

Smoking detection and distance analysis

Lorenzo Maria Alberto Paoria

12 marzo 2025

Sommario

L'idea è quella di automatizzare il processo di riconoscimento della violazione della nuova legge introdotta a Milano nel 2025 per semplificare il lavoro degli operatori.

1 L'idea

Il progetto "Smoking detection and distance analysis" nasce dall'idea di una soluzione per una nuova legge introdotta a Milano nel gennaio 2025. Essa punta obbliga i fumatori a fumare ad almeno 10 metri di distanza da una persona non fumatrice. Quindi il progetto cerca di automatizzare il più possibile la ricerca di infrazioni tramite l'uso di modelli pre-trained per il calcolo delle distanze e riconoscimento dei fumatori. In questo modo tramite un sistema di avvisi si potrebbe pensare di avvisare gli operatori in modo che possano successivamente andare a verificare la situazione.



Figura 1: Logo del programma creato tramite IA

2 Tecnologie utilizzate

2.1 Google colab

Google Colab (o Google Colaboratory) è un ambiente di sviluppo basato su cloud che permette di scrivere ed eseguire codice Python direttamente dal browser. È particolarmente utile per il machine learning, la data science e altre applicazioni che richiedono calcoli intensivi, poiché offre accesso gratuito a GPU e TPU.

2.2 Python

Python è un linguaggio di programmazione semplice, potente e versatile, utilizzato in molti ambiti, tra cui sviluppo web, data science, intelligenza artificiale e automazione. Per questo progetto vengono usati i jupyter notebook per migliorare l'efficienza sulla scrittura e esecuzione del codice.

2.3 Dataset

Il dataset è stato trovato sul sito di Roboflow, esso offre un set di immagini noisefree con la presenza di soggetti che possono fumare o meno. Esso presenta 3 set di immagini uno di allenamento, uno di validazione e l'ultimo per i test.

2.4 YOLO10

YOLO (You Only Look Once) è un algoritmo di Object Detection che permette di individuare e classificare oggetti in un'immagine o in un video in tempo reale. È uno dei modelli più veloci ed efficienti per il riconoscimento di oggetti. Nel progetto viene utilizzato per fare la detection delle persone, fumatrici e non e per le sigarette

2.5 Depth everything V2

Depth Anything V2 è un modello avanzato per la stima della profondità da immagini monoculari, sviluppato per fornire previsioni di profondità più dettagliate e robuste rispetto alla versione precedente. Per questo progetto abbiamo usato un modello vits ovvero un modello più piccolo e utilizzabile sul nostro ambiente Colab.

3 Schema progetto

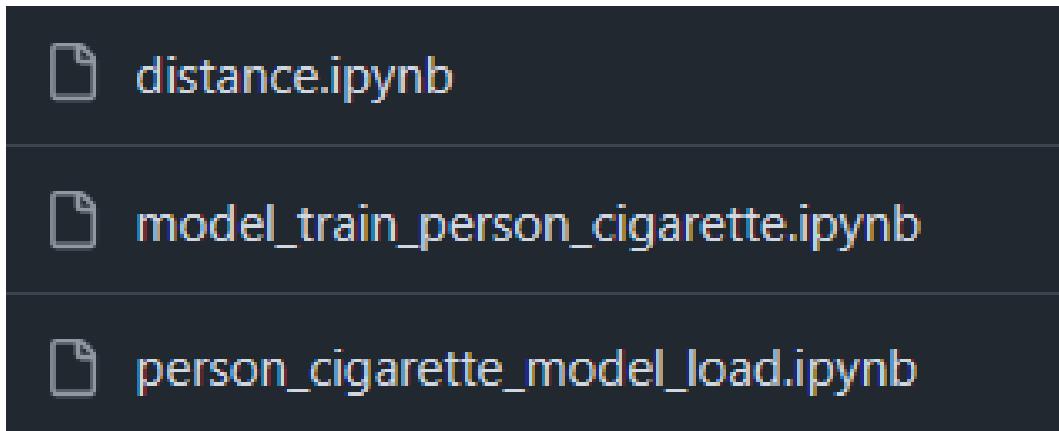


Figura 2: Schema progetto

Nel progetto abbiamo 3 jupiter notebook:

1. `model.train_person_cigarette` viene usato per l'allenamento del modello sulla base del dataset di allenamento e per ogni epoca di allenamento una fase di validazione dei risultati tramite l'apposito set di validazione. L'allenamento viene lanciato con un massimo di 100 epoche e una patience di 10 epoche.
2. `person_cigarette_model.load` viene usato per caricare il modello precedentemente allenato e viene lanciato in inferenza sul set di immagini per il test. Esso restituirà le immagini con le bounding box disegnate con sopra scritta la classe di appartenenza e la confidence della bb. Inoltre vengono salvate le coordinate per ogni foto delle bounding box sui fumatori e non fumatori in modo tale che successivamente possiamo prenderle per il calcolo delle distanze.
3. `distance` invece viene usato per calcolare le distanze tra le bounding box dei fumatori e dei non fumatori, esso usa la lunghezza focale della camera e una mappa di profondità per stimare l'asse z della foto tramite il modello `depthAnything-v2`.

4 Acquisizione dataset e Google Colab

Una volta scaricato il dataset da Roboflow con l'apposito formato per il modello YOLO, vediamo che esso contiene tre cartelle una con all'interno le immagini per il training del modello, uno per la validazione del modello e in fine quello per i test. L'intero progetto verrà modellato per l'uso su googleColab, infatti i modelli utilizzati sono modelli "leggeri" e per questo i risultati ottenuti non sono dei migliori. Tramite Colab esso ci da una possibilità di prendere in prestito una gpu T4 che ci permette di avere dei risultati in un tempo non troppo esteso, d'altra parte però abbiamo un uso ridotto della RAM.

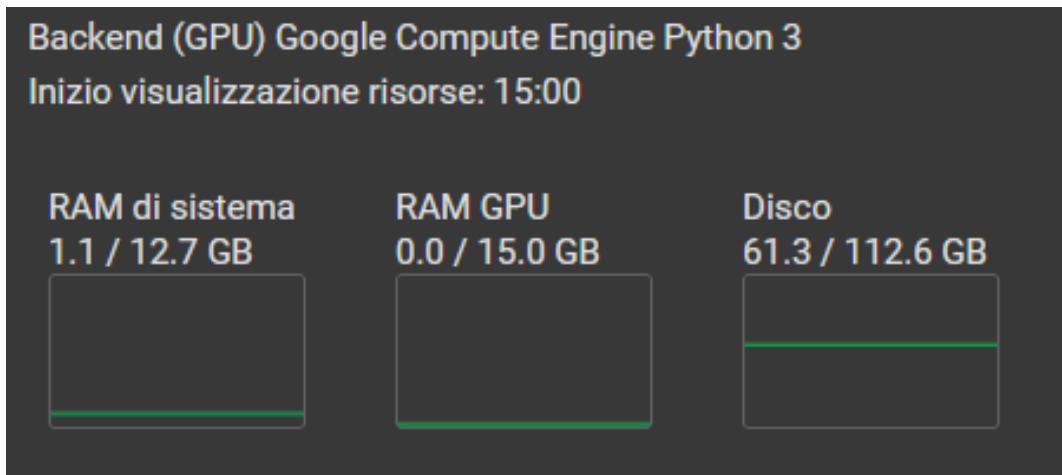


Figura 3: Risorse Google Colab

Da ora in poi seguiremo lo sviluppo della foto numero 11:



Figura 4: Foto numero 11 del set di test

5 Modello in allenamento

Ora che abbiamo sia il dataset che l'ambiente di sviluppo creiamo tramite jupyter notebook il primo programma che si occupa di allenare il modello. Infatti tramite il set di foto di allenamento viene allenato un modello base di YOLO per il riconoscimento dei fumatori, dei non fumatori e delle sigarette. Per ogni epoca di training in seguito ne segue una di validazione, inoltre alla fine dell'esecuzione del programma avremo due pth differenti. Il primo pth sarà l'ultimo modello quindi l'ultima esecuzione, mentre il secondo sarà il modello con il best mAP ovvero secondo le metriche di precisione e richiamo il modello che ha le migliori prestazioni viene salvato in modo da tenere sempre il modello migliori anche con l'avanzare delle epoche. Il modello verrà fatto allenare fin quando l'early stopping non entrerà in azione. Infatti tramite questo noi possiamo dire al programma che se il modello non migliora nell'arco di 10 epoche allora si può fermare e prendiamo i modelli salvati fino a quel punto. Per le metriche mAP usiamo in particolare mAP@0.5 ovvero il modello è considerato "corretto" se la previsione di bounding box ha un IoU (IoU è l'area di sovrapposizione tra il bounding box predetto e il bounding box reale diviso per l'area di unione tra i due) maggiore di 0.5 rispetto al box reale.

6 Modello in inferenza

Ora che abbiamo un modello ben allenato sul dataset lo possiamo lanciare in inferenza sul set di test. Il programma che si occupa di questo ovvero il person_cigarette_model.load non solo disegnerà le bounding box dei fumatori, non fumatori e delle sigarette ma scrivere in dei json le coordinate delle bounding box per ogni foto in modo tale che successivamente possiamo prenderli per calcolare le distanze. Sopra le bouding box troviamo dei dati, il primo è la classe di appartenenza, infatti la classe 0 sono le sigarette mentre la classe 1 sono i non fumatori e la classe 2 i fumatori. Oltre a questo possiamo vedere la soglia di confidenza per la bouding box.

Da qui possiamo vedere i risultati per la foto numero 11:

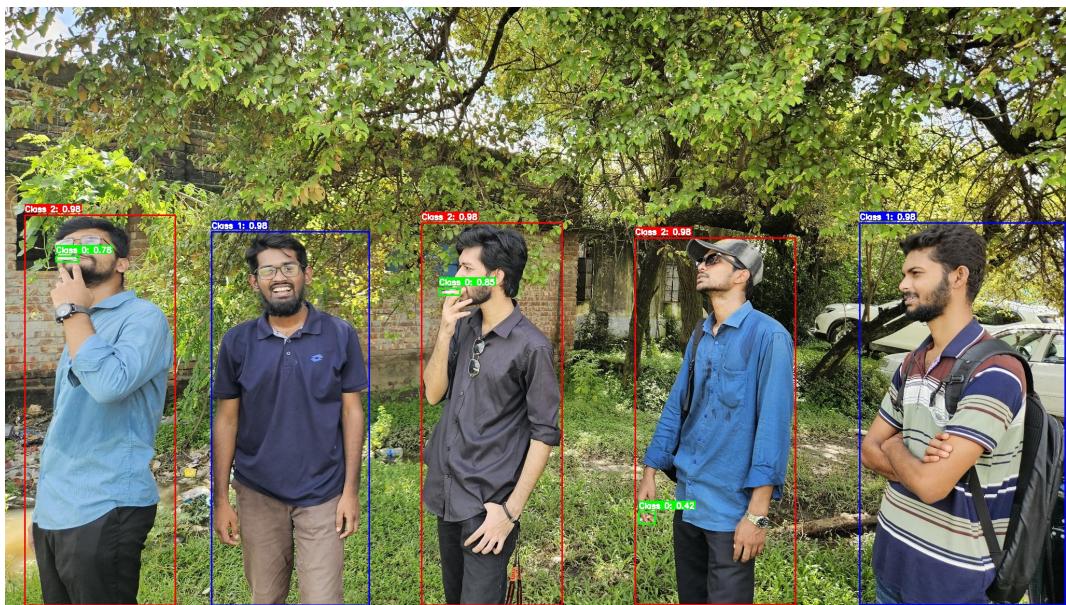


Figura 5: Foto numero 11 del set di test processato dal modello in inferenza

7 Calcolo delle distanze

Una volta processate tutte le immagini del test possiamo iniziare a calcolare le distanze. Noi sappiamo appunto che l'immagine è in 2d quindi ci manca appunto l'asse delle Z, ottenibile utilizzando un modello in grado di capire la profondità degli oggetti analizzando caratteristiche visive come:

1. Struttura e contorni degli oggetti
2. Dimensione e prospettiva
3. Disposizione spaziale e occlusioni

Esso ci ritornerà una mappa della profondità colorata in modo tale che gli oggetti più vicini siano delineati da un colore differente.



Figura 6: mappa di profondità della foto numero 11

In questo modo possiamo assegnare una stima della profondità dell'immagine e calcolarci così la distanza tra i soggetti fumatori dai non.

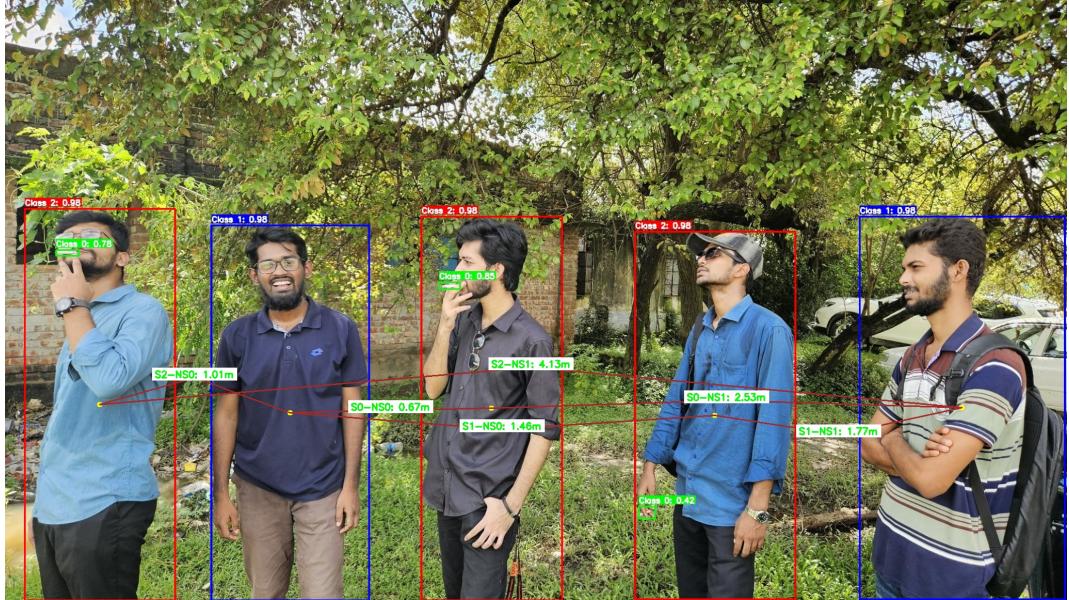


Figura 7: Foto numero 11 con le distanze calcolate

Il calcolo è stato eseguito per la maggior parte dalla funzione calculate_3d_distance nel seguente modo:

1. Calcola il centro delle bounding box delle persone,
2. Controlliamo che i centri delle bb siano all'interno della depth map,
3. Calcoliamo la profondità media in un'area di 5 pixel intorno al centro per evitare errori nel caso di presenza di rumori,

4. Dopodiché vengono convertite le coordinate da 2D a 3D, infatti i punti centrali delle bounding box vengono scalati rispetto alla profondità usando la formula di proiezione pinhole camera per calcolare le coordinate reali di X e di Y,
5. Viene dopodichè calcolata la distanza euclidea in 3D ovvero la distanza tra i due punti.

