Invisible Man

E.I. 2021/22

Mazzitelli Davide
Motta Giacomo

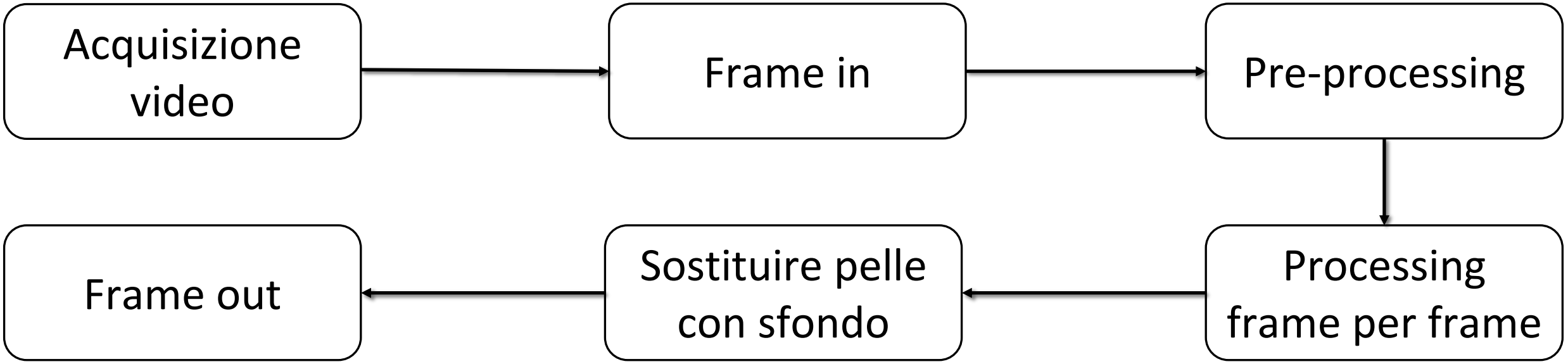
Problema

Si vuole realizzare una applicazione che, dato uno stream video fa diventare la persona ripresa nel video invisibile

Considerazioni

- Autofocus : I video utilizzati come input devono essere sprovvisti di autofocus
- Background : Il primo frame del video passato in input al programma costituirà il frame di background
- Rumore : in fase di pre-processing tenere conto del rumore che potrebbe creare artefatti

Procedimento



Pre-processing

Azioni di miglioramento della qualità dell'immagine per semplificare le elaborazioni successive

Rimozione del rumore

Mitigare il rumore causato da circuiti digitali e dal trasferimento del file video

Come?

Tramite l'applicazione di un filtro mediano

```
%applicazione filtro mediano 3x3
image(:,:,1) = medfilt2(image(:,:,1));
image(:,:,2) = medfilt2(image(:,:,2));
image(:,:,3) = medfilt2(image(:,:,3));
```

Accuratezza media migliorata del 5%

Skin detection

Abbiamo valutato e testato diversi metodi per skin detection basati sull'elaborazione con piani colore al fine di valutare il miglior procedimento correlato al nostro problema.

In particolare abbiamo cercato di estrarre una soglia ottimale che includa i pixel di pelle del frame

RGB

Soglia utilizzata illustrata nella ricerca
(https://medium.com/swlh/human-skin-color-classification-using-the-threshold-classifier-rgb-ycbcr-hsv-python-code-d34d51febdf8)

Utilizzo di morfologia matematica (chiusura) per eliminare dall'immagine occhi e bocca

Problemi:

- Risultati poco pleasing
- Illuminazione della scena

YCbCr

Conversione di ogni frame da RGB a YCbCr e creazione maschera utilizzando soltanto le componenti Cb e Cr.

Soglia utilizzata illustrata nella ricerca (Explicit Image Detection using YCbCr Space Color Model as Skin Detection).

Problematiche:

- Risultati poco pleasing
- Imprecisioni nell'identificazione di pixel di pelle

YCbCr

Utilizzo di un dataset per l’estrazione di features media e deviazione standard.

Dataset: SFA per la face recognition; 3354 immagini di pelle da 35 pixel ciascuna.

soglia ottima attorno alla media delle componenti Cb e Cr utilizzando un peso in modo tale che sia abbastanza inclusiva

Problematiche:

- Imprecisioni nel riconoscimento della pelle

Risultati

Abbiamo quindi eseguito lo stesso procedimento utilizzando però lo spazio colore HSV.

