studio dei classificatori, otteniamo un totale di 1752 immagini, così suddivise nelle varie classi:

- *Parcel*: 553 campioni (31,5% del totale)
- *Envelope*: 420 campioni (24,1 % del totale)
- *Enveloped* parcel:162 campioni (9,3 % del totale)
- *Flyer*: 143 campioni (8,2 % del totale)
- *Double*: 445 campioni (25,4 % del totale)
- *Other*: 29 campioni (1,5 % del totale)

È necessario dunque fare attenzione al fortissimo sbilanciamento del dataset[9]. Se le classi *Parcel*, *Envelope* e *Double* sono sufficientemente bilanciate, non si può dire lo stesso di *Enveloped parcel* e *Flyer*. La classe *Enveloped Parcel*, già ostica da distinguere anche ad occhio da un operatore su un impianto, risulta dunque ulteriormente penalizzata da un numero di campioni molto ridotto. Poiché, grazie a diverse metriche di *accuratezza* e a tecniche di

[9] Somasundaram, Akila & Reddy, U. Srinivasulu. Data Imbalance: Effects and Solutions for Classification of Large and Highly Imbalanced Data, 2016.

resampling e down sampling o all'uso di opportuni pesi, è possibile superare il problema dello sbilanciamento non è necessario prendere nuovi campioni. Tuttavia, questo non vale per la classe *other*: la sua scarsità di campioni è troppo marcata.

Si decide dunque di ridurre il numero di classi di studio, escludendo dalla trattazione il caso *other*. Questa scelta non si ritiene impattare sul valore del lavoro, poiché le occorrenze degli *other* sono talmente ridotte e diverse tra loro che un *training* di classificatori basati su 29 campioni sarebbe totalmente insufficiente. Per lo stesso motivo, riuscire a raccogliere nuovi campioni per implementare la classe fino ad un numero sufficientemente rappresentativo richiederebbe troppo tempo, per ottenere oltretutto un miglioramento iniquo vista la reale importanza della classe.

5. La selezione ed estrazione delle features

Trattandosi della parte fondamentale del *machine learning*, le features necessitano di particolare attenzione. I vari classificatori, infatti, costruiranno le logiche decisionali proprio sulla base delle features relative ai dati di training utilizzati.

A questo punto, grazie alla strategia precedentemente spiegata, abbiamo a disposizione per ogni corpo etichettato un contorno che lo rappresenti. Essendo i contorni precisi e robusti è utile trarne vantaggio ed estrarre da questi quante più possibili informazioni su cui basare poi la classificazione.

La libreria grafica *open-CV* che è stata utilizzata per l'estrazione dei contorni permette anche di estrarne numerose e validissime caratteristiche. Essendo la stessa libreria da utilizzare in produzione, risulta ottimo utilizzarla anche per estrarre le *features* dal dataset. [10]

Di seguito una trattazione specifica per tutte le *features* che si è scelto di estrarre, per permettere al lettore di tenere traccia dei valori su cui verrà effettuata la classificazione. Ove possibile sarà presente anche la trattazione con le immagini rappresentanti le *features* (come di consueto sull'immagine *RGB)* per permettere al lettore di avere una più veloce comprensione.

5.1.Contour: area e perimetro

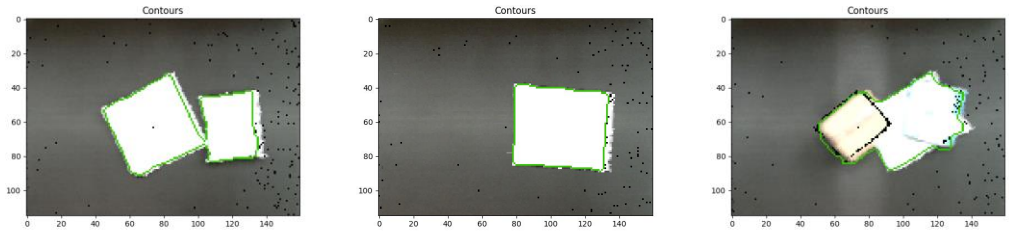


Figura 28-Contorni

[10] Hanbay, Kazım & Golgiyaz, Sedat & Fatih Talu, Muhammed. Real time fabric defect detection system on Opencv platforms,2017.

L'area del contorno misura il numero di pixel contenuti all'interno della linea indicata in verde (Figura 28) e viene indicata con cnt_{area}. Ugualmente è possibile utilizzare il contorno per calcolarne la lunghezza, ovvero il numero dei pixel che lo pongono, indicato con $cnt_{perimeter}$

5.2.Bounding box: altezza, larghezza e area

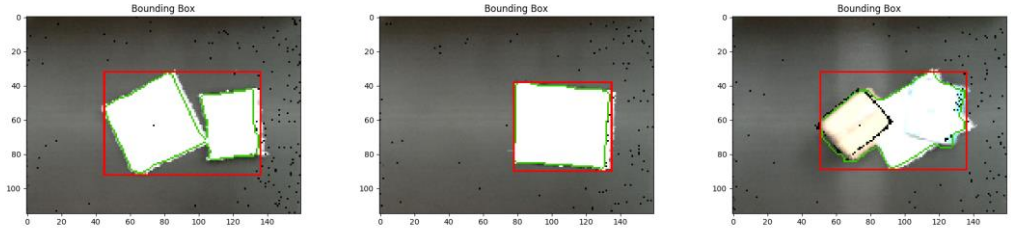


Figura 29-Bounding box

Il rettangolo che racchiude l'intero contorno con le linee parallele ai bordi dell'immagine (quindi perfettamente orizzontali e verticali) prende il nome di *bounding-box*, indicato in rosso in Figura 29. Da questo è possibile estrarre utilissime informazioni come altezza, spessore e area del rettangolo, rispettivamente: box_{height}, box_{width} e box_{area}.

5.3.Aspect ratio

Mettendo in relazione tra di loro le misure riguardanti la *bounding-box.*Per avere una indicazione utile che permetta di capire se il corpo si sviluppa in verticale o in orizzontale è utile stabilire il rapporto che prende il nome di *aspect ratio.*

$$aspect\ ratio = \frac{box_{width}}{box_{height}}$$

5.4.Extent ratio

Altro rapporto di grande importanza è indicato come *extent ratio* e mette in relazione le aree di contorno e bounding box.

$$extent\ ratio = \frac{cnt_{area}}{box_{area}}$$

Poiché box_{area} sarà sempre maggiore di cnt_{area} (per definizione *bounding box* è il rettangolo che racchiude il contorno), questo rapporto sarà contenuto tra i valori 0-1.

5.5.Hull Area

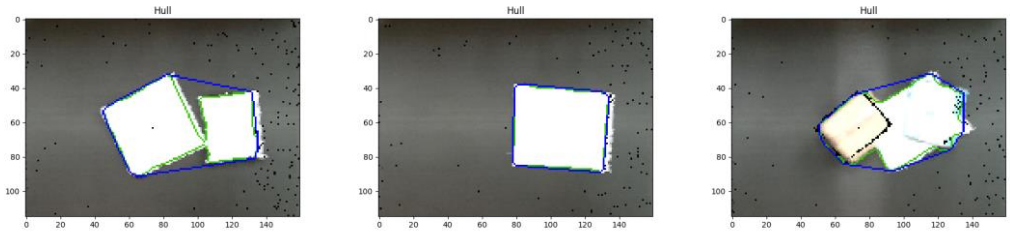


Figura 30-Hull

In modo simile alla costruzione della *bounding-box,* un'altra entità che è utile prendere in considerazione è *hull*, ovvero il guscio che contiene i punti estremi del contorno. Quando i colli avranno una forma simil rettangolare è immediato notare come questo andrà a sovrapporsi quasi perfettamente al contorno (*flyer, parcel),* nei casi invece di un pacco con forma non definita come *double* e *envelope,* si nota un discostamento tra *hull* (linea blu, in Figura 30) e *contour* (linea verde). Di questo guscio, che di fatto è come fosse un ulteriore contorno, è possibile calcolarne l'area, ovvero i pixel da esso contenuti, che indicheremo con $hull_{area}$.

5.6.Contour solidity

Definito il concetto di *hull*, è utile definire un ulteriore rapporto, detto *solidity* che relaziona le aree del contorno e del guscio intorno ad esso.

$$solidity\ ratio = \frac{cnt_{area}}{hull_{area}}$$

Come detto per *extent ratio,* anche questo rapportò avrà valori compresi tra 0-1.

5.7.Ellisse equivalente: asse maggiore ed asse minore

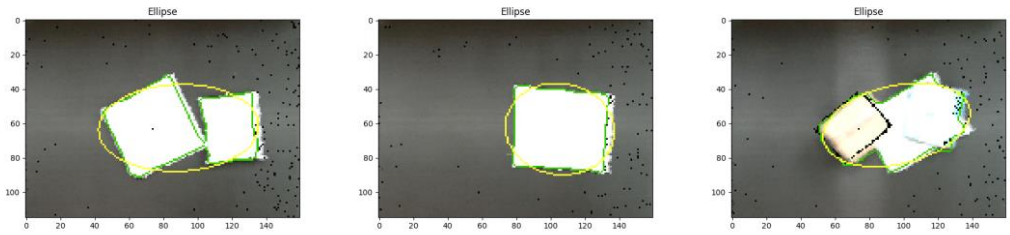


Figura 31-Ellisse equivalente

Dal contorno è possibile calcolare l'ellisse equivalente, rappresentata in giallo in Figura 31, che con il contorno condivide il centro. L'ellisse avrà area uguale all'area del contorno e di questa si possono individuare asse maggiore ($axis_m$), asse minore ($axis_M$) ed il loro rapporto $\frac{axis_m}{axis_M}$, detto *axis ratio.*

5.8.Convexity defects

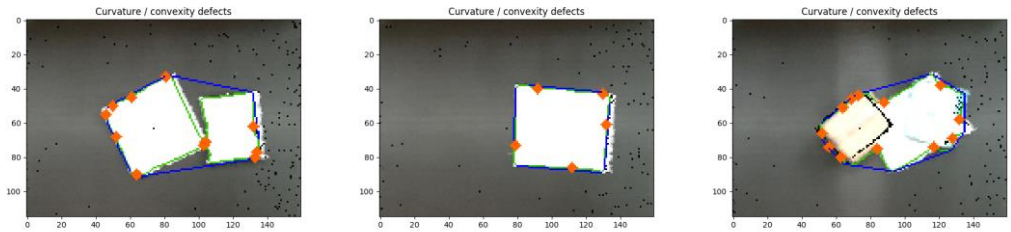


Figura 32-Convexity defects

Sfruttando le due entità sopra spiegate, ovvero il *contorno* e la relativa *hull*, è possibile individuare i punti di disomogeneità, dove la figura non è regolare e il contorno si allontana oltre ad un certo numero di pixel dalla *hull.* Si ottiene un vettore di punti (rappresentati in arancione in Figura 32), la cui lunghezza può essere sfruttata come informazione sul numero di difetti e dunque il grado di irregolarità del collo in fase di misura. La lunghezza di questo vettore prende il nome di *curvature.*

5.9. Minimum rectangle area

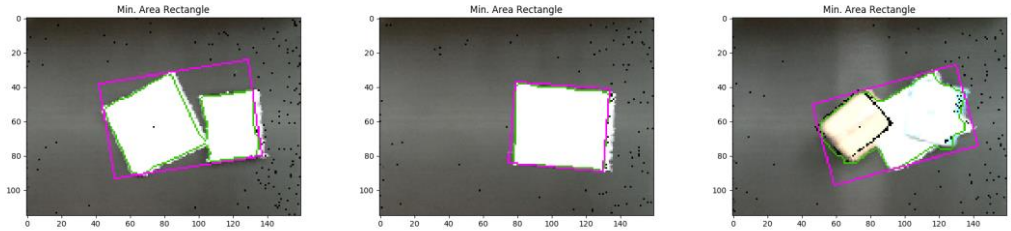


Figura 33-Minimum area rectangle

Unendo tra di loro i concetti di *bounding-box* e di *ellisse equivalente*, parallelamente è possibile calcolare il rettangolo di area minima *minAreaRect*, raffigurato in viola in Figura 33,

che racchiude il contorno trovato. In questo caso è importante notare che il rettangolo non ha i lati orizzontali e verticali e dunque segue la rotazione degli assi del collo misurato.

Di questo rettangolo, così come per le precedenti entità, ne estraiamo l'area indicandola con $minAreaRect_{area}$.

5.10. Extent minima second

Come già proposto per il rapporto tra area del contorno e della *bounding box*, riprendiamo ora il concetto di *extent ratio* per il rettangolo ad area minima, rapportando sempre il valore dell'area del contorno con il rettangolo di area minima.

$$extent\ second\ = \frac{cnt_{area}}{minAreaRect_{area}}$$

5.11. Solidity minima

Ultimo rapporto importante per la classificazione prende nuovamente il valore dell'area del rettangolo di area minima e la rapporta all'area della *bounding box.* Questo perché, quando un corpo è perfettamente orizzontale e i corpi hanno un basso livello di difetti, il rapporto tenderà ad 1.

$$solidity\ minima\ = \frac{minAreaRect_{area}}{box_{area}}$$

5.12. HU moments

Nell'elaborazione delle immagini ed in visione artificiale il momento di un'immagine, in analogia con il concetto di momento, è una particolare media dell'intensità dei pixel che compongono l'immagine. In senso più generale, sono detti momenti anche le funzioni di tali medie, che godono di particolari proprietà o caratteristiche. È possibile inoltre calcolare momenti che siano contemporaneamente invarianti a traslazione, cambiamenti di scala e

rotazione. L'insieme di momenti più usati che godono di queste proprietà è l'insieme dei sette *momenti di Hu.* [11]

5.13. Istogrammi

Avendo a disposizione la *DEPTH*[12], che dopo la sottrazione della reference ci permette di avere una rappresentazione fisica dell'altezza dei colli in visione (indipendentemente dall'impianto in cui ci si trova), è utile sfruttare anche questa informazione sotto forma di istogrammi[13]. Con la stessa libreria di *computer vision è possibile*, una volta selezionato il contorno, realizzare il conteggio dei pixel interni ad esso contenuti in determinati intervalli. In questo modo, definendo gli intervalli del conteggio dei pixel che ci interessano, avremo un conteggio con maggiore sensibilità quanto più grande sarà il numero di steps di conteggi. Ad esempio, se tra 0 e 10 impostassimo 10 steps avremo 10 diversi conteggi; se selezionassimo 5, invece, conteremo solo il numero dei pixel che ricade in ciascuno dei 5 intervalli.

È molto importante notare come, contando i pixel i questo modo, si perde il valore spaziale dell'informazione. Sapremo che ci saranno *n* pixel relativi ad un intervallo, ma non sapremo come saranno disposti.