7.0.1 Formula di proiezione pinhole camera

La formula di proiezione della camera a foro stenopeico (pinhole camera) descrive la relazione tra le coordinate tridimensionali (3D) di un punto nello spazio e le sue coordinate bidimensionali (2D) sull'immagine catturata dalla fotocamera. Questo modello semplificato è fondamentale in computer vision e grafica computerizzata per comprendere come le fotocamere trasformano una scena 3D in un'immagine 2D e viene usata anche per modelli di camera che non sono a pinhole per semplificare i calcoli. Nel modello della camera a foro stenopeico, la relazione tra un punto nello spazio 3D e la sua proiezione sul piano dell'immagine è descritta così:

1. Coordinate del punto 3D: (X,Y,Z)(X,Y,Z)
2. Coordinate del punto 2D sul piano dell'immagine: (x,y)(x,y)
3. Lunghezza focale della fotocamera: focal_length

La relazione tra un punto nello spazio 3D e la sua proiezione sul piano dell'immagine è data dalle equazioni:

$$x = \frac{\text{focal_length} \cdot X}{Z}$$

$$y = \frac{\text{focal_length} \cdot Y}{Z}$$

Dove:

1. (X,Y,Z)(X,Y,Z) sono le coordinate del punto nello spazio 3D.
2. (x,y)(x,y) sono le coordinate sul piano dell'immagine.
3. focal_length è la distanza tra il foro stenopeico e il piano dell'immagine (lunghezza focale).

Possiamo però invertire le formule per poter passare da uno spazio 2D a 3D nel seguente modo:

$$X = \frac{x \cdot Z}{\text{focal_length}}$$

$$Y = \frac{y \cdot Z}{\text{focal_length}}$$

Formule prese su <https://pytorch3d.org/docs/cameras>

7.0.2 Formula distanza Euclidea in 3D

La distanza euclidea tra due punti $P_1(x_1, y_1, z_1)$ e $P_2(x_2, y_2, z_2)$ nello spazio tridimensionale è data dalla formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Dove:

1. (x_1, y_1, z_1) sono le coordinate del primo punto.
2. (x_2, y_2, z_2) sono le coordinate del secondo punto.
3. d è la distanza tra i due punti.

La distanza euclidea è come tirare una corda tra due punti e misurarne la lunghezza.

8 Problematiche di sviluppo

Il programma se messo in un ambiente di sviluppo dotato di risorse migliori potrebbe essere molto più accurato, infatti si potrebbero utilizzare modelli più accurati che sono più complicati da gestire con delle risorse molto basse. Lo stesso depth anything viene utilizzato in una versione vitS che è molto meno precisa ma molto più leggera.

Inoltre anche avere più dati per il dataset come la fotocamera utilizzata, l'obbiettivo e quindi le caratteristiche come la lunghezza focale e il sensore potrebbero migliorare di molto l'accuratezza del programma. Oltre a magari molte più foto per far in modo che l'addestramento dei modelli sia migliorato ulteriormente.

Con l'attuale progettazione inoltre si stima che ci sia un possibile errore medio compreso tra i 40 centimetri e 1 metro, questo sempre dovuto al dataset ma anche ai modelli utilizzati. Con le giuste accortezze si potrebbe diminuire l'errore medio fino a 10/30 centimetri.

9 Risultati vari

Di seguito un altro set di foto:

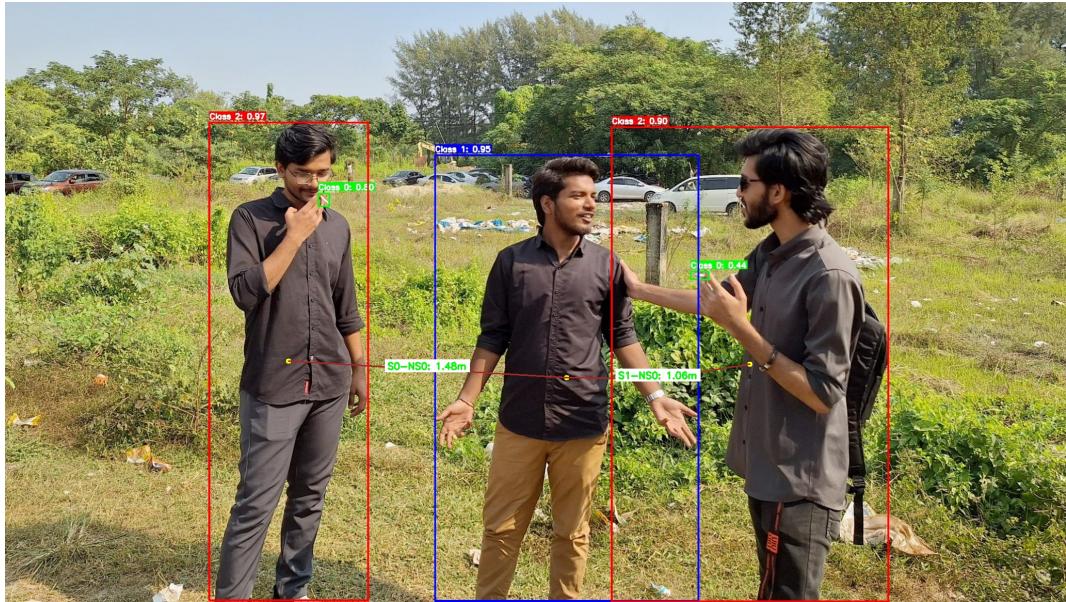


Figura 8: Foto numero 52



Figura 9: Foto numero 63

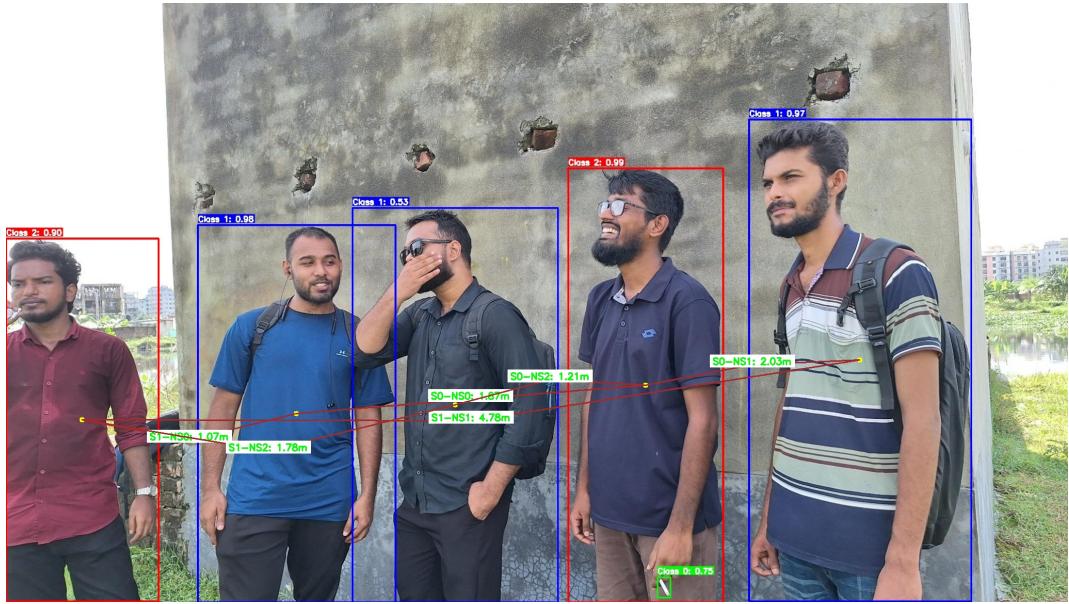


Figura 10: Foto numero 68

