Probabilistic Skin Detection

CSE 6367 – Computer Vision
Vassilis Athitsos
University of Texas at Arlington

Probabilistic Skin Detection

- Maximum likelihood approach:
  - given training data:
    - Estimate P(RGB | skin), P(RGB | non-skin)
  - given test data:
    - For every pixel, compute P(skin | RGB), using Bayes rule.

- P(RGB | skin):
  - probability that we will observe rgb, when we know that the pixel is a skin pixel.

Obtaining Training Data

`frame2`

`frame2(80:120, 137:172, :)`

```
% finding a training sample
figure(2); imshow(frame2(80:120, 137:172, :) / 255);
sample = frame2(80:120, 137:172, :);
```

- Find subwindows that only contain skin pixels.
  - For good results, collect data from many images.
  - Here, for simplicity, we only use data from one image.

A Simple Gaussian Model

- All we need: mean and std.
- Assumption: colors are mutually independent.

```
sample_red = sample(:, :, 1);
sample_green = sample(:, :, 2);
sample_blue = sample(:, :, 3);

sample_red = sample_red(:);
sample_green = sample_green(:);
sample_blue = sample_blue(:);
```

```
red_mean = mean(sample_red);
green_mean = mean(sample_green);
blue_mean = mean(sample_blue);

red_std = std(sample_red);
green_std = std(sample_green);
blue_std = std(sample_blue);
```

Probability of a Color

- Given the means and stds for each color:
  - What is P(RGB | skin)?
  - How can it be decomposed?

Probability of a Color

- Given the means and stds for each color:
  - What is P(RGB | skin)?
  - How can it be decomposed?
- Assuming that colors are independent:
  - P(RGB | skin) =

    = P(R | skin) * P(G | skin) * P(B | skin)

    = N(R, red_mean, red_std) *

    N(G, green_mean, green_std) *

    N(B, blue_mean, blue_std)

% N is the normal (Gaussian) distribution

Applying the Skin Model

```
frame20 = double(imread('frame20.bmp'));
[rows,cols, bands] = size(frame20);

skin_detection = zeros(rows, cols);

for row = 1:rows
    for col = 1:cols
        red = frame20(row, col, 1);
        green = frame20(row, col, 2);
        blue = frame20(row, col, 3);

        red_pr = gaussian_probability(red_mean, red_std, red);
        green_pr = gaussian_probability(green_mean, green_std, green);
        blue_pr = gaussian_probability(blue_mean, blue_std, blue);
        prob = red_pr * green_pr * blue_pr;
        skin_detection(row, col) = prob;
    end
end
```

Results

frame20

skin_detection

skin_detection > 0.0000001

Switching to Normalized rg Space

- r = R / (R+G+B);
- g = G / (R+G+B);
- Intuition: intensity does not matter.

```
sample_total = sample_red + sample_green + sample_blue;
sample_red2 = sample_red ./ sample_total;
sample_red2(isnan(sample_red2)) = 0;
sample_green2 = sample_green ./ sample_total;
sample_green2(isnan(sample_green2)) = 0;

r_mean = mean(sample_red2);
g_mean = mean(sample_green2);
r_std = std(sample_red2);
g_std = std(sample_green2);
```

Probability of a Color in rg Space

- Given the means and stds for each color:
  - What is P(RGB | skin)?
  - How can it be decomposed?

Probability of a Color

- Given the means and stds for each color:
  - What is P(RGB | skin)?
  - How can it be decomposed?
- Assuming that colors are independent:
  - r = R / (R+G+B);
  - g = G / (R+G+B);
  - P(RGB | skin) = P(rg | skin)

    = P(r | skin) * P(g | skin)

    = N(r, r_mean, r_std) * N(g, g_mean, g_std)

Results

frame20

skin_detection2

skin_detection2 > 10

Compare to Results Using RGB

`frame20`

`skin_detection`

`skin_detection > 0.0000001`

Probability Question

- Why are some values > 1?
- Why does $P(rg \mid skin) > 1$ for some rg?
  - Why is it not violating the rule of probability theory that probabilities always <= 1?

