**Documento tecnico per l’Applicazione di Computer Vision di Systematik s.r.l.**

Systematik s.r.l. anno 2022

**Indice**

* **Introduzione**
* **Descrizione dell’applicazione**
* **Workflow dell’applicazione**
* **Visualizzazione degli errori di produzione**
* **Object detection**
* **Generare un processo in background**
* **Networking**
* **Esempi di comandi**
* **Hardware**
* **Software**
* **Problematiche tecniche**
* **Risultati**

**Introduzione**

Il Sistema di Computer Vision di Systematik s.r.l. è un sistema composto da hardware e software con l’obbiettivo di migliorare il lavoro giornaliero ddi rilevare errori durante il processo industriale in maniera automatica, andando così ad ottimizzare il processo stesso. Attraverso l’introduzione di una o più videocamere all’ interno del processo industriale, è possibile connette queste videocamere ad un computer apposito, il quale visualizza le immagini, ed attraverso un sistema di algoritmi informatici, è in grado di rilevare errori nel prodotto con un’efficienza pari ad un lavoratore medio.

**Descrizione dell’applicazione**

Questa Applicazione di Computer Vision permette di:

1. Comunicare attraverso socket con il microcontrollore PLC connesso ai macchinari industriali.
2. Quando viene attivato un evento da remoto tramite PLC, permette di scattare una foto ad una o più videocamere connesse in locale.
3. Processare la foto per verificarne la presenza di errori nel prodotto finito.
4. Inviare il risultato dell’elaborazione ad un’interfaccia ed alla PLC connessa al macchinario.

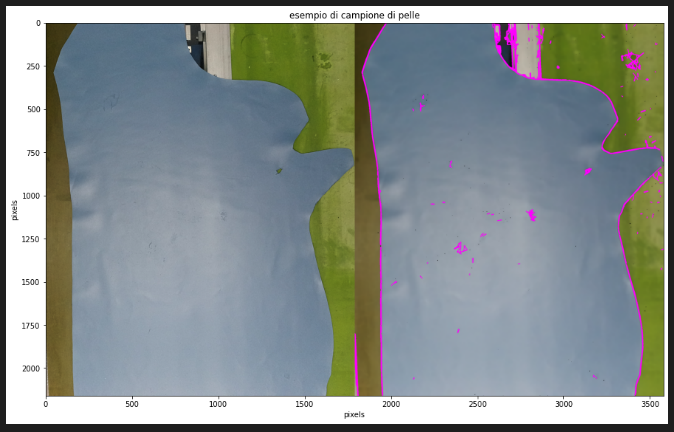
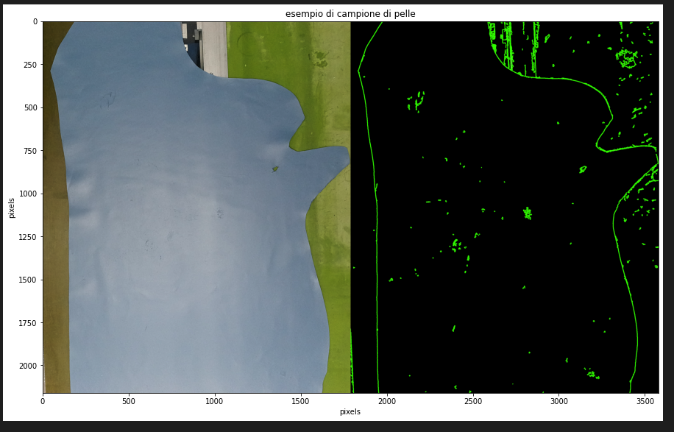
Quest’app non è niente che di meno di un processo sul server lanciato in background. Questo processo deve mantenere attiva la connessione con la videocamera e con la PLC.

**Workflow dell’applicazione**

L’applicazione è un file scritto in Python denominato *app.py* un processo. Questo file contiene l’intero codice sorgente sistema di computer vision.

**Visualizzazione degli errori di produzione**

Fra gli scopi dell’applicazione vi è quello di individuare gli errori nel pezzo di fabbricazione. Questo processo porta a risultati simili ai sottostanti. In entrambe le figure si vede il prodotto di fabbricazione a sinistra ed a destra la stessa immagine rielaborata al fine di evidenziare gli errori di produzione.