Abbiamo quindi ottenuto un insieme di risultati di diversi metodi:

Risultati

	True positive	True negative	False positive	False negative	Accuracy
RGB	81%	96%	4%	19%	91%
YCbCr	93%	89%	11%	7%	90%
YCbCr(m, std)	93%	88%	12%	7%	90%
HSV(m, std)	99%	82%	18%	1%	86%

Selezione metodo ottimale

Per la risoluzione del nostro problema le caratteristiche più importanti da considerare sono i risultati dei True positive e dei False negative .

La presenza di falsi positivi nelle parti di immagini che costituirebbero il background non costituiscono un grande problema anche se presenti, in quanto verrebbero sostituiti con il frame di background

Mentre l'identificazione del più alto numero possibile di pixel di pelle rappresenta l'obiettivo principale.

Abbiamo quindi deciso di utilizzare come metodo ottimale il procedimento che utilizza lo spazio colore HSV.

Questo procedimento ci permette di ottenere infatti il più alto numero di True positive mantenendo comunque una soglia non eccessiva di False positive

Metodo ottimale: estrazione delle features

- Creazione lista di nomi delle immagini di pelle presenti nel dataset SFA
- Ciclando la lista, leggo l'immagine e la trasformo in formato HSV
- Creazione del vettore delle features (train values)
- Calcolo della media e della deviazione standard per componente

```
names_skin = createdata("names_skin.txt");

imTmp = (imread("35_skin/skin-1-1.jpg"));
imTmp=rgb2hsv(imTmp);
[righe, colonne, ch] = size(imTmp);
imTmp = reshape(imTmp,righe*colonne, 3);
train_values = imTmp;

for j = 2:(1118*3)
    im = (imread("35_skin/"+names_skin{j}));
    im=rgb2hsv(im);
    [righe, colonne, ch] = size(im);
    im = reshape(im,righe*colonne, 3);
    train_values = [train_values; im];
end

m = mean(train_values); % media per colore

s = std(double(train_values)); %deviaz standard
```

Media ottenuta:
m=[0.0715,0.4212,0.5936]
Deviazione standard ottenuta:
s=[0.0302,0.0992,0.1745]

Metodo ottimale:

- Salvataggio del frame di background e applicazione ad esso del filtro mediano (3x3)
- Apertura dello stream video di output
- Caricamento tramite comando load di media e deviazione standard; setting di variabile k (peso utilizzato nella determinazione della soglia)
- Ciclare con un while su ogni frame del video la funzione process_frame (parametri: frame, media, deviazione standard, k, background)

```
vid = VideoReader('video_da_processare');

if hasFrame(vid)
    background = readFrame(vid);
end

%applicazione filtro mediano 3x3
background(:,:,1) = medfilt2(background(:,:,1));
background(:,:,2) = medfilt2(background(:,:,2));
background(:,:,3) = medfilt2(background(:,:,3));

video_out = VideoWriter('video_out.avi');
video_out.FrameRate=30;
open(video_out);

load("meanhsv.mat");
load("stdhsv.mat");
k=1.5;

while hasFrame(vid)

    frame = readFrame(vid);

    frame = process_framehsv(frame, m, s, k, background);
```

Metodo ottimale: process_frame

- Applicazione al frame del filtro mediano 3x3
- Applicazione al frame di un filtro di media 3x3
- Conversione del frame in formato HSV
- Creazione della maschera di skin detection utilizzando solo i componenti H e S
- Applicazione alla maschera di morfologia matematica in modo da eliminare occhi e bocca dalla maschera
- Sottrazione all'immagine originale della maschera creata
- Somma del risultato ottenuto con l'applicazione della maschera al frame di background

```
function outImage = process_framehsv(image, m, s, k, background)


%applicazione filtro mediano 3x3
image(:,:,1) = medfilt2(image(:,:,1));
image(:,:,2) = medfilt2(image(:,:,2));
image(:,:,3) = medfilt2(image(:,:,3));
f5 = fspecial('average',3);
image(:,:,1) =imfilter(image(:,:,1),f5);
image(:,:,2) =imfilter(image(:,:,2),f5);
image(:,:,3) =imfilter(image(:,:,3),f5);

imagehsv=rgb2hsv(image);

mask_h = (imagehsv(:,:,1) >= m(1) - k*s(1)) & (imagehsv(:,:,1) <= m(1) + k*s(1));
mask_s = (imagehsv(:,:,2) >= m(2) - 2*k*s(2)) ;


predicted = mask_h & mask_s ;

s = strel('disk', 17);
predicted=imclose(predicted,s);
%imshow(predicted);

imageTmp = image - (255*uint8(predicted));

outImage = imageTmp + (uint8(predicted).* background);



end
```