Probability Question

- Why are some values > 1?
- Why does P(rg | skin) > 1 for some rg?
  - Why is it not violating the rule of probability theory that probabilities always <= 1?
- Answer: because P(rg | skin) is a *density* function, not a discrete probability function.

Towards a Nonparametric Model

- How many colors are there?
  - Assuming we use 8 bits per color.

Towards a Nonparametric Model

- How many colors are there?
  - Assuming we use 8 bits per color.
- 256 * 256 * 256 = 16.777 million colors

- How can we overcome the Gaussian assumption?
  - Assume we have lots of training data.

Towards a Nonparametric Model

- How many colors are there?
  - Assuming we use 8 bits per color.
- 256 * 256 * 256 = 16.777 million colors

- How can we overcome the Gaussian assumption?
  - Assume we have lots of training data.
- By estimating explicitly P(RGB | skin) for each RGB.

Color Histograms

- Simplest form:
  - A 256x256x256 array.
- Given training samples of skin:
  - For each bin (R, G, B) of histogram:
    - Count how many pixels have color RGB in the training samples.
- What is P(RGB | skin) according to the histogram?

Color Histograms

- Simplest form:
  - A 256x256x256 array.
- Given training samples of skin:
  - For each bin (R, G, B) of histogram:
    - Count how many pixels have color RGB in the training samples.
- What is P(RGB | skin) according to the histogram?
  - histogram(R,G,B) / sum of all histogram bins.

Practical Considerations

- Estimating 16.8 million numbers requires too much training data.
  - How much?

Practical Considerations

- Estimating 16.8 million numbers requires too much training data.
  - How much?
- If we want 10 more pixels than the number of bins:
  - 168 million skin pixels.
  - 67200 50x50 skin patches.

Practical Considerations

- Estimating 16.8 million numbers requires too much training data.
  - How much?
- If we want 10 more pixels than the number of bins:
  - 168 million skin pixels.
  - 67200 50x50 skin patches.
- Remedy: make a coarser histogram.

A 32x32x32 Histogram

- A 32x32x32 array.
  - 32,768 entries.
- If we want 10 more pixels than the number of bins:
  - 327,680 million skin pixels.
  - 131 50x50 skin patches.
- Color (R,G,B) maps to bin:
  - floor(R/8, G/8, B/8).
    - Assuming bins are numbered starting at 0.
  - Matlab formula: floor(R/8, G/8, B/8) + 1.

```
function result = detect_skin2(image, positive_histogram)

% function result = detect_skin2(image, positive_histogram)

vertical_size = size(image, 1);
horizontal_size = size(image, 2);
histogram_bins =  size(positive_histogram, 1);
factor = 256 / histogram_bins;

result = zeros(vertical_size, horizontal_size);

for vertical = 1: vertical_size
    for horizontal = 1: horizontal_size
        red = image(vertical, horizontal, 1);
        green = image(vertical, horizontal, 2);
        blue = image(vertical, horizontal, 3);

        r_index = floor(red / factor) + 1;
        g_index = floor(green / factor) + 1;
        b_index = floor(blue / factor) + 1;

        skin_value = positive_histogram(r_index, g_index, b_index);
        result(vertical, horizontal) = skin_value;
    end
end
```

Results

frame20

result

result > 0.0002

Compared to Gaussian rg Model

frame20

skin_detection2

skin_detection2 > 10

Parametric and Non-parametric Models

- Parametric models:
  - We assume type of distribution.
  - We compute parameters.
- Gaussians are parametric distributions.
  - Parameters are mean and std.
- Histograms are *non-parametric* distributions.
  - No assumption about how values are distributed.
  - Plus: Fewer assumptions → more robust system.
  - Minus: Must estimate a lot more numbers → we need a lot more training data.

What Is Missing?

- We have tried three skin color models:
  - A Gaussian RGB distribution.
  - A Gaussian rg distribution.
  - A histogram-based distribution.
- Using each of them, we compute for each pixel:

What Is Missing?