5.13.1. Istogramma 0-400mm: 20 steps

Sfruttando questa caratteristica di sensibilità variabile, consapevoli che il maggior discernimento è necessario tra gli 0 e i 400mm, poiché in queste dovremo distinguere tra il diverso numero di colli, questo intervallo sarà analizzato da un istogramma caratterizzato da 20 *steps* di conteggio. Fisicamente questo significa che potremo distinguere intervalli di

---

[11] Huang, Zhihu & Leng, J. Analysis of Hu's moment invariants on image scaling and rotation, 2010.

[12] Zhong, Henry & Kanhere, Salil & Chou, Chun. QuickFind: Fast and contact-free object detection using a depth sensor, 2016.

[13] F. Tombari, S. Salti, L. Stefano, Unique Signatures of Histograms for Local Surface Description, 2010.

variazione di *20mm*. Perdendo il valore spaziale, se una *double* fosse costituito da più colli di altezze ben diverse tra loro avremo più picchi ben distinti nell'istogramma. Nel caso in cui le altezze cadessero nello stesso intervallo, l'istogramma avrà un unico picco.

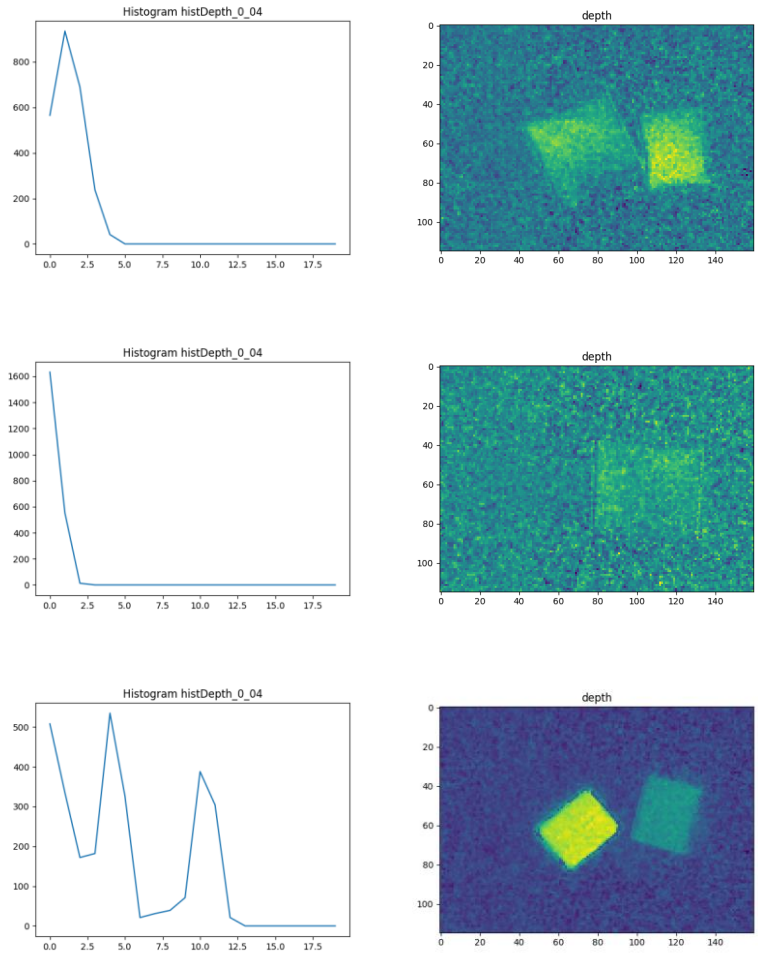


Figura 34-Istogrammi 0-400 mm

5.13.2. Istogramma 400mm-1000mm: 10 steps

Avendo meno occorrenze nel dataset (e dunque meno importanza) il caso in cui i colli superassero i 400mm, l'intervallo è stato settato con meno sensibilità, riducendola a 10 step. Il massimo di 1000mm è stato settato da specifiche di impianto, non predisposto per processare colli con dimensioni maggiori di 1000mm. Nei tre casi in esame come esempio, nessuno supera i 400mm di altezza e pertanto si nota che tutti i valori sono nulli dentro l'istogramma.

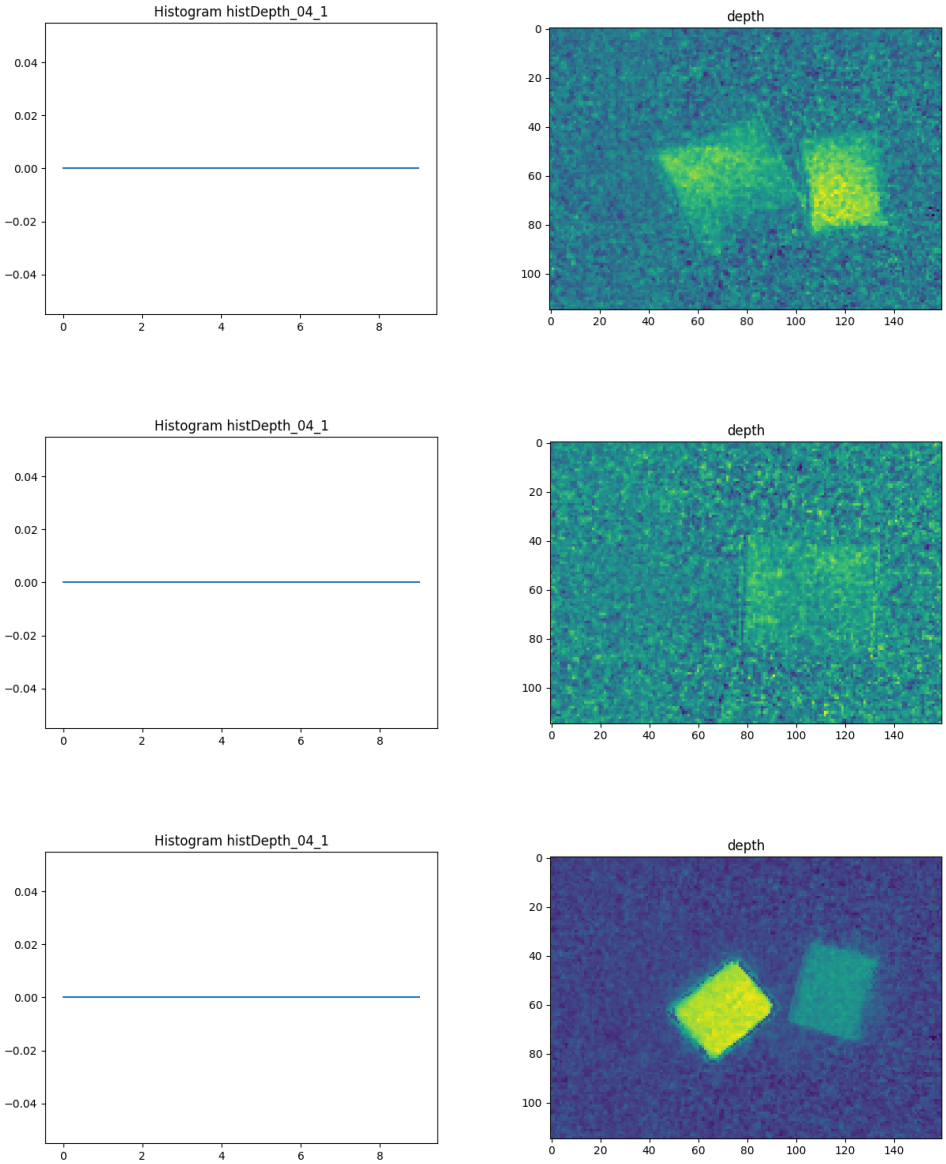


Figura 35-Istogrammi 400-1000 mm

5.14. Invarianti alla rotazione

È necessario effettuare una distinzione. Le features ora indicate si dividono in due fondamentali famiglie: *varianti* e *invarianti* [14] alla rotazione:

[14] Ye, Zhang & Hongsong, Qu. Rotation invariant feature lines transform for image matching, 2014.

- *Varianti*: sono le *features* su cui influisce il fatto che il corpo venga analizzato a differenti angoli di rotazione. Tra le *features* selezionate sono varianti alla rotazione: box_{height}, box_{width} e box_{area}, $extent\ ratio$, $solidity\ ratio$,
- *Invarianti*: sono le *features* su cui **non** influisce il fatto che il corpo venga analizzato a differenti angoli di rotazione. Tra le *features* selezionate sono invarianti alla rotazione: cnt_{area}, $cnt_{perimeter}$, $hull_{area}$, $axis_m$, $axis_M$, *axis ratio, curvature,* $minAreaRect_{area}$, $solidity\ minima$, *extent second, istogrammi* ed, infine, i *momenti di HU.*

Ogni singola misura, dunque, dopo la fase di estrazione delle *features* darà come risultato un vettore di valori ciascuno corrispondente ad ogni *feature* estratta: avendo 23 *features* legate direttamente alle caratteristiche del contorno e 20 + 10 per i due istogrammi, il vettore avrà, dunque, dimensioni 53x1.

Riassumendo:

Features	**Cardinalità**	**Invariante alla rotazione**
Area contorno	1x1	SÌ
Perimetro contorno	1x1	SÌ
Altezza Bounding Box	1x1	NO
Larghezza Bounding Box	1x1	NO
Area Bounding Box	1x1	NO
Aspect Ratio	1x1	NO
Extent Ratio	1x1	NO
Area Hull	1x1	NO
Solidity	1x1	NO
Asse maggiore	1x1	SÌ
Asse minore	1x1	SÌ
Rapporto Assi	1x1	SÌ
Convexity	1x1	SÌ
Area ettangolo minimo	1x1	SÌ
Extent minima second	1x1	SÌ
Solidity minima	1x1	SÌ
HU moments	1x7	SÌ
Istogrammi 0-400	1x20	SÌ
Istogrammi 400-1000	1x10	SÌ
tot	**1x53**	

6. Machine learning: strategie di classificazione e analisi risultati

In generale, per poter utilizzare gli algoritmi di apprendimento automatico, la prima cosa da fare è trovare quale sia il miglior criterio per separare le classi di dati del *training* set (fase di *training*). I dati raccolti permettono, una volta etichettati, di creare un *dataframe* su cui andare ad effettuare le operazioni di *training* e *testing* di 1723 righe per 54 colonne (53 di *features* + 1 *label*). Poiché si tratta di fatto di una serie ridotta di vettori non si presenterà alcuna difficoltà nell'effettuare il *training* e lo studio dei tempi di *training* non è per questo utile alla soluzione problema trattato.

Al contrario, trattandosi di una applicazione industriale dove la velocità di classificazione risulta fondamentale, è bene dare una introduzione qualitativa dei vari classificatori che sono stati utilizzati in modo tale da comprendere quali parametri possano più o meno impattare sul tempo di classificazione *online*.

Per questo studio sono stati utilizzati i classici classificatori di *machine learning: svm, knn, random forest, naive Bayes.* Vari tentativi sono stati fatti per testare l'utilizzo di *reti neurali*, tuttavia, come si presenterà di seguito, per il tipo individuato di soluzione e di dati a disposizione risulta più efficace una aggregazione delle *posterior probabilities* di *svm, random forest e knn* in quanto si sfrutta migliormente la spazialità dei dati e la ricorrenza di errori simili che il classificatore *naive bayes* riesce a riconoscere e ad arginare. Dunque, trattandosi di una tecnologia effettivamente scelta per le soluzioni proposte, non se ne considererà il tempo di predizione.

Inoltre, per quanto riguarda le accuratezze, è bene anticipare che gli sforzi per le strategie saranno volti a massimizzare l'affidabilità rispetto alla *double detection*, pur cercando ugualmente di tenere delle accuratezze di classificazioni nelle restanti classi che permettano di giustificarne la presenza nella trattazione rispetto all'esclusione come la classe *other*.

6.1. Analisi classificatori e complessità computazionale

Risulta utile per la trattazione utilizzare la seguente convenzione:

- Numero campioni di training: n_{train}
- Numero classi: $n_{classes}$
- Numero features: n_{feat}

6.1.1. SVM

Dato un insieme di campioni per l'addestramento detto *training set*, ognuno dei quali etichettato con la classe di appartenenza fra le possibili classi, un classificatore SVM costruisce un modello che assegnerà ai nuovi dati in input una etichetta di classificazione. Un modello SVM è una rappresentazione dei dati di *training* come punti in uno spazio ad n_{feat} dimensioni, mappati in modo tale che gli esempi appartenenti alle diverse classi siano chiaramente separati da uno spazio il più possibile ampio.

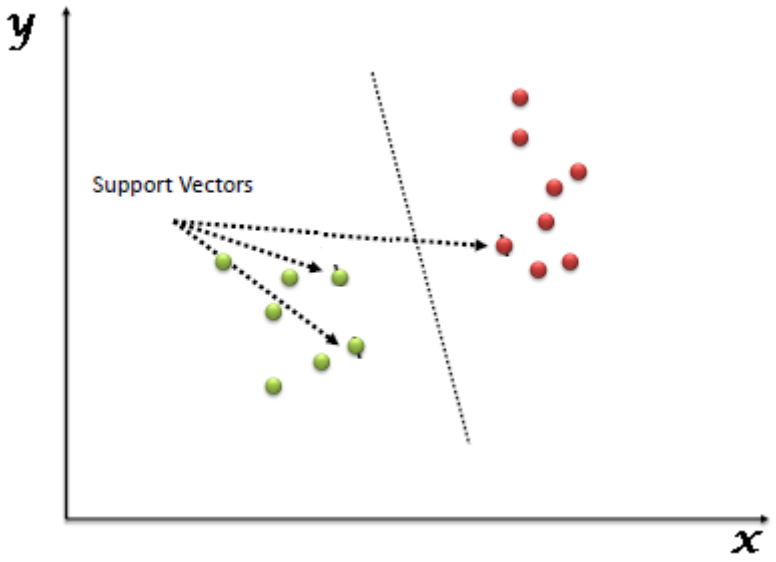


Figura 36-Esempio SVM

I nuovi campioni da classificare sono quindi mappati nello spazio e la predizione della classe di appartenenza viene fatta sulla base del lato nel quale questo campione ricade. Il classificatore calcola le distanze dai *vettori di supporto* (iperpiani, nel caso a n_{feat} dimensioni) individuati durante il *training*.