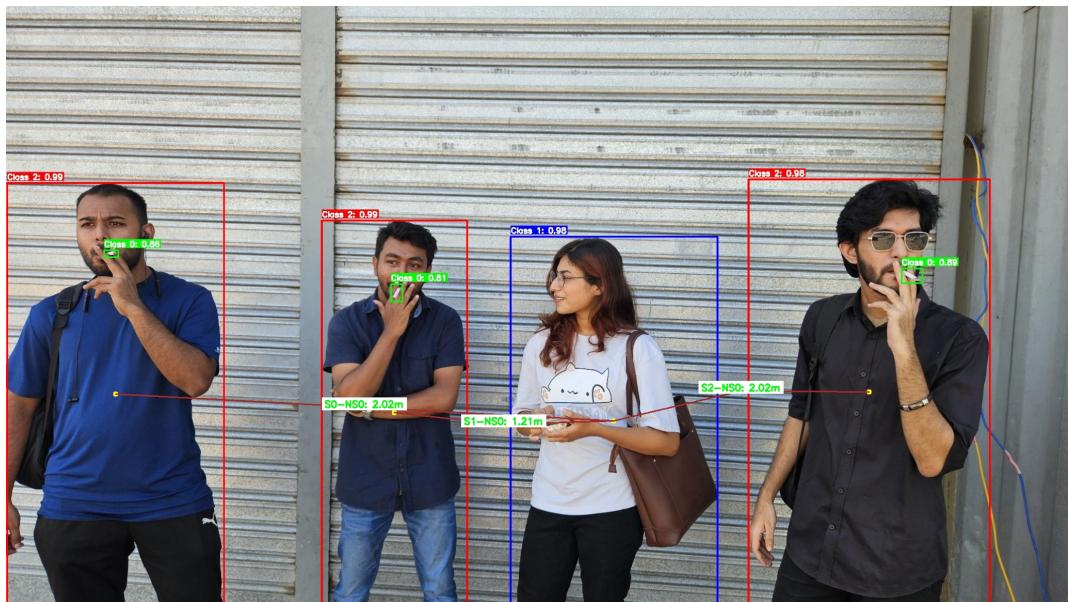


Figura 11: Foto numero 70



Figura 12: Foto numero 72

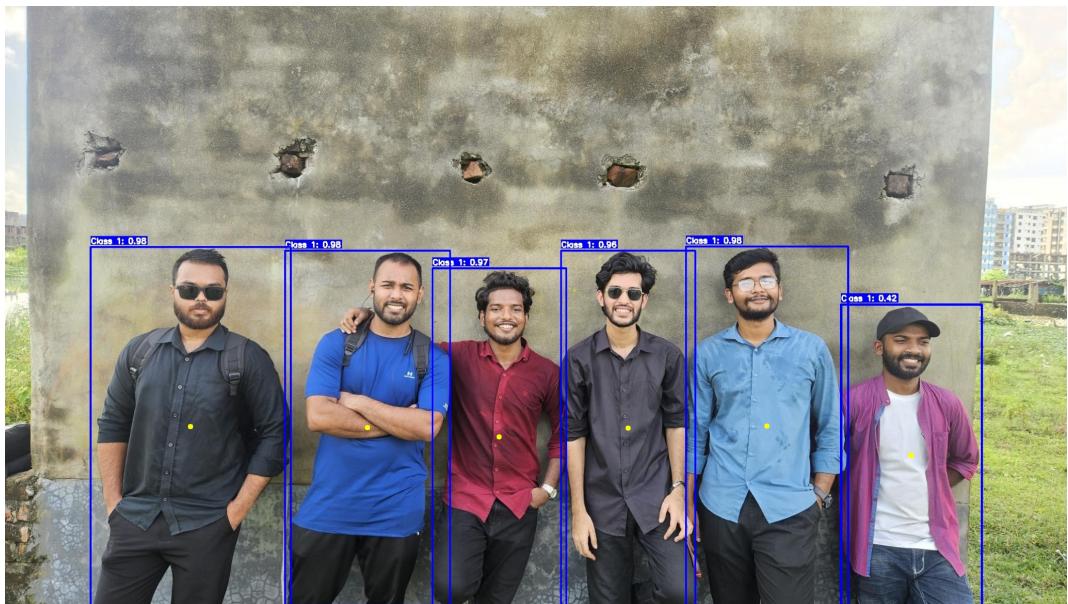


Figura 13: Foto numero 73