- We have tried three skin color models:
  - A Gaussian RGB distribution.
  - A Gaussian rg distribution.
  - A histogram-based distribution.
- Using each of them, we compute for each pixel:
  - P(RGB | skin).
- What do we *really* want to compute?

What Is Missing?

- We have tried three skin color models:
  - A Gaussian RGB distribution.
  - A Gaussian rg distribution.
  - A histogram-based distribution.
- Using each of them, we compute for each pixel:
  - P(RGB | skin).
- What do we *really* want to compute?
  - P(skin | RGB).
- What is the difference?

Review of Bayes Rule

- Relating P(skin | RGB) to P(RGB | skin):

Review of Bayes Rule

- Relating P(skin | RGB) to P(RGB | skin):

  $P(\text{skin} \mid \text{RGB}) = P(\text{RGB} \mid \text{skin}) * P(\text{skin}) / P(\text{RGB})$.

- What is P(RGB)?

Review of Bayes Rule

- Relating P(skin | RGB) to P(RGB | skin):

  P(skin | RGB) = P(RGB | skin) * P(skin) / P(RGB).

- What is P(RGB)?

  P(RGB) = P(RGB | skin) * P(skin) + P(RGB | non_skin) * P(non_skin)

- We need P(RGB | non_skin) and P(skin).
  - How do we get P(RGB | non_skin)?

Review of Bayes Rule

- Relating P(skin | RGB) to P(RGB | skin):

  P(skin | RGB) = P(RGB | skin) * P(skin) / P(RGB).

- What is P(RGB)?

  P(RGB) = P(RGB | skin) * P(skin) + P(RGB | non_skin) * P(non_skin)

- We need P(RGB | non_skin) and P(skin).
  - How do we get P(RGB | non_skin)?

Non-skin Color Histogram

- A skin histogram estimates P(RGB | skin).
  – Based on skin samples.
- A non-skin histogram estimates P(RGB | non-skin).
  – Based on non-skin samples.

Implementation

- Pick manually P(skin).
  - I always use 0.5.
    - It is clearly too high.
    - Results are good.
- Then, P(skin | RGB) =

P(RGB | skin) * 0.5 / (0.5 * P(RGB | skin) + 0.5 * P(RGB | non_skin)) =

P(RGB | skin) / (P(RGB | skin) + P(RGB | non_skin))

- Full implementation: code/detect_skin.m

Calling detect_skin.m

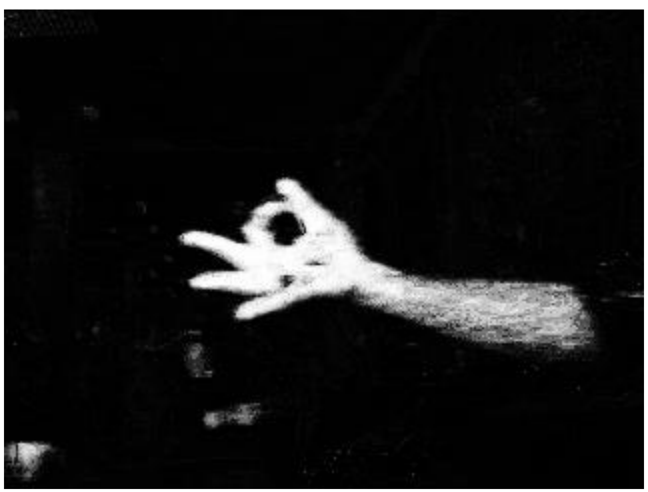
```
% read histograms
clear;
negative_histogram = read_double_image('negatives.bin');
positive_histogram = read_double_image('positives.bin');

frame20 = double(imread('frame20.bmp'));
%figure(1); imshow(frame2 / 255);

result = detect_skin(frame20, positive_histogram,  negative_histogram);
figure (5); imshow(result, []);
```

Results

frame20



result

result > 0.2

Compare to Using Only Skin Histogram

frame20

result

result > 0.0002

Results

frame20

result

result > 0.2

Compared to Gaussian rg Model

`frame20`

`skin_detection2`

`skin_detection2 > 10`