Bisogna notare che il classificatore *SVM* è un classificatore binario: dato un insieme di classi, questo viene ridotto di volta in volta a due classi: classe considerata contro classe del coniugato (nel nostro caso, e.g., *flyer VS not-flyer*)

Il calcolo delle distanze verrà dunque realizzato tante volte quante il valore di $n_{classes}$ e verrà predetta la classe al cui *iperpiano* il nuovo campione risulta più distante nella direzione della classe stessa. Nel caso lineare, per ogni nuovo campione verrà calcolata la distanza d dall'iperpiano relativo alla classe del nuovo punto x_0:

$$d = proj_w(x - x_0) = \frac{|x_0 * w + b|}{\|w\|}$$

Questa operazione viene realizzata tante volte quante le classi, cambiando iperpiano rispetto a cui viene calcolata la distanza.

6.1.2. Random Forest

Durante il training viene spezzato il vettore delle features in lotti e vengono utilizzati per creare più alberi di classificazione. In predizione, per l'assegnazione della classe viene scelta la classe che ottiene il maggior numero di voti dei singoli alberi che realizzano la foresta. Nella peggiore delle ipotesi, la lunghezza dell'albero non supera la lunghezza del lotto massimo di features, che risulta essere $n_{feat}/2$.

Dunque, trattandosi ogni scelta di proseguimento nell'albero (destra o sinistra) di fatto di una sottrazione e di una comparazione con la soglia, nel *worst-case scenario*, ovvero dove si deve percorrere tutti gli albero fino all'ultimo nodo per ottenere una classificazione, saranno da realizzare $N_{trees} * N_{nodes}$ sottrazioni.

6.1.3. K-Nearest Neighbor

Risulta essere di fatto la classificazione più onerosa in termini computazionali. Per un nuovo x_0 da classificare in fase di predizione, il classificatore calcola le distanze di questo

punto da tutti i campioni di training. Dunque, si verifica il calcolo di n_{train} distanze in uno spazio ad n_{feat} dimensioni. Riassumendo nel seguente pseudo codice:

1) For i = 1 to n_{train} DO:

compute distance (X_0 , X_i)

End for

2) Compute set of indices for K smallest distances and return majority label

Risulta quindi evidente che il classificatore avrà un tanto maggiore sforzo computazionale quanto maggiori saranno gli n_{train} e i n_{feat} .

6.1.4. Naive Bayes

Un classificatore bayesiano è un classificatore basato sull'applicazione del teorema di Bayes.

Il classificatore bayesiano richiede la conoscenza delle probabilità a priori e condizionate relative al problema, quantità che è possibile ottenere come risultato di classificazione dai vari classificatori sopra descritti. Per costruzione, il classificatore bayesiano minimizza il rischio di misclassificazione.

Durante il training viene ricavato con il *maximum likelihood method* [15]un modello di gaussiana multivariata per ogni classe attraverso gli n_{train} campioni di training a disposizione. Durante la classificazione si testa il vettore di input in modo da ottenere la probabilità del vettore di ingresso di appartenere a ciascuno dei modelli di riferimento delle classi. Viene infine scelta la classe che presenta la maggiore probabilità di appartenenza.

$$f_x(x_1, \dots, x_k) = \frac{exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))}{\sqrt{(2\pi)^k}|\Sigma|}$$

Dove k rappresenta il numero di features su cui è stato effettuato il training.

[15] Guo, Shenyang. Maximum Likelihood Estimator. Chinese Sociological Review, 2013.

Importante è notare che in questa fase gli n_{train} e n_{feat} saranno differenti da quelli utilizzati per i precedenti classificatori.

6.2. Strategia 1: strutturazione baseline

Come prima strategia di approccio al problema, nell' ottica di costruire una *baseline* con cui confrontare successive strategie più raffinate, si affronta la classificazione senza selezionare in alcun modo le *features*. Il *training* dei classificatori e l'*ottimizzazione* dei parametri di questi sono stati effettuati con la libreria *opensource* scikit-learn. Grazie a questa libreria i parametri dei classificatori possono essere agevolmente ottimizzati, attraverso la funzione *grid-search*, che permette di testare multiple combinazioni di *training* e di selezionare i migliori parametri in base alle prestazioni di *testing.* Tale strategia è stata utilizzata per tutti i classificatori di seguito riportati, che vengono dunque presentati nella configurazione ottima per il problema in questione.

Per tenere conto dello sbilanciamento del *dataset*, non volendo incorrere in rischi di *overfitting* in una fase preliminare per la delineazione della *baseline* di confronto, si è preferito non utilizzare metodi di *balancing* come il *resampling*, ma di tenere conto dello sbilanciamento pesando le varie classi con i pesi di rappresentazione di seguito descritti:

- Parcel: 0.32
- Envelope: 0.244
- Enveloped parcel: 0.094
- Flyer: 0.082
- Double: 0.26

Di seguito si presenta lo studio delle prestazioni dei singoli classificatori, con una particolare successiva attenzione ai vantaggi della loro combinazione in ottica di sfruttamento delle diversità di classificazione dei singoli.

Inoltre, verranno presentati i risultati relativi al tempo di classificazione medio richiesto per ciascuno dei classificatori. Questi tempi, essendo rilevati su un PC con stesse caratteristiche del PC utilizzato in produzione, risultano essere attendibili e sufficientemente veritieri per valutare le prestazioni complessive *offline*.

6.2.1. Random Tree

Dalla matrice di confusione si può notare che l'unica classe performante è *parcel*. Il classificatore *Random Tree* non è robusto [16]e non offre di fatto vantaggi per l'applicazione da realizzare.

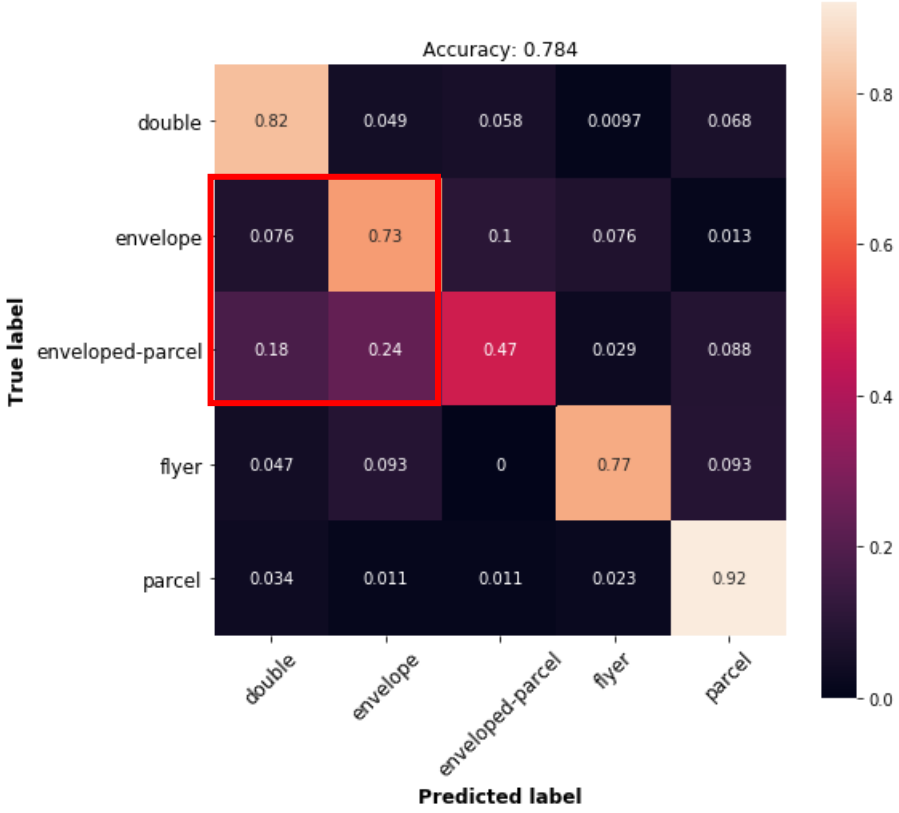


Figura 37-Matrice di confusione Random Tree

[16] Ali, K. On the link between error correlation and error reduction in decision tree ensembles, 1995.

È utile evidenziare, come d'altronde ci si aspettava in base a quanto espresso precedentemente, che la maggior parte della confusione nel sistema si crea a ridosso della distinzione tra *envelope* ed *enveloped parcel.* In queste condizioni di *performance* risulta di fatto inutile inserire la distinzione tra queste due classi, sarebbe più opportuno racchiuderle in una unica classe *envelope*. Per evitare di dover effettuare questa scelta, gli sforzi di classificazione saranno, oltre che per la classe *double*, anche per raggiungere una performance di classificazione delle altre classi tali da giustificarne l'esistenza nel problema.

Utilizzando la tecnica *k-fold cross validation* otteniamo una varianza di classificazione $\sigma_{class} = 0.0098$

6.2.2. Random Forest

Per sua natura il classificatore *Random Forest* [17] presenta una robustezza di classificazione maggiore rispetto al precedentemente trattato *Random Tree*. Trattandosi di un metodo *ensemble*[18], composto da *n* alberi di classificazione, presenta una varianza minore, così come una matrice di confusione più pulita. Tutte le classi subiscono un miglioramento: significativo nel *double* (8%); più ridotto, ma comunque apprezzabile, per la distinzione tra *envelope* ed *enveloped parcel.*

---

[17] Pretorius, Arnu & Bierman, Surette & J. Steel, Sarel. A meta-analysis of research in random forests for classification, 2016.

[18] Tumer, Kagan and Joydeep Ghosh. "Error Correlation and Error Reduction in Ensemble Classifiers." Connect, 1996.

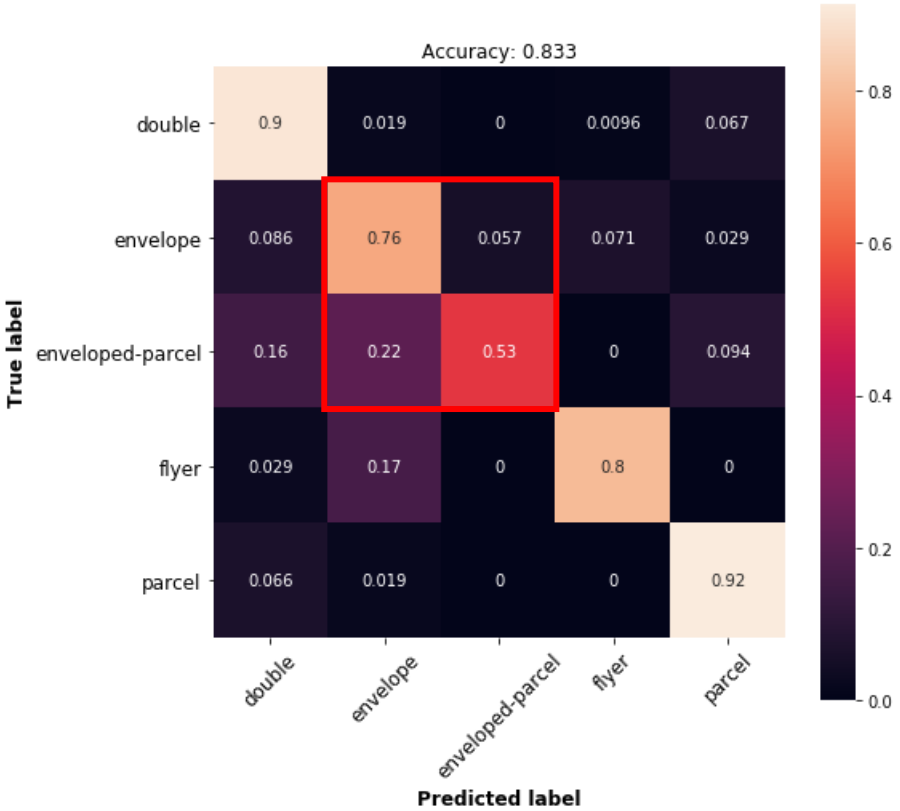


Figura 38-Matrice di confusione Random Forest

Dopo la fase di tuning ottimale dei parametri del classificatore, si riportano di seguito i valori scelti per il *training*:

- criterion="gini"
- max_features = "log2"
- n_estimators=200

Utilizzando la tecnica *k-fold cross validation* con $k = 10$ otteniamo una varianza di classificazione $\sigma_{class} = 0.008$, evidenziando nuovamente il vantaggio dell'utilizzo di un metodo *ensemble.*

Per quanto riguarda il tempo di classificazione, una volta effettuato il training e il testing, si è riclassificata una parte sostanziosa del *dataset* per poter ottenere un tempo di classificazione medio che fosse attendibile grazie all'uso di un numero consistente di campioni. Su un totale di 1000 classificazioni, il Random Forest prediction time risulta essere $AVG_Pred.time_{RF} =$ 0.0006 sec .

6.2.3. Support Vector Machine

Il classificatore SVM[19], realizzato con un kernel lineare, lavora di fatto su 53 dimensioni, tante quante sono le features. Spazialmente riesce in modo efficace a distinguere tra le varie classi di appartenenza, infatti nella matrice di confusione si nota un generale netto miglioramento per ogni classe, ad eccezione di *enveloped parcel*. Questa risente sia del ristretto numero di campioni su cui poter effettuare il training, sia della difficoltà intrinseca della distinzione tra *envelope* ed *enveloped parcel*.

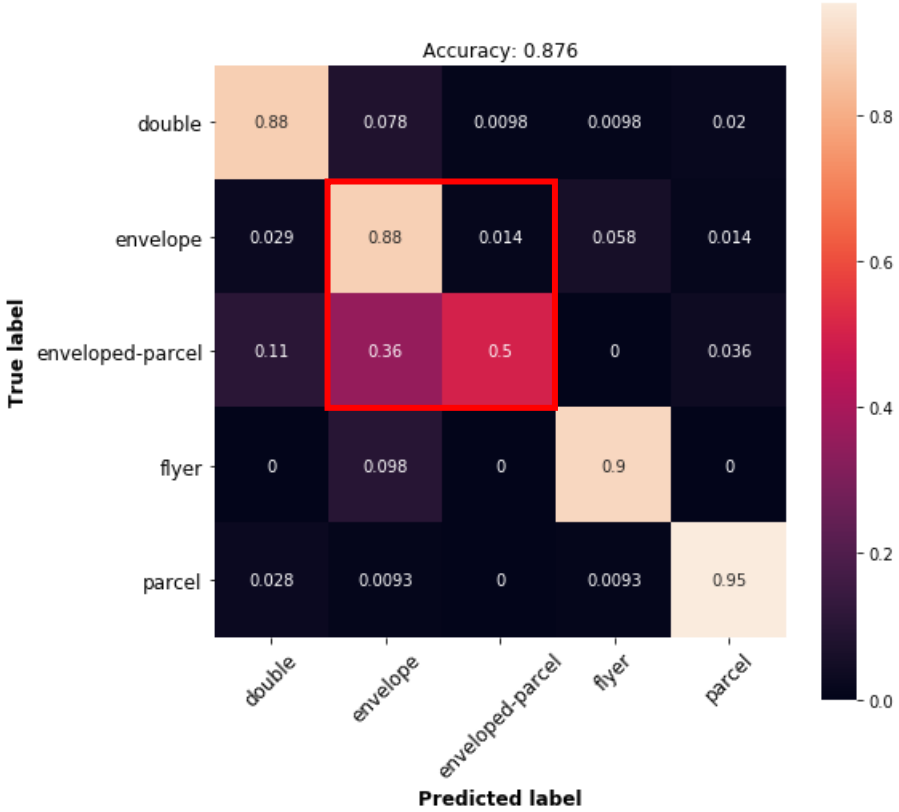


Figura 39-Matrice di confusione SVM

Con il solito metodo di *k-fold cross validation* con $k = 10$ otteniamo una varianza di classificazione pari a $\sigma_{class} = 0.006$, relativamente migliore rispetto al *Random Forest*.