Metodo ottimale:

- Scrivo nel file video_out il frame processato
- Alla fine del ciclo: chiudo lo stream video

```
    writeVideo(video_out,frame);

end

close(video_out);

delete(vid);
```

Metodo ottimale: Risultati

Le immagini trattate per calcolare i nostri risultati sono le immagini del dataset SFA che già disponeva di coppie immagine originale-immagine ground truth

- Caricamento tramite comando load di media e deviazione standard utilizzate per processare i frame e set del peso k utilizzato
- readlists(), creazione di due liste contenenti i nomi delle immagini originali su cui effettuare la valutazione (list_ori) e i nomi delle immagini di ground truth (list_gt)
- Set di accuracy_tot, TP_tot, FP_tot, FN_tot a 0
- Nel ciclo for: salvo nella variabile im_ori l'immagine i-esima da testare
- Salvo nella variabile im_gt l'immagine i-esima di ground truth
- Binarizzo l'immagine di ground truth
- Tramite la funzione `process_predicted(im_ori, k, m, s)` processo l'immagine da testare in modo da essere confrontata con quella di ground truth
- Creazione della matrice di confusione attraverso la funzione confmat
- Incremento accuracy_tot, TP_tot, FP_tot, FN_tot dei risultati contenuti nella matrice di confusione
- Fuori dal ciclo: calcolo accuratezza, TP, FP e FN medi dividendo per 1118

```
clear;
close all;

load("meanhsv.mat");
load("stdhsv.mat");
k=1.5;
[list_ori, list_gt] = readlists();
accuracy_tot = 0;
TP_tot = 0;
FP_tot=0;
FN_tot=0;
for i = 1 : 1118
    im_ori = imread(list_ori{i});
    im_gt = imread(list_gt{i});

    im_gt = rgb2gray(im_gt);
    im_gt = im2bw(im_gt,0.2);

    im_ori = process_predicted(im_ori, k, m, s);

    results = confmat(im_gt, im_ori);

    accuracy_tot = accuracy_tot + results.accuracy;
    TP_tot=TP_tot + results.cm(2,2);
    FP_tot=FP_tot + results.cm(1,2);
    FN_tot = FN_tot + results.cm(2,1);
end
Mean_accuracy = accuracy_tot/1118
TP_tot=TP_tot/1118
FP_tot = FP_tot/1118
FN_tot = FN_tot/1118
```

Procedimento: Risultati

- Applicazione al frame del filtro mediano 3x3
- Applicazione al frame di un filtro di media 3x3
- Conversione del frame in formato HSV
- Creazione della maschera di skin detection
- Applicazione alla maschera di morfologia matematica

```
function outImage = process_predicted(image, k, m, s)


%applicazione filtro mediano
image(:,:,1) = medfilt2(image(:,:,1));
image(:,:,2) = medfilt2(image(:,:,2));
image(:,:,3) = medfilt2(image(:,:,3));
f5 = fspecial('average',3);
image(:,:,1) =imfilter(image(:,:,1),f5);
image(:,:,2) =imfilter(image(:,:,2),f5);
image(:,:,3) =imfilter(image(:,:,3),f5);

imagehsv=rgb2hsv(image);

mask_h = (imagehsv(:,:,1) >= m(1) - k*s(1)) & (imagehsv(:,:,1) <= m(1) + k*s(1)
mask_s = (imagehsv(:,:,2) >= m(2) - 2*k*s(2)) ;
% & (image(:,:,2) <= m(2) + k*s(2))

predicted = mask_h & mask_s ;

s = strel('disk', 17);
outImage=imclose(predicted,s);

end
```

Conclusioni

Abbiamo scelto il procedimento che fa uso dello spazio colore HSV sulla base delle caratteristiche che più potevano essere utili ai fini del problema posto.

Lavorando su Hue e Saturation, si ottiene facilmente una separazione di pelle dagli indumenti, e il range molto inclusivo della maschera permette di rimuovere anche i capelli.

La presenza di falsi positivi facenti parte del background, inoltre, è una caratteristica che non costituisce un grande problema, in quanto le porzioni di immagine individuate verrebbero poi sostituite con il background.

I true-positive ed i false-negative, invece, sono le caratteristiche su cui ci siamo maggiormente concentrati, in quanto permettono di determinare un miglior risultato sulla base del problema posto.