# Image processing

Repository del tema d'anno di image processing

## Quick start

In questa breve sezione si descrive la demo rapida del progetto. Riferisi alle sezioni successive per ulteriori informazioni.

1. Installare python3.8 e pip (dipende dal sistema operativo)
2. Installare le dipendenze del progetto: Linux e Mac OS:

```
$: pip install -r requirements.txt
```

Su Windows

```
$: pip install -r requirements-win.txt
```

3. Eseguire il setup con le immagini di test per ricavare `homography.bin`:

```
$: python main.py setup --checkerboard-cols 9 --checkerboard-rows 6 --
checkerboard-size 25 --directory "test/"
```

4. Eseguire la demo con le immagini di test:

```
$: python main.py detect --model 'checkpoints/pretrained.pt --directory 'test/' --
pixel-unit 0.75
```

5. Per chiudere le finestre per scorrere le immagini (si può uscire premendo `q`).

## Guida all'uso

## Prerequisiti

Per poter utilizzare il sistema è necessario disporre di una macchina con Python 3.8 installato nel PATH di sistema e pip per installare le dipendenze.

## Installazione dipendenze

Le dipendenze sono listate in `requirements.txt` (`requirements-win.txt` per i sistemi operativi windows). Si può avviare l'installazione con:

```
$: pip install -r requirements.txt
```

(sostituire requirements.txt con la versione windows se necessario)

## Avvio del progetto

Una volta installate le librerie, il programma può essere avviato eseguendo `main.py`. Per ottenere la lista degli argomenti a linea di comando con cui può essere avviato è sufficente avviare lo script con il flag `-h`:

```
$: python main.py -h
```

## Walkthrough demo

Per poter avviare una demo è necessario dapprima scegliere una sorgente video con inquadratura fissa. Si può specificare una (e solo una) delle seguenti sorgenti:

1. --directory DIRECTORY: utilizza come sorgente video le immaigini contenute nella directory specificata (path specificato in DIRECTORY)
2. --camera CAMERA: utilizza come sorgente video una delle videocamere collegate al computer(CAMERA è un indice numerico che le identifica in cv2);
3. --image IMAGE: utilizza una singola immagine come sorgente video (IMAGE è il suo path);
4. --video VIDEO: utilizza un video come sorgente (VIDEO è il suo path);

Nel repository è presente una directory di test a scopo dimostrativo. All'interno ci sono 7 immagini, una per la taratura e 6 con soggetti posti a uno o due metri di distanza (come descritto nella relazione). Sebbene le proporzioni dei soggetti non siano accurte, le misure vengono prese dalla posizione dei piedi e quindi sono invarianti rispetto alla altezza del soggetto.

Inoltre è necessario fornire al sistema una rete neurale addestrata basata su YOLO in formato compatibile con pytorch. Si possono ottenere delle versioni pre-addestrate a sul repository ufficiale (https://github.com/ultralytics/yolov5)[https://github.com/ultralytics/yolov5]. Il repository ne include una versione di piccole dimensioni a scopo dimostrativo nella cartella checkpoints.

Per avviare il setup (taratura) bisogna richiamare l'azione specificando le dimensioni della scacchiera presente nell'immagine di calibrazione:

```
$: python main.py setup --checkerboard-cols 9 --checkerboard-rows 6 --
checkerboard-size 25 --directory "test/"
```

I parametri `--checkerboard-cols 9` `--checkerboard-rows 6` e `--checkerboard-size 25` devono rappresentare le dimensioni della scacchiera in termini di colonne e righe e dimensione lato della singola cella (in mm).

Si può sostituire `--directory "test/"` con la sorgente video che si desidera (comprese immagini, camera e video). Ad esempio, se il PC è predisposto di una webcam, si può utilizzare come sorgente `--camera 0`.

Verrà visualizzata una finestra che mostra la posizione della scacchiera rilevata. Il setup generà i dati per la correzione prospettica e li salverà in `homography.bin`. Una volta generati questi dati, si può eseguire il task di rilevamento distanza:

```
$: python main.py detect --model 'checkpoints/pretrained.pt --directory 'test/'
```

```
Output:
-------------------------
|     1     |     2     |
-------------------------
|     3     |     4     |
-------------------------
```

In output verrà mostrata una finestra con quattro immagini: (1) l'originale, (2) l'immagine con la prospettiva corretta, (3) una immagine che indica le persone rilevate e (4) una immagine con le distanze rilevate dall software.

Anche in questo caso, si può selezionare qualsiasi sorgente si desidera a scelta tra `--camera`, `--image`, `--directory`, `--video`.

Le finestre possono essere chiuse premendo il tasto `q`.

## Setupless walkthrough demo

Un metodo di funzionamento alternativo, usato principalmente a scopo di test, è quello di saltare la fase di setup ed avviare il software con:

```
$: python main.py detect --model 'checkpoints/pretrained.pt --directory 'test/' --
pixel-unit 0.75
```

Il software tenterà comunque di caricare `homography.bin`, ma se questa non è presente, utilizzerà la `--pixel-unit` e una trasformazione identitaria (quindi nessuna correzione prospettica). Questo metodo può essere utilizzato per valutare le performance di object detection del sistema, ma non restituisce misure accurate.

## Note macchina virtuale

Insieme al progetto è fornita una macchina virtuale in formato .ova con le dipendenze già installate.

Sono presenti due script `setup.sh` e `detect.sh` che eseguono le relative operazioni descritte in **Walkthrough demo**. Pertanto è sufficente avviare questi due script in ordine per eseguire la demo.

Inoltre, è incluso il dataset KORTE, che nell'ambito del progetto, è stato usato per calcolare la confusion matrix e le metriche di accuratezza della rete neurale. Per avviare l'evaluation:

```
$: python evaluation_main.py
```

Il dataset non è incluso nel repository originale per motivi di spazio.