[19] Chu, Wei & Sathiya Keerthi, S & Ong, Chong. A General Formulation For Support Vector Machines, 2002.

Riclassificando, così come prima, 1000 campioni del dataset otteniamo per il classificatore SVM un $AVG_Pred.time_{SVM} = 0.0005 \text{ sec}$.

6.2.4. K-Nearest Neighbor

K-nn [20]isulta di fatto essere il migliore classificatore preso singolarmente sul fronte di classificazione della classe *double*. Permane forte l'incertezza di classificazione tra *envelope* ed *enveloped parcel*, così come negli altri classificatori fino a qui rappresentati. La fase di *tuning* per ottenere le performance migliori ha permesso di selezionare i seguenti parametri di training:

- leaf_size = 10
- n_neighbors=3

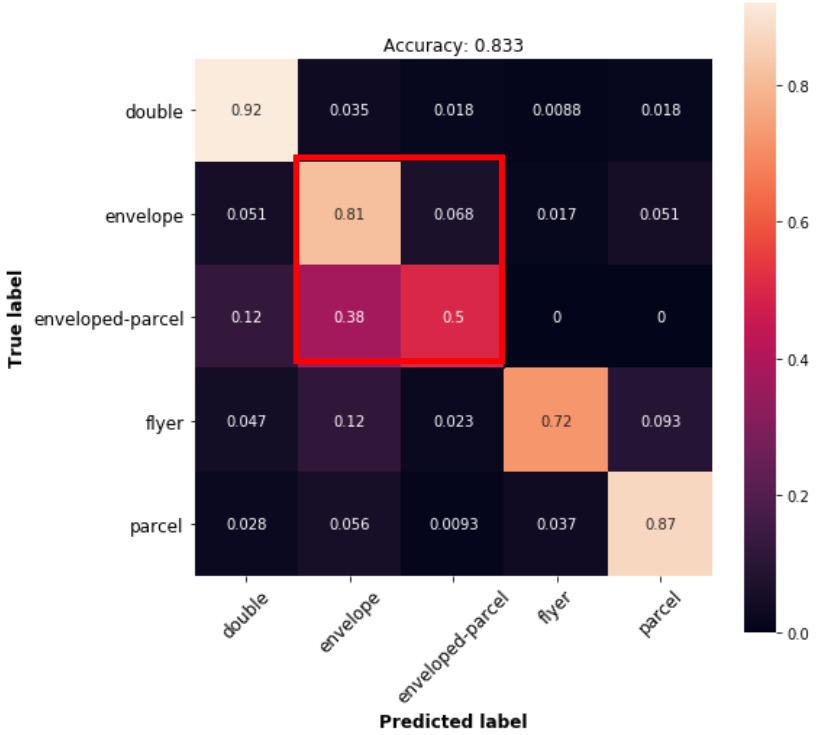


Figura 40-Matrice di confusione K-NN

[20]Altman. An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician, 1996.

Sottolineando quanto già espresso rispetto alla pesantezza computazionale di questo algoritmo, possiamo notare come, dovendo calcolare circa 1200 distanze per trovare la classe dei tre vicini al nuovo campione, il tempo medio di classificazione risulta maggiore dei classificatori precedenti. Riclassificando, così come prima, 1000 campioni del dataset otteniamo per il classificatore *K-NN* un $AVG_Pred.time_{K-NN} = 0.001\text{sec}$.

Per quanto riguarda la varianza di *k-fold cross validation* con *k = 10*, otteniamo $\sigma_{class} = 0.007$

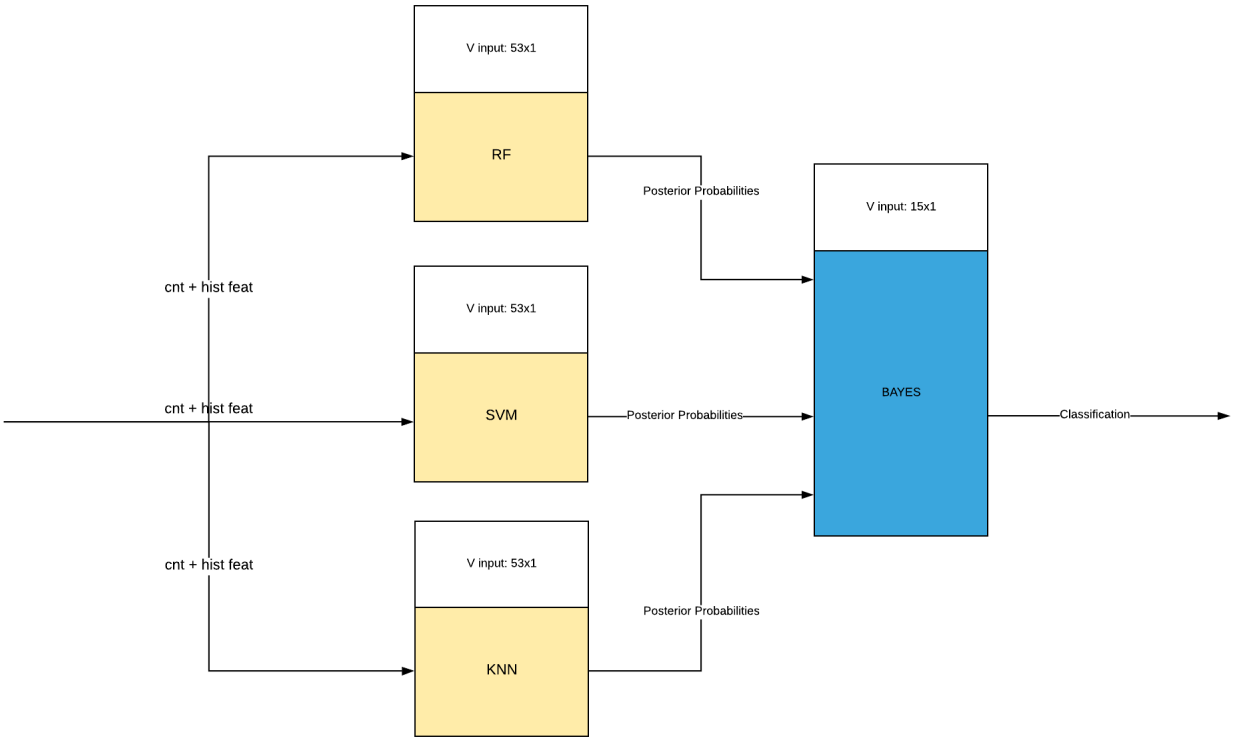
6.2.5. Aggregazione posterior: Naive Bayes

Riassumendo quanto detto per i precedenti classificatori, considerando di maggiore importanza per ciascun classificatore l'accuratezza complessiva, l'accuratezza specifica verso la classe *double*, l'accuratezza peggiore per ciascun classificatore (sempre relativa al caso *enveloped parcel*) indicata con *Acc. Minor*, il tempo di predizione medio e, infine, la varianza dell'accuratezza per una *k-fold cross validation* con *k = 10*, si racchiudono queste informazioni nella seguente tabella:

	Accuracy	Acc. Double	Acc.minor	Class. Time(sec)	K-fold var.
Random Forest	0.73	0.9	0.53	0.0006	0.098
SVM	0.87	0.88	0.5	0.0005	0.006
K-NN	0.83	0.92	0.5	0.0009	0.007

Non potendo accettare una classificazione di una classe *enveloped parcel* che risulterebbe sbagliata la metà delle volte, è necessario cercare di migliorare le prestazioni complessive del sistema. In quest'ottica è utilissimo muoversi sfruttando quanto già prodotto, ovvero i tre classificatori *Random Forest, SVM, K-NN.* Questi, insieme alla classificazione in sé che indica l'appartenenza alla relativa classe risultante, permettono anche di risalire alle *posterior probabilities,* ovvero le probabilità di appartenere a ciascuna classe. Essendo cinque classi, per un totale di tre classificatori, avremo a disposizione un nuovo vettore di *features* composto da $n_{classes} * \ n_{classifiers}$ (5x3 elementi).

Sfruttando questo nuovo vettore per effettuare il *training* di un classificatore *Naive Bayes* [21] posto a valle dei tre già presentati, possiamo rafforzare le singole classificazioni combinando le *posterior* risultanti. Lo schema risultante sarà dunque il seguente:



Grazie a questo procedimento è possibile migliorare notevolmente le prestazioni, come si evince dalla matrice di confusione finale risultante dall'aggregazione. Questo perché i vari classificatori sono affetti da errori di classificazione differenti, complementari, ma sistematici. Le *diversities* sono dunque benefiche e permettono di ottenere per *double* e *parcel* una stabilizzazione ben al di sopra dello *0.9* di accuratezza. Anche la classe *flyer* subisce un ottimo impatto positivo di questa strategia, ma sono *enveloped parcel* e *envelope* a ottenere il miglior beneficio relativo, sebbene sia ancora presente una grande confusione tra le due classi, specialmente in relazione alle altre accuratezze.

[21] Vikramkumar & B, Vijaykumar & Tripathy, Trilochan. Bayes and Naive Bayes Classifier, 2014.

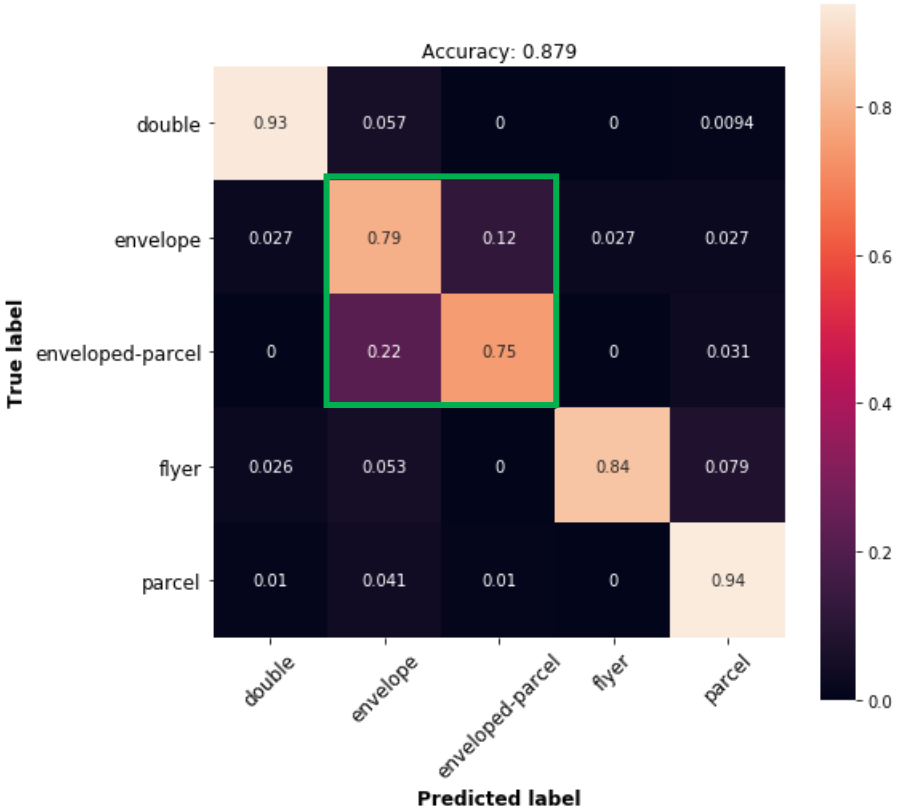


Figura 41-Matrice di confusione Baseline

A livello di tempi di classificazione richiesti per questo ulteriore step, il classificatore *Naive Bayes* richiede un tempo di classificazione medio pari a $AVG_Pred.time_{NB} = 0.0005\text{sec}$ e dunque con uno sforzo temporale iniquo riusciamo a migliorare notevolmente le prestazioni del sistema complessivo.

6.2.6. Considerazioni aggiuntive

La libreria utilizzata (*scikit-learn*) non permette la parallelizzazione dei calcoli e dei classificatori. Dunque, sebbene concettualmente i classificatori siano in parallelo per come vengono trattate le *posterior probabilities*, in un ambito temporale è necessario che si completino in successione le tre classificazioni in modo sequenziale per procedere alla successiva *posterior aggregation*[22]. Con un tempo totale medio di 0,003 secondi, risultiamo

[22] Sigletos, Georgios & Paliouras, Georgios & Spyropoulos, Costantine & Stamatopoulos, Takis. Meta-learning beyond classification, 2004.

comunque abbondantemente soddisfare la richiesta temporale di classificazione di *50ms* e abbiamo quindi molto margine temporale per una strategia più strutturata che permetta migliorare le prestazioni.

6.3.Test Multi frame: media VS aggregazione

Dovendo cercare di comprendere se l'utilizzo di una informazione legata a diversi *frames* sequenziali tra loro possa o meno migliorare la soluzione proposta, è necessario strutturare un nuovo *dataset* che permetta di effettuare dei test specifici. Di seguito si tratta appunto questa procedura, ponendo grande attenzione ai test che permettano di valutarne l'utilità.

6.3.1. Nuova struttura del dataset

Prima di delineare una strategia più raffinata di classificazione, in modo tale da effettuare un migliore *tuning* dei parametri dei vari classificatori, è bene una ulteriore analisi dei dati a disposizione. Come detto inizialmente, il *dataset* è composto da una struttura disomogenea di dati sia a livello di impianti da cui sono stati raccolti, sia a livello di modalità. In particolare, alcuni dati sono presenti a singolo frame, altri in modalità *multiframe.*

Anche a livello *multiframe* è presente una grande varianza del numero di *frames*. Abbiamo a disposizione per ciascun campionamento dai 2 ai 15 *frames*, fino ad alcuni casi dove abbiamo addirittura 30 *frames*.