****

Questo sistema avviene attraverso un flusso di funzioni specifiche per rielaborare l’immagine. Molte di queste funzioni appartengono alla libreria open source opencv.

Flusso di lavoro per generare immagini con contorni e imperfezioni disegnate:

1. Caricare l’immagine
2. Trasformare in scala di grigi
3. Sfocare l’immagine
4. Generare una threshold di intensità
5. Utilizzare la funzione *Canny* per il riconoscimento di contorni
6. Utilizzare le funzioni *FindContours* e *DrawContours* per generare l’immagine con i contorni trovati
7. Visualizzare l’immagine

**Object detection**

****

**Generare un processo in background**

Come trasformare il file *app.py* in un processo in background nel server?

Questo file viene trasformato in un processo in background (ovvero un processo demone) attraverso il software da linea di comando open source chiamato **nohup***.*

Utilizzando questo software è possibile *far girare* continuamente l’applicazione su di un server, senza doversi preoccupare di lasciare aperto il terminale di comunicazione con il server. Attraverso nohup in combinazione con Python è possibile non dover compilare il file prima di restituirlo al server, aumentando così la velocità di sviluppo e manutenzione. Il programma da linea di comando nohupuna volta eseguito, restituisce un valore numerico che equivale al Process ID (PID), ovvero all’id identificativo del processo. Questo PID serve in casi di debugging in quanto permette di arrestare il processo attraverso il programma da linea di comando **kill <PID>***.*

**Networking & Sockets**

Un socket, in informatica, è un'astrazione software, gestita dal sistema operativo, che rappresenta un canale di comunicazione di rete tra un processo e una risorsa. Per un programmatore, un socket è un particolare oggetto sul quale leggere e scrivere i dati da trasmettere o ricevere.

Noi useremo i socket per creare connessioni fra il microcontrollore PLC ed il nostro server per la computer vision. I socket vengono generati in rete locale e comunicano attraverso cavo ethernet. Per connettersi ad un macchinario attraverso socket è neccessario il protocollo di comunicazione, l’indirizzo ip e la porta della macchina.

IP : porta =====socket TCP IP ====== IP : porta

**Esempi di comandi**

**Generare un demone dell’applicazione con nohup**

## [input] nohup python3 app.py %1> logs.out %2 > logs.err

[output] 188929

**Uccidere il processo demone dell’applicazione**

**[input] kill -f 188929**

Questo comando permette di generare un processo in background e generare due file prodotti dal processo. Uno chiamato *logs.out* dove vi sono gli standard outputs del programma (esempio *print)*. L’altro file è chiamato *logs.err* e contiene gli standard errors del processo. Così facendo è possibile visionare i file in maniera divisa fra errori ed output. Così facendo inoltre è possibile visionare in un qualsiasi momento questi file anche quando il processo è attivo. Per motivi di sicurezza sarebbe meglio scaricarli dal server prima di leggerli.

**Hardware**

Le specifiche hardware per questo sistema sono molto flessibili, in quanto è possibile utilizzare diversi microcontrollori PLC, diversi server computer e diverse videocamere per lo stesso sistema. Il sistema è pensato per essere configurato su di un tipo di hardware, ovvero specifiche videocamere, PLC e tipi di server computer, ma cambiando i parametri per l’acquisizione e l’elaborazione delle immagini è possibile un qualsiasi sistema hardware si voglia.

In specifico riporto i nomi tecnici della nostra strumentazione per l’intero sistema di Computer Vision.

* Videocamere
  1. Videocamera stereo a 16 mega pixel
* Computers
  1. Latte Panda

**Software**

* Linguaggi di programmazione
  1. Python 3.7
  2. PLC programmazione a blocchi
* Librerie software
  1. Opencv
  2. Tensorflow
  3. Sockets

**Problemi tecnici**

**Risultati - Indice**

1. Ritaglio dello sfondo
2. Individuazione degli errori
3. Accuratezza nell’individuazione di errori
4. riflesso della luce provoca cambiamenti di colore sulla pelle