Nell'ottica di comprendere se, o meno, i dati in versione *multiframe* [23]nascondano anche un'informazione temporale implicita che possa rafforzare la classificazione, è necessario trovare un compromesso tra la rappresentazione di una classe ed il numero di frame da sfruttare. Ad esempio, con *n=10 frames*, abbiamo a disposizione i seguenti valori di campioni:

[23] Canavan, Shaun & Johnson, Benjamin & Reale, Michael & Zhang, Yong & Yin, Lijun & R. Sullins, John. Evaluation of Multi-frame Fusion Based Face Classification, 2010.

- *Parcel*: 103 campioni
- *Envelope*: 55 campioni
- *Enveloped* parcel: 34 campioni
- *Flyer*: 28 campioni
- *Double*: 34 campioni

Risulta evidente come un dataset così sbilanciato e con così poche rappresentazioni per *flyer e double* sia inutile ai fini di una analisi solida. Con *n=3,* invece, si ottiene un lieve miglioramento dei campioni utilizzabili dal *dataset* iniziale. Infatti:

- *Parcel*: 320 campioni
- *Envelope*: 80 campioni
- *Enveloped parcel*: 60 campioni
- *Flyer*: 98 campioni
- *Double*: 100 campioni

Le classi continuano ad essere sottorappresentate, in quanto la maggior parte dei dati a disposizione è stata raccolta con campionamenti a singoli *frames*. Tuttavia, tenendo come numero di riferimento di *frames n=3*, e utilizzando l'impianto demo presente nella sede in azienda, è stato possibile ribilanciare questo dataset *multiframe* con *n=3* fino ad ottenere per ciascuna classe i seguenti valori rappresentativi, utilizzati poi nella trattazione di seguito:

- *Parcel*: 156 campioni
- *Envelope*: 150 campioni
- *Enveloped parcel*: 150 campioni
- *Flyer*: 150 campioni
- *Double*: 150 campioni

Se precedentemente nel *dataset* mono frame il campione utilizzato era quello centrale, adesso, con *frames n=3,* i tre frame risultano essere quello centrale (lo stesso già utilizzato), il

precedente ed il successivo. Facendo riferimento all'immagine Figura 42-Posizione frames, dove viene rappresentato il frame centrale (dunque il secondo frame), il primo e il terzo frame corrisponderanno alle posizioni del collo con il suo centro più vicino alle linee laterali indicate in Figura 42, rossa e blu.

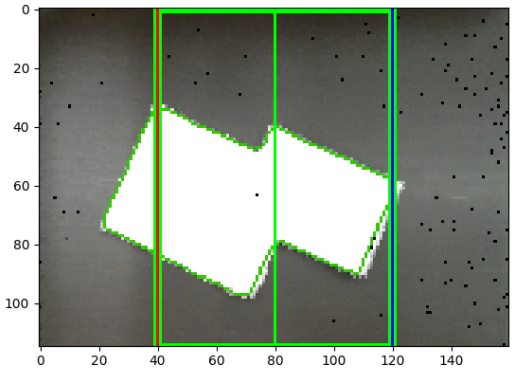
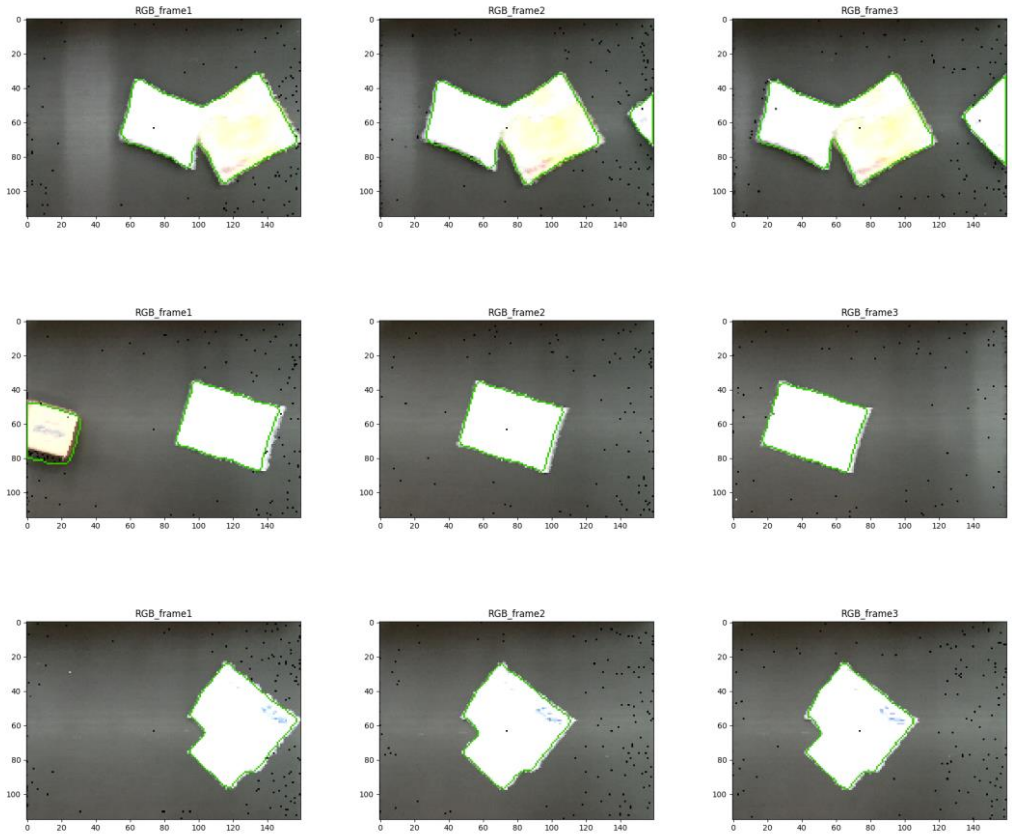


Figura 42-Posizione frames

Come risultato, le sequenza di frame avranno la struttura mostrata nella serie di esempi di Figura 43:



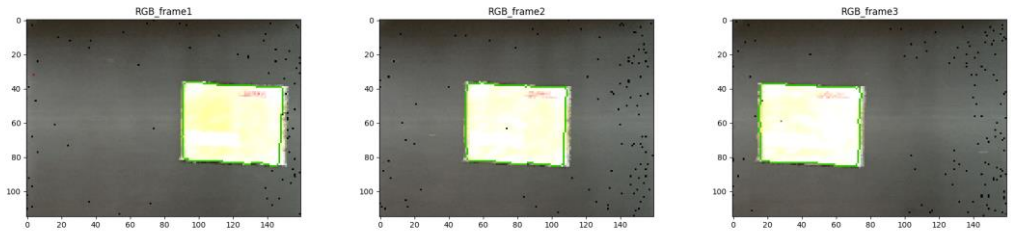


Figura 43-Esempi sequenze di frames

6.3.2. Strategia di analisi

Dato che ogni immagine darà origine ad un nuovo vettore di *features* indipendenti, per ogni tripla di *frames* il tempo necessario per l'estrazione delle *features* sarà circa tre volte superiore al tempo del caso monoframe (le operazioni sono le stesse, ma ripetute tre volte). Vengono quindi performati due test per confrontarli con il caso di classificazione singola. In tutti i casi si utilizzeranno classificatori *Random Forest* per mantenere una comparabilità dei risultati ottenuti.

6.3.2.1.Classificazione singolo frame

Utilizzando le features estratte dai soli *frames* centrali del dataset *multiframe* raccolto, si effettuano *training* e *testing* di un classificatore *random forest.* Le prestazioni di accuratezza andranno poi analizzate confrontandole con i successivi casi, quindi se ne rimanda la trattazione.

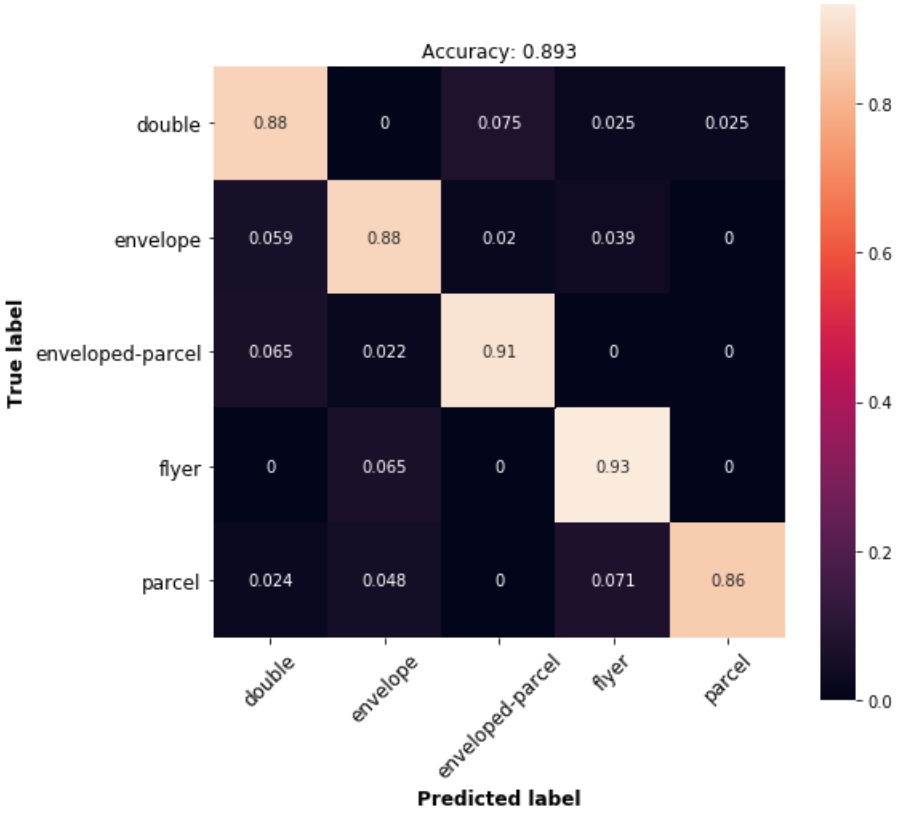


Figura 44-Matrice di confusione test multiframe, singola classificazione

6.3.2.2.Media

Si utilizzano le *features* dei tre *frames* per irrobustire al rumore la *features extraction* e valutare quanto effettivamente il rumore possa affiggere le performance di classificazione.

Tutte le triple di vettori relativi ai set di *frames* vengono tra loro mediate e con queste si effettua il *training* di un singolo classificatore *random forest*, come da schema in Figura 45.

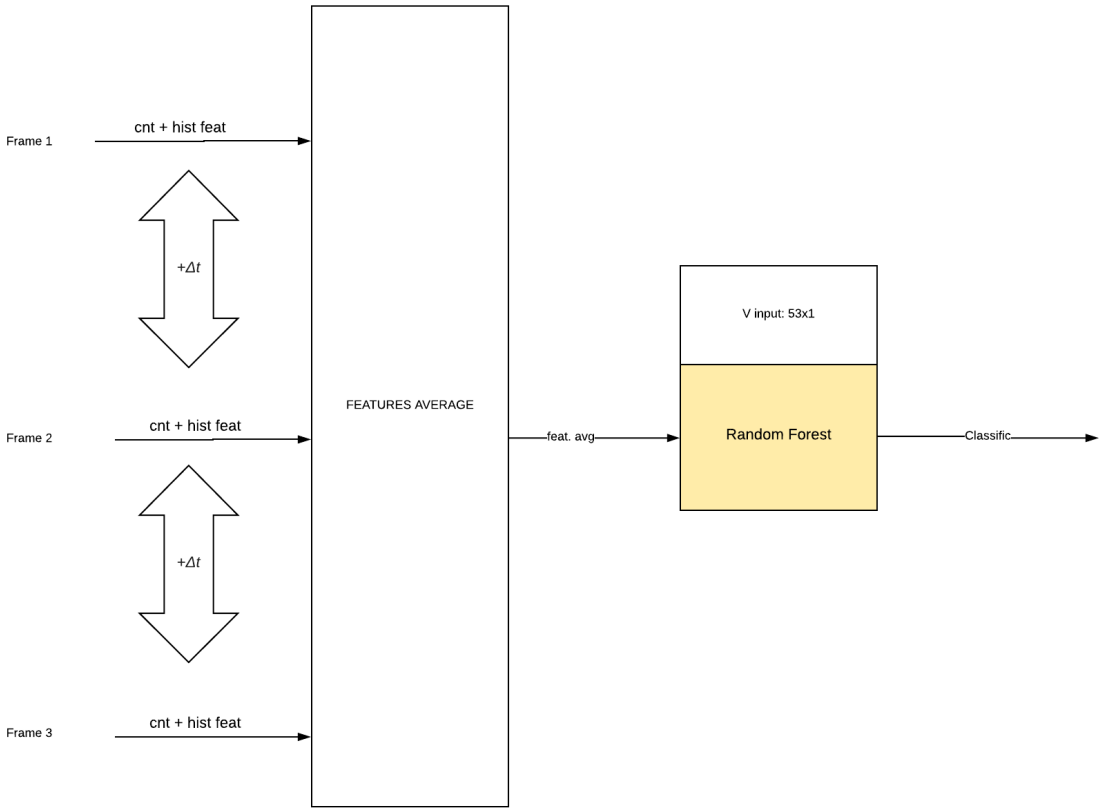


Figura 45-Schema test media

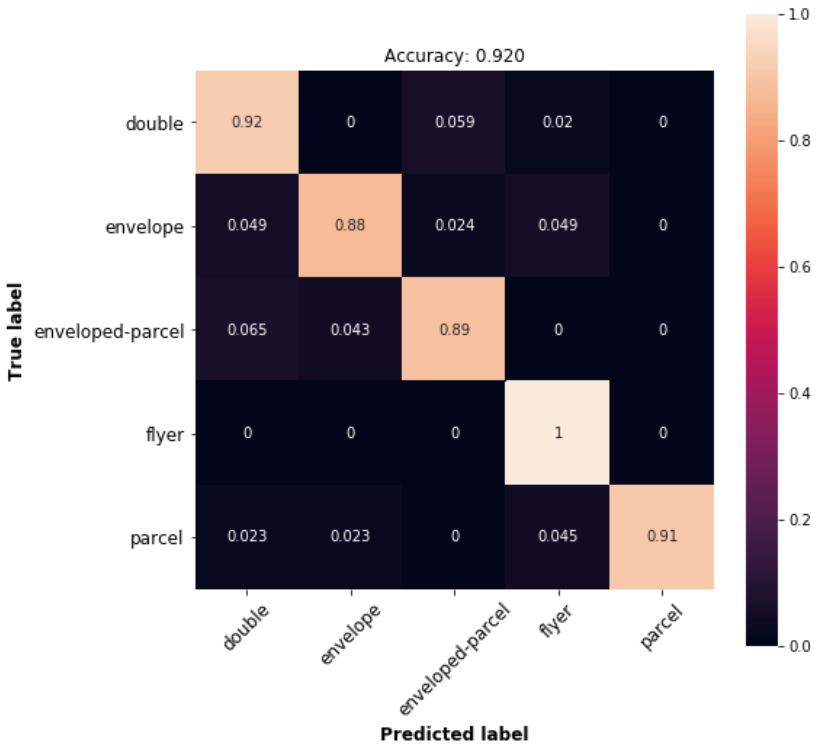


Figura 46-Matrice di confusione test multiframe, media frames

6.3.2.3.Aggregazione posterior probabilities

Si utilizzano i tre *frames* e le relative *features* estratte per effettuare il *training* di tre classificatori *random forest* indipendenti. Di questi si aggregano le *posterior probabilities* per arrivare ad un'unica classificazione che tenga conto dei tre *frames*.

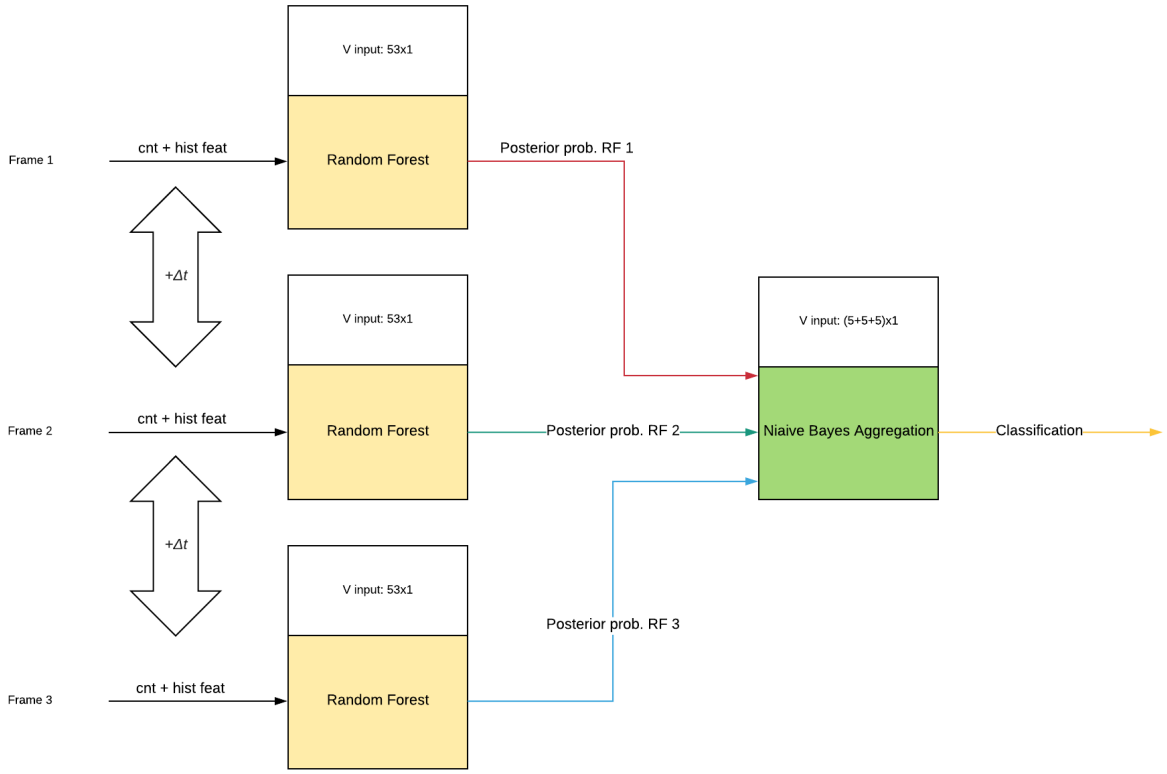


Figura 47-Schema test aggregazione

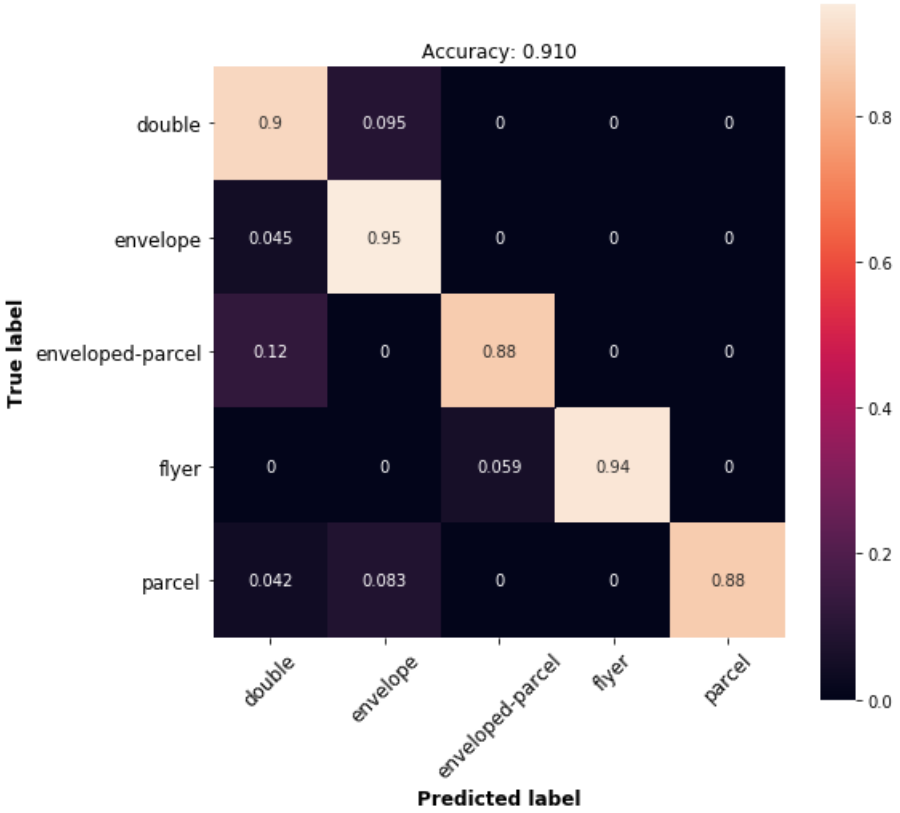


Figura 48-Matrice di confusione test multiframe, aggregazione

6.3.3. Conclusione

Dai risultati appena presentati, si nota come sia attraverso la media delle *features*, sia attraverso l'aggregazione delle *posterior*, non si ottenga un miglioramento significativo rispetto alla classificazione a singolo frame.

Analizzando la media della varianza delle features tra i tre *frames*, riportata nelle tabelle seguenti per ciascuna *feature* estratta, si nota come il rumore non apporti una componente significativa. Per esempio, per l'area del contorno influisce in media per circa 4 *pixel*, su un totale che è nell'ordine di migliaia. Questa bassissima varianza si può ricondurre all'estrazione dei contorni (da cui le *features* vengono poi estratte), che viene notevolmente irrobustita dallo sfruttamento di tre informazioni differenti come *rgb*, *depth* e *ir*.

	cnt_area	perimeter	h	w	rect_area	aspect_ratio	extent	hull_area	solidity	MA	ma	axis_ratio	curvature	areaMinRect
0	3.657049	4.112857	6.922514	7.200324	3.609488	0.007107	0.007064	3.545312	0.007156	2.139059	2.248492	0.044165	0.697719	12.395404

	extentMinima	extentMinimaSecond	Hu0	Hu1	Hu2	Hu3	Hu4	Hu5	Hu6
0	0.007072	0.007214	0.003342	0.004125	3.587241e-10	4.376433e-09	3.490445e-10	3.554537e-10	3.818080e-10

	hist_0_04_0	hist_0_04_1	hist_0_04_2	hist_0_04_3	hist_0_04_4	hist_0_04_5	hist_0_04_6	hist_0_04_7	hist_0_04_8	hist_0_04_9	hist_0_04_10
0	1.172289	1.174607	1.087933	0.92377	0.840583	0.805852	0.677629	0.511095	0.401241	0.292708	0.264283

	hist_04_1_1	hist_04_1_2	hist_04_1_3	hist_04_1_4	hist_04_1_5	hist_04_1_6	hist_04_1_7	hist_04_1_8	hist_04_1_9
0	0.008468	0.003079	0.00077	0.003849	0.003576	0.001333	0.002873	0.004073	0.005182

	hist_0_04_11	hist_0_04_12	hist_0_04_13	hist_0_04_14	hist_0_04_15	hist_0_04_16	hist_0_04_17	hist_0_04_18	hist_0_04_19	hist_04_1_0
0	0.244677	0.182376	0.128726	0.118839	0.094138	0.087315	0.054678	0.049853	0.031816	0.0263

Figura 49-Varianze medie in triple di frames

Si può notare come, in generale nei tre test appena prodotti, le prestazioni del singolo classificatore *random forest* siano superiori rispetto allo stesso classificatore del caso *monoframe* realizzato come baseline. Questo perché il *dataset* raccolto per il caso *multiframe* non presenta una grande varianza di colli su cui lavorare, sebbene abbia il vantaggio di essere bilanciato.

Limitarsi all'uso di questi dati aprirebbe al rischio di un drastico calo di accuratezza una volta di fronte alla variabilità che caratterizza il reale funzionamento degli impianti. Inoltre, il *dataset* iniziale è composto da sottoinsiemi di dati proveniente da più impianti, mentre il presente è solo limitato all'impianto demo.

Pertanto, visto il ridotto apporto positivo di una analisi basata su più frame e nella convinzione che il *dataset* originale, per quanto sbilanciato, permetta di avere una miglior variabilità di dati con cui addestrare e testare i classificatori (la classe *double* è rappresentata dal triplo dei campioni!), si preferisce strutturare una nuova strategia basata su classificazioni a singolo frame. Così facendo, inoltre, si permette di evitare un tripla estrazione di *features*, che rallenterebbe (a questo punto quasi inutilmente) l'intero processo.

Se il miglioramento delle prestazioni non si può cercare in nuove informazioni di carattere temporale racchiuse nella sequenzialità di più frame, un irrobustimento delle accuratezze si può cercare di ottenere attraverso una migliore selezione delle *features* e del *tuning* dei parametri caratteristici dei classificatori.

6.4. Strategia 2: classificazioni successive con classificatori specializzati

Per strutturare una classificazione più efficace di quella proposta come *baseline* è necessario uno studio più approfondito delle *features.* Abbiamo già discusso quanto il numero di classi tra cui distinguere e il numero di *features* che si utilizza possa far variare il tempo impiegato per la classificazione. Dunque, se spezzassimo la classificazione in più fasi che utilizzano un numero di *features* ridotto, o un numero di classi minore, è auspicabile raggiungere una classificazione più affidabile e che richieda intervalli minori di tempo.

Le *features*, come già esposto approfonditamente, possono essere divise in due sotto vettori:

1. *Features relative ai contorni*: vettore contenente 23 scalari, relativi agli aspetti morfologici del collo in analisi.
2. *Features riguardanti gli istogrammi*: vettore contenente 30 scalari, riguardanti l'estensione delle zone appartenenti ad una certa fascia di altezza.

L'utilizzo di questi vettori in modo separato permette la creazione di classificatori in grado di compiere distintamente decisioni riguardanti la morfologia e gli istogrammi (dunque, le altezze). Inoltre, utilizzando classi *ad hoc,* la taratura dei parametri dei classificatori può portare ad una classificazione migliore ed in minor tempo tra le nuove classi.

Nell'ottica di cambiare lo spazio delle classi, è necessario strutturare delle classi "maggiori", i cui elementi appartenenti siano accomunati da elementi in comune. La struttura delle classi è la seguente:

- *Double*: Data l'importanza di questa classe (fulcro centrale del problema) si lascia costituita esattamente come quella iniziale.
- *Non* double: Classe che raggruppa il coniugato dei *Double.* Raggruppa dunque:
    - *Flyer*
    - Parcel
    - Enveloped parcel

- Envelope

La classe *Non Double* presenta internamente due macroclassi, che raggruppano le sottoclassi appena presentate. Dunque, *Non Double* racchiude:

- *Rigid:* è composta dai *Flyer* e dai *Parcel*. Queste classi sono entrambe composte da colli morfologicamente ben definiti e, appunto, rigidi. Le superfici sono ben chiare ed uniformi e dunque negli istogrammi ci si aspetta un unico picco.
- *Enveloped*: è composta dagli *Envelope* e dagli *Enveloped parcel*. Morfologicamente i corpi, non essendo rigidi, hanno forme molto più difformi e le altezze sono distribuite su più intervalli.

Saranno dunque suddivise come riportato in Figura 50:

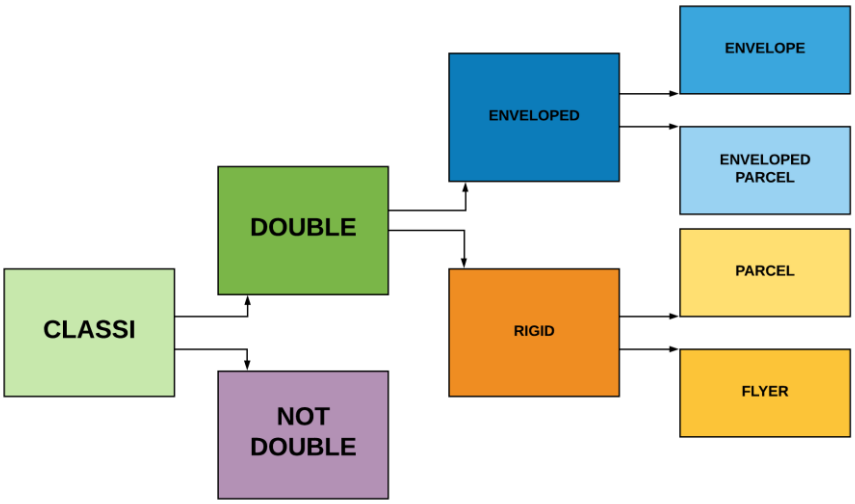


Figura 50-Nuova divisione in macroclassi

Seguendo la struttura delle classi così decisa, le fasi successive di classificazione vengono naturalmente sequenziali permettendo ispezioni successive che riescano a discernere sempre più nel dettaglio, indicate in Figura 51:

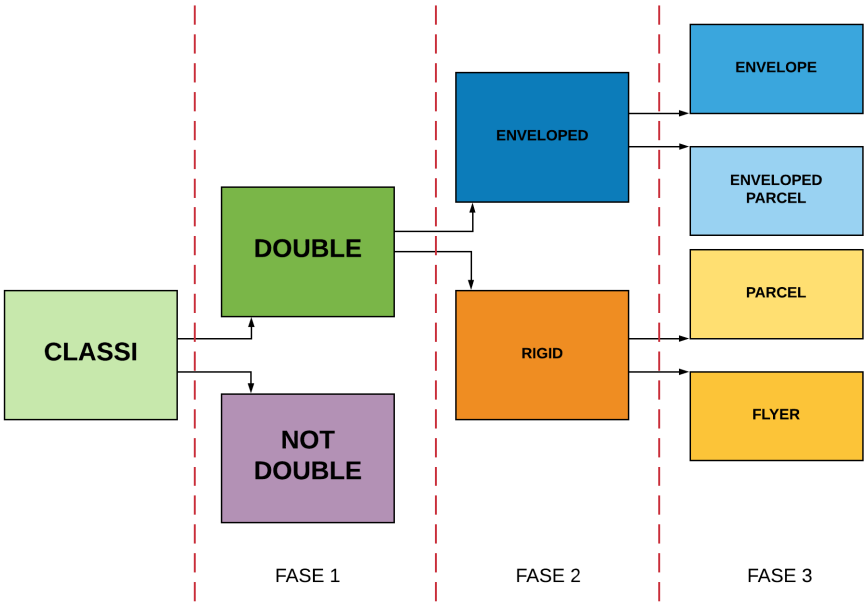


Figura 51-Fasi classificazione sequenziale

I classificatori strutturati per le singole fasi, le prestazioni singole e dell'intera strategia verranno presentati fase per fase. Il *training* viene effettuato sul *dataset* a singolo frame iniziale per poter far sì che sia composto dal numero più alto di campioni possibile. Invece, il *testing* è stato performato sul *dataset multiframe*, selezionando solo il frame centrale. Questo perché il nuovo *dataset* presenta la maggior parte di campioni nuovi per ciascuna classe e queste sono tutte bilanciate. In questo modo le prestazioni si possono ritenere già più vicine alla realtà rispetto al caso della *baseline.* Spesso infatti i classificatori, una volta usciti dalla fase *testing* e *training*, subiscono un importante peggioramento delle prestazioni in fase di funzionamento reale.

Verranno presentate primariamente le prestazioni in termini di accuratezza e, infine, le qualità temporali delle varie fasi.

6.4.1. Prima fase: double detection

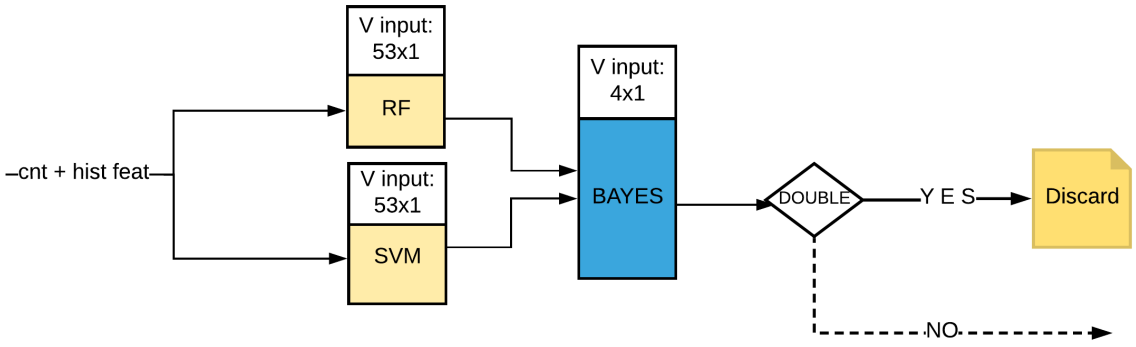


Figura 52-Schema sequenziale, prima fase

In ottica di *double-detection*, l'obiettivo primario è il riuscire a identificare nel minor tempo possibile e in modo robusto il verificarsi di tale situazione. Una volta identificata, si procede con lo scartare i pacchi e inviarli verso la nuova fase di singolarizzazione. Dunque, la classificazione *Double vs. Non-Double* è necessario sia la prima tra le classificazioni successive e se ne identifica la strutturare come da Figura 52. Qualora riscontrata la positività di *double*, nessuna ulteriore fase sarà necessaria.

Data l'importanza della classe e la necessità di rapidità di decisione, la struttura di classificazione resta simile alla *baseline.* Un classificatore *random forest* posto in parallelo ad un classificatore *svm*, con uno stadio di aggregazione attraverso *naive bayes*, permette di ottenere ottime prestazioni. La presenza di due sole classi di confronto permette di ottenere una rapidità di classificazione apprezzabile.

Rispetto le *features* non è risultato apprezzabile fare alcuna selezione in questa fase, il *double* infatti si distingue dai *non-double* sia dal punto di vista morfologico (primo sotto vettore), sia dal punto di vista degli istogrammi (secondo sotto vettore). Due pacchi di altezze ben distinte, ma a contatto, daranno vita a due picchi ben definiti e distanti ed è opportuno dunque avere tutte queste informazioni subito a disposizione dei classificatori per sfruttarle in una unica fase e senza perdere tempo.

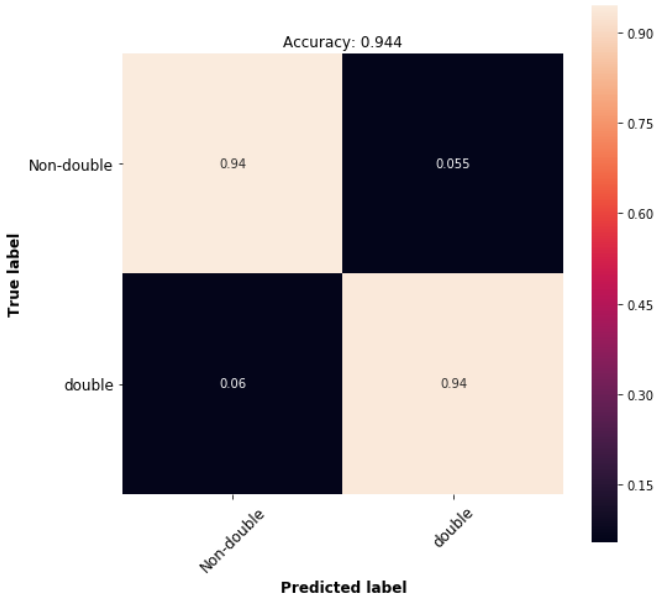


Figura 53-Matrice di confusione classificazione sequenziale, prima fase

La mancanza di un classificatore *K-NN* è dovuta all'obiettivo di massimizzare il più possibile la velocità di esecuzione. Come già esposto, il dover calcolare circa 1700 distanze del nuovo campione dai punti di *training* sarebbe stato giustificabile solo se avesse aumentato le prestazioni in modo significativo.

6.4.2. Seconda fase: rigid vs. envelope

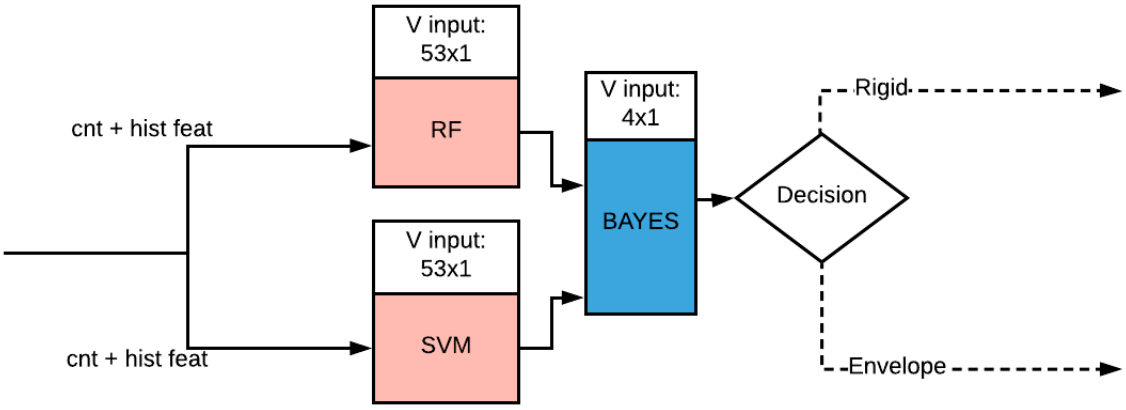


Figura 54-Schema sequenziale, seconda fase

Nel caso in cui la *fase 1* determinasse l'assenza del caso *double*, è necessario procedere in una ulteriore analisi. La distinzione da effettuare ora si compone della scelta tra *rigid* ed

envelope, la riduzione delle classi in macroclassi permette una taratura dei parametri ottimale e, come prima, una rapidità di classificazione non indifferente.

Anche in questo caso, trattandosi di una "scrematura", si ripropone la medesima struttura (indicata in Figura 54) con l'intero vettore delle *features* e una fase di aggregazione che permetta di ridurre il più possibile il rischio di misclassificazione.

Si riporta la matrice di confusione, comprendente anche la prima fase, in modo da comprendere subito gli effetti totali del sistema:

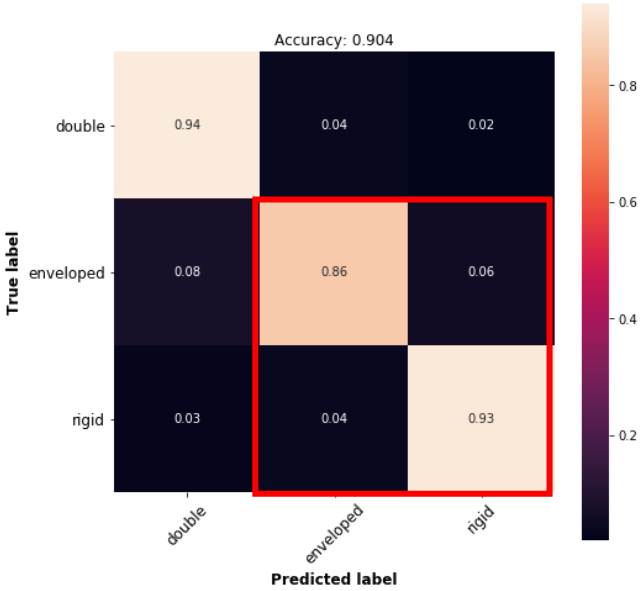


Figura 55-Matrice di confusione classificazione sequenziale, seconda fase

Si noti come la matrice di confusione racchiuda le prestazioni anche della *fase 1*. Infatti, lo 0.93 e 0.86 tengono conto delle classificazioni errate contate come *non double* che porteranno a una penalizzazione dei classificatori seguenti.

6.4.3. Terza fase: evelope vs. env. Parcel e flyer vs. parcel

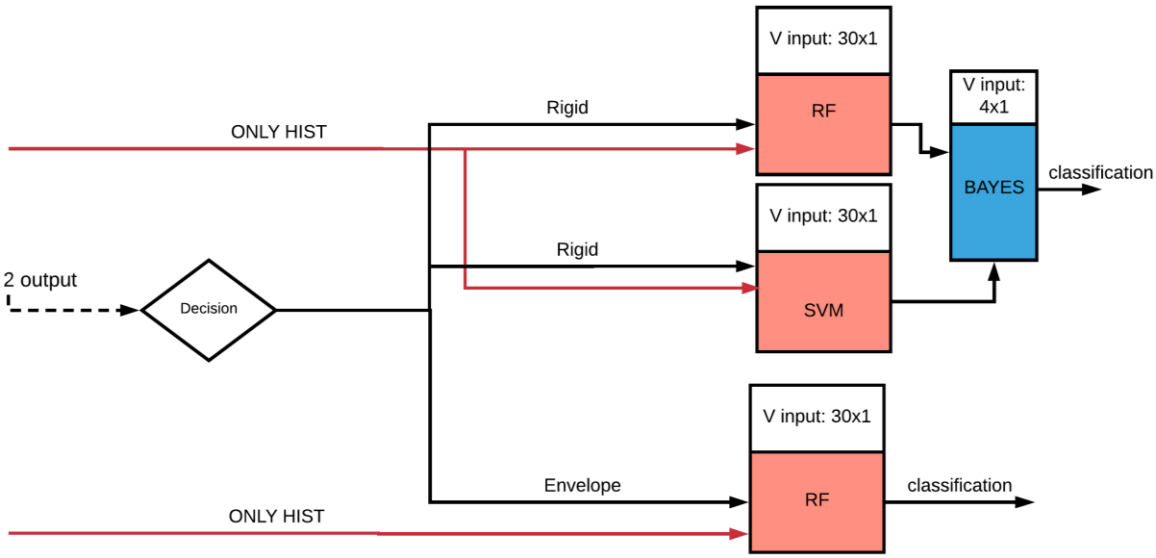


Figura 56-Schema sequenziale, terza fase

Infine, per una decisione più raffinata, è necessario comprendere a quale delle sottoclassi appartenga il collo in analisi. Per questa analisi è utile sfruttare solamente il vettore relativo agli istogrammi.

Seguendo lo schema di Figura 56, ci sono due possibili casi, dipendenti dalla precedente classificazione:

1. Output 2 →*Rigid*: si attua una simile tecnica con aggregazione delle posterior per ridurre la confusione di classificazione.
2. Output 2 →Envelope: in questo caso è sufficiente utilizzare un semplice singolo *Random Forest* per riuscire ad ottenere ottime prestazioni. Utilizzare le *posterior probabilities* non porta nessuna miglioria in questa fase.

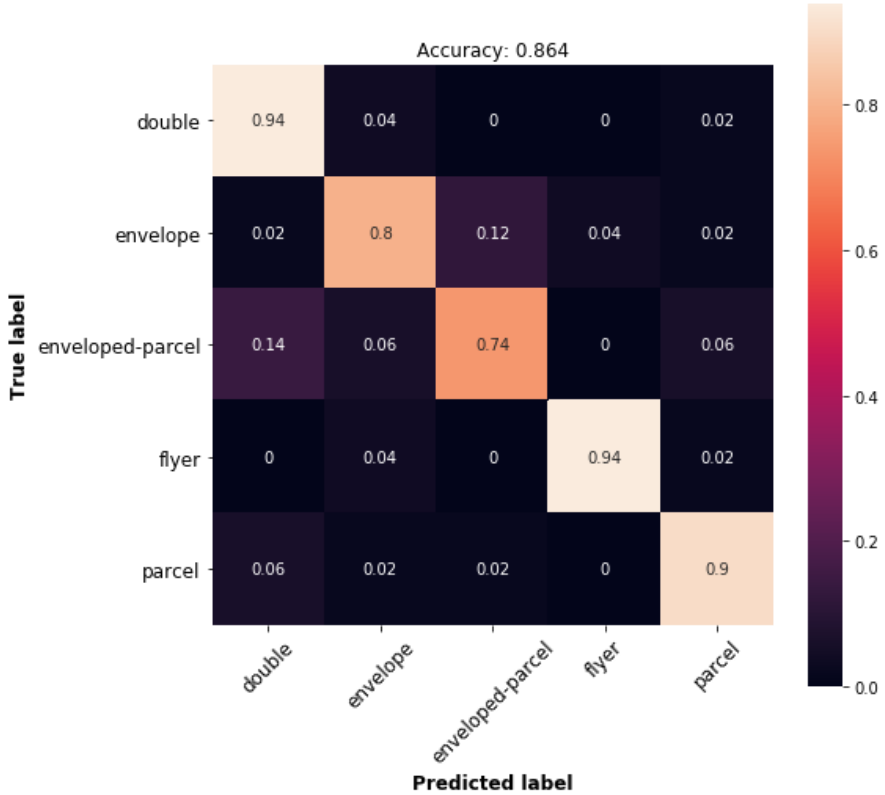


Figura 57-Matrice di confusione classificazione sequenziale, terza fase

6.4.4. Confronto

La strategia vista nello schema in Figura 58, che ne descrive la sua totalità è rappresentata di seguito, dove si descrive il tipo dei classificatori, l'input e la sua dimensione e i possibili *output*:

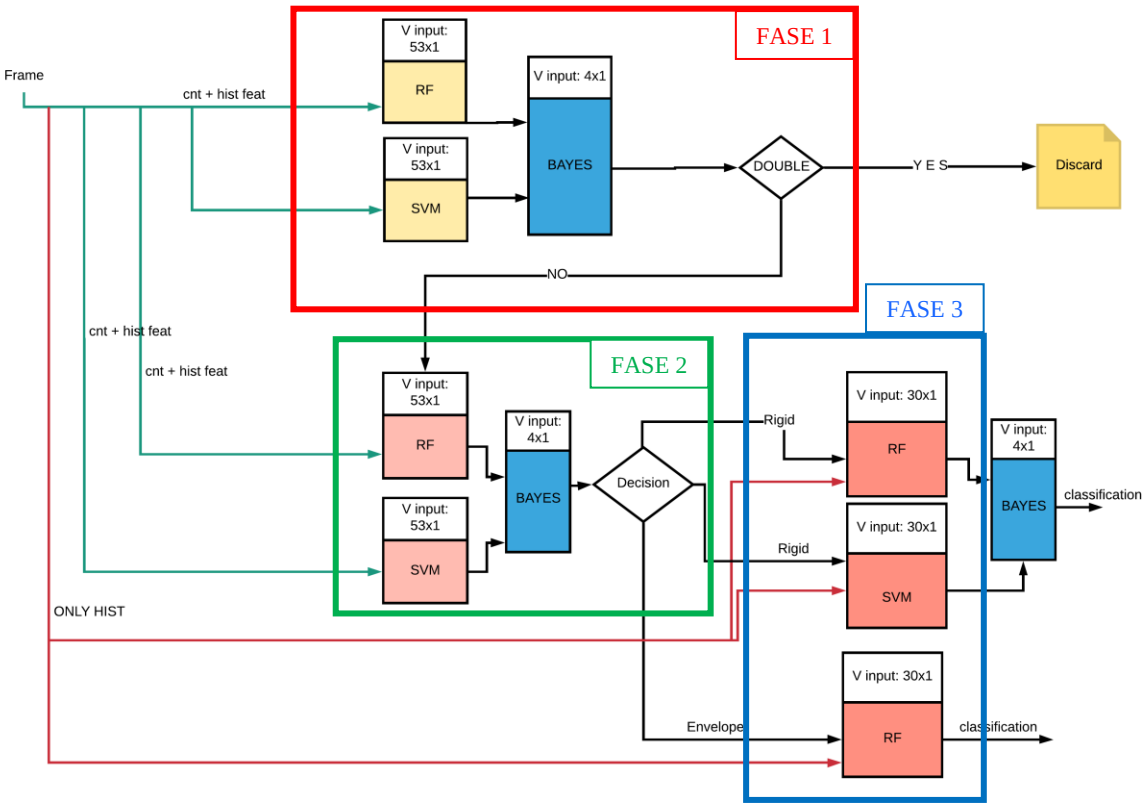


Figura 58-Schema sequenziale complessivo

	Class.Time(Sec)
Fase 1	0,001
Fase 2	0.002
Fase 3	0.001

Rispetto alla *baseline* proposta, si può notare come in generale vi sia un netto miglioramento. *Double*, *Parcel e Flyer* presentano una accuratezza superiore allo 0.9. Per il caso *flyer* si riscontra un aumento di ben un decimo rispetto alla *baseline*.

Sebbene in termini di accuratezza rispetto alla *baseline* per *double* non ci si discosti in modo significativo, è utile sottolineare come in termini di velocità di predizione si riesca a migliorare notevolmente le prestazioni. Il caso *Double* viene identificato robustamente in un tempo poco superiore ad 1/3 del tempo di classificazione iniziale. Questo permette di dare, in tempi brevissimi, un segnale alla parte successiva dell'impianto nel caso di *double* e predisporne lo

scarto, oppure più tempo a disposizione per le analisi successive maggiormente approfondite nel caso di *non-double.*

Non vi è molto vantaggio, invece, per quanto riguarda le classi *enveloped parcel* ed *envelope:* la grande confusione che si verifica tra le due classi è intrinseca nelle *features* estratte, che non permettono ai classificatori di distinguere a dovere tra i casi.

Tuttavia, si ritiene che ai fini del problema il risultato raggiunto per il *double* sia soddisfacente e, in ottica di manipolazioni da effettuare nelle fasi successive, la distinzione E*nvelope vs. Rigid* sia un ottimo compromesso.

Si può dunque ritenere soddisfacente la strategia *features-based* strutturata in quanto compie una *double detection* agile e robusta. Per le altre classi, di secondaria importanza, si è comunque garantita una classificazione di massima attendibile per i *flyer e parcel,* mentre per *envelope ed enveloped parcel* si riesce a riconoscere abbastanza attendibilmente la famiglia di appartenenza (ovvero le buste), ma non è stato possibile riconoscere con una accuratezza pari alle altre classi tra le sottofamiglie della classe.

7. Limitazioni in produzione

Una volta ritenuti soddisfacenti le prestazioni dei classificatori e delle strategie strutturate si è ritenuto opportuno andare oltre le prestazioni effettuate *offline*, di cui trattato fino ad ora, effettuando un tentativo di confronto simil *online* da immagini prese da un nuovo impianto e non utilizzate precedentemente né in fase di *training,* né in fase di *testing.*

Durante questa operazione si è potuto notare come in alcune situazioni, particolari ma comunque frequenti, l'estrazione dei contorni presenti un problema che porta a vanificare l'intera successiva fase di classificazione.

Il nastro trasportatore viene fissato alle estremità attraverso una operazione che nel punto di giuntura lo rende più soggetto all'usura e all'abrasione. Questo causa una perdita di colore di una sezione perpendicolare al senso di scorrimento su cui sono disposti i pacchi.

Cambiare le soglie dei contorni per filtrare questo fenomeno (che intacca l'integrità della *depth* e della *rgb)* porta a perdere una biunivocità tra forma dei contorni e la morfologia del collo in analisi e dunque non rende applicabile in questa situazione la strategia fin qui identificata e strutturata.

Figura 59-Problema usura del nastro

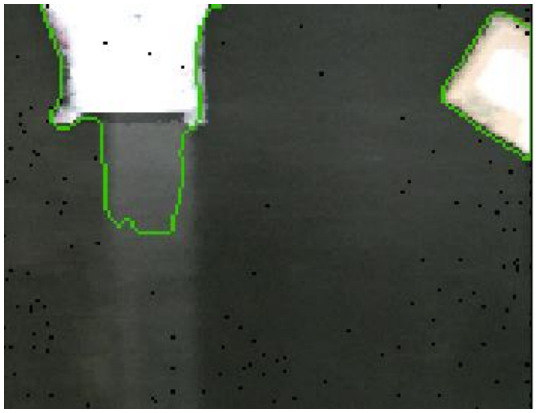


Figura 60-Risultato usura del nastro in analisi

8. Conclusioni e sviluppi futuri

L'intero percorso descritto ha permesso di sviluppare una metodologia di progetto che permetta di ripetere tutte le fasi trattate in modo agevole ed aumentare i dati a disposizione per garantire sempre più generalità ai classificatori. Attraverso il programma di etichettamento sviluppato, i nuovi dati possono essere etichettati e le *features* possono essere estratte facilmente per essere aggiunte al dataset già sviluppato. Inoltre, le *features* possono essere ampliate implementando nuove misure direttamente nell'applicativo per permettere un veloce riesame dell'intero *dataset* e delle successive prestazioni dei classificatori.

La strategia di classificazione ed i classificatori realizzati risultano pronti per l'esportazione e per la messa *online* sul macchinario. Soddisfano egregiamente i requisiti temporali richiesti e rispetto alle classi *double, flyer* e *parcel* si riescono a ottenere prestazioni discretamente buone. Le prestazioni calano drasticamente nell'individuazione e nella distinzione di *enveloped parcel* ed *envelope*, in cui un approccio *feature based* non riesce a garantire un fattore discriminante sufficientemente valido per ritenerlo attuabile in produzione.

Per cercare di superare l'inaccuratezza appena descritta e, soprattutto, le difficoltà generate dall'usura del nastro, che farebbero degenerare le performance dei i classificatori su impianti che presentano questa complicazione, è necessario andare oltre la metodologia proposta.

Il programma per la creazione del *dataset* crea e salva di default la maschera di ogni singola immagine etichettata. Questo permette di sfruttare il grande sforzo realizzato verso la generazione di un *dataset* robusto e sfruttare quest'ultimo attraverso tecniche di *deep learning* non *features-based*, come *reti convoluzionali* per la segmentazione delle immagini e, di seguito, per la loro classificazione. Il valore aggiunto che distinguerebbe le tecniche convoluzionali risiede nello sfruttare le informazioni spaziali che le immagini racchiudono intrinsecamente nei *pixel* e nel riconoscimento dei loro *pattern*. Se per l'estrazione dei contorni le triple di immagini

(*rgb, depth, ir)* vengono sfruttate per individuare robustamente i contorni, nel caso di tecniche *convoluzionali* queste tre informazioni indipendenti possono aiutare a fornire una classificazione che tenga conto di una mole di informazioni decisamente maggiore di quella prodotta con l'approccio *feature based.* Si potrebbero analizzare separatamente *rgb, depth, ir* per ricercare pattern significativi in ciascuna di esse.

Aumentando la mole di informazioni da processare e, con essa, la complessità dei calcoli da essere effettuati dal calcolatore, aumenteranno anche le caratteristiche dell'hardware da utilizzare in modo tale da rispettare i vincoli temporali di classificazione e parallelamente aumenteranno anche i costi. Per questo motivo, prima di affrontare il problema in ottica *deep learning*, è stato necessario effettuare lo step *feature-based*, in quanto se avesse prodotto prestazioni ottime di classificazioni, avrebbe permesso un investimento hardware minore. Tuttavia, per concludere, in ottica di sola *double detection*, si ritiene che le prestazioni raggiunte permettano di implementare le strategie negli impianti come ulteriore controllo, aumentando quanto meno la robustezza e la versatilità, nell'attesa di uno sviluppo migliore con le strategie *deep learning* ora brevemente presentate.

9. Bibliografia

1. Tan, Chee Fai & Wahidin, L.S. & Khalil, Siti Nurhaida & Tamaldin, Noreffendy & Hu, Jun & Rauterberg, Matthias. (2016). The application of expert system: A review of research and applications.
2. L.Culjak, D. Abram, T. Pribanic, H. Dzapo and M. Cifrek, "A brief introduction to OpenCV," *2012 Proceedings of the 35th International Convention MIPRO*, Opatija, 2012.
3. Papari, Giuseppe & Petkov, Nicolai. Edge and line-oriented contour detection: State of the art, 2011.
4. Mahamkali, Naveenkumar & Vadivel. OpenCV for Computer Vision Applications, 2015.
5. S. Cho, R.M. Haralick, and S. Yi, Improvement of Kittler and Illingworth's Minimum Error Thresholding," Pattern Recognition, 1989.
6. Bora, Dibya. Importance of Image Enhancement Techniques in Color Image Segmentation: A Comprehensive and Comparative Study. Indian Journal of Scientific Research, 2017.
7. Cruz, Leandro & Lucio, Djalma & Velho, Luiz. Kinect and RGBD Images: Challenges and Applications, 2012.
8. A, Dziech & Besbas, Wajdi & Nabout, Adnan & N, EldinFast Algorithm for Closed Contour Extraction, 1997.
9. Somasundaram, Akila & Reddy, U. Srinivasulu. Data Imbalance: Effects and Solutions for Classification of Large and Highly Imbalanced Data, 2016.
10. Hanbay, Kazım & Golgiyaz, Sedat & Fatih Talu, Muhammed. Real time fabric defect detection system on Opencv platforms,2017.

11. Huang, Zhihu & Leng, J. Analysis of Hu's moment invariants on image scaling and rotation, 2010.
12. Zhong, Henry & Kanhere, Salil & Chou, Chun. QuickFind: Fast and contact-free object detection using a depth sensor, 2016.
13. F. Tombari, S. Salti, L. Stefano, Unique Signatures of Histograms for Local Surface Description, 2010.
14. Ye, Zhang & Hongsong, Qu. Rotation invariant feature lines transform for image matching, 2014.
15. Guo, Shenyang. Maximum Likelihood Estimator. Chinese Sociological Review, 2013.
16. Ali, K. On the link between error correlation and error reduction in decision tree ensembles, 1995.
17. Pretorius, Arnu & Bierman, Surette & J. Steel, Sarel. A meta-analysis of research in random forests for classification, 2016.
18. Tumer, Kagan and Joydeep Ghosh. “Error Correlation and Error Reduction in Ensemble Classifiers.” Connect, 1996.
19. Chu, Wei & Sathiya Keerthi, S & Ong, Chong. A General Formulation For Support Vector Machines, 2002.
20. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician, 1996.
21. Vikramkumar & B, Vijaykumar & Tripathy, Trilochan. Bayes and Naive Bayes Classifier, 2014.
22. Sigletos, Georgios & Paliouras, Georgios & Spyropoulos, Costantine & Stamatopoulos, Takis. Meta-learning beyond classification, 2004.
23. Canavan, Shaun & Johnson, Benjamin & Reale, Michael & Zhang, Yong & Yin, Lijun & R. Sullins, John. Evaluation of Multi-frame Fusion Based Face Classification, 2010.