

# Visual measurement of jackets by structured prediction

J. Serrat, Oriol Ramos

Master in CV, module M3, course 2017-18



Computer Vision Center, {oriolrt | joans}@cvc.uab.es

# Index

1 Problem

2 Image capture

3 Samples

4 Method

5 What to do

# Problem

A company sells jackets custom tailored to the sizes specified by their clients through its website.

The screenshot shows a website for 'CUSTOM CLOTHING MADE EASIER'. The main banner features a man in a dark suit with a red and blue striped tie. Text on the banner includes 'CUSTOM SUITS FROM 189€ DESIGN NOW >', 'CUSTOM SHIRTS FROM 42€ DESIGN NOW >', and 'Linen Waves 100% Linen Shirts Collection'. Below the banner, there's a section titled 'LATEST NEWS' with three items: 'GET A FREE SUIT' (with a cartoon character icon), 'Linen Waves 100% Linen Shirts Collection' (with a man in sunglasses), and 'Discover our new clothing line for women' (with a woman in a blazer). Each news item has a 'See More' button.

# Problem

Sometimes customers return items complaining they don't fit well.

Causes :

- ▶ customers don't measure themselves well with a tailor tape
- ▶ input wrong measures to the web page
- ▶ tailor had a bad day
- ▶ ??

There's a need for comparing the actual measures of an item with those provided by customers.

## Goal

Perform automatic measurements on each garment at *pre-shipment time* for quality control, by means of computer vision techniques.

# Problem

What to measure

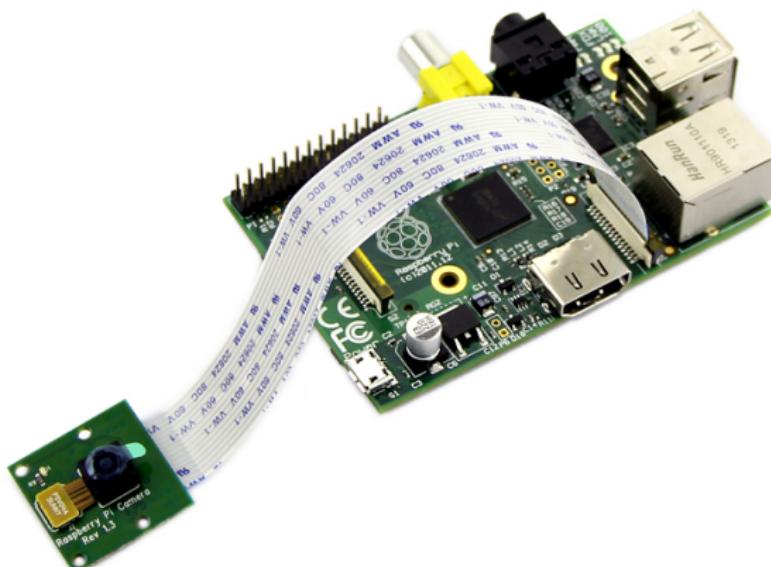


# Image capture



# Image capture

- ▶ The chosen vision system: Raspberry Pi + Raspicam



[www.raspberrypi.org](http://www.raspberrypi.org)

# Image capture

## Why ?

- ▶ Raspberry Pi is a Linux mini computer
- ▶ managed to install powerful Python image processing and machine learning libraries we needed
- ▶ Raspicam is a 5 Mp color camera, fully controlled
- ▶ inexpensive,  $\sim \text{€}100$
- ▶ slow but can be optimized
- ▶ can do learning to measure in a desktop PC and just measure here

# Samples

We are provided 23 jackets as working samples, from which

- ▶ 16 normal color, style, size
- ▶ 7 less frequent

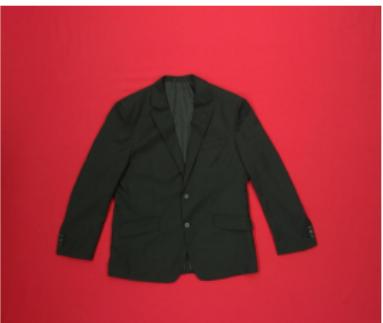
# Normal samples



1



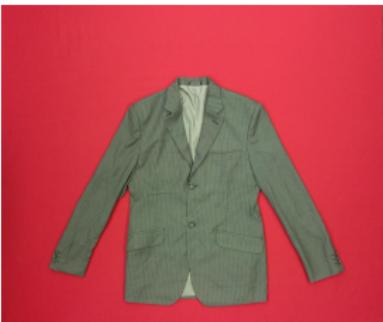
2



3



4

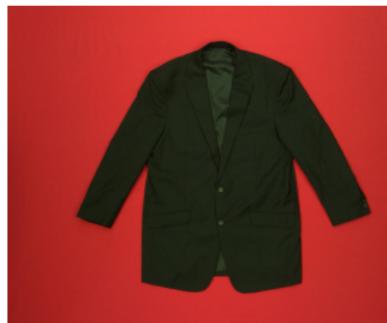


5

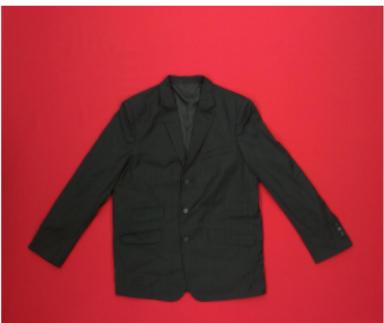


6

# Normal samples



7



8



9



10



11

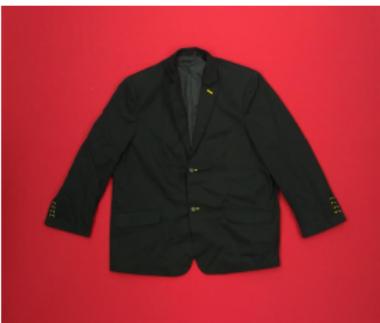


12

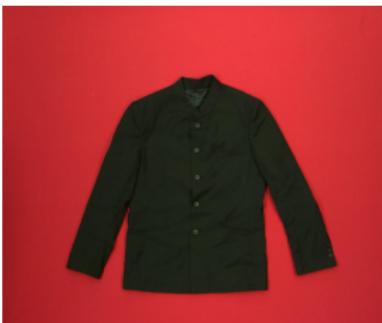
# Normal samples



13



14



15



16

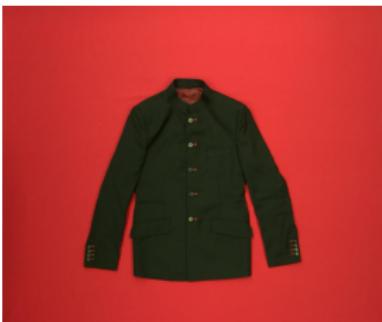
# Less frequent samples



17



18



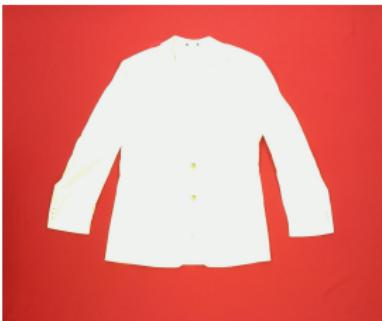
19



20



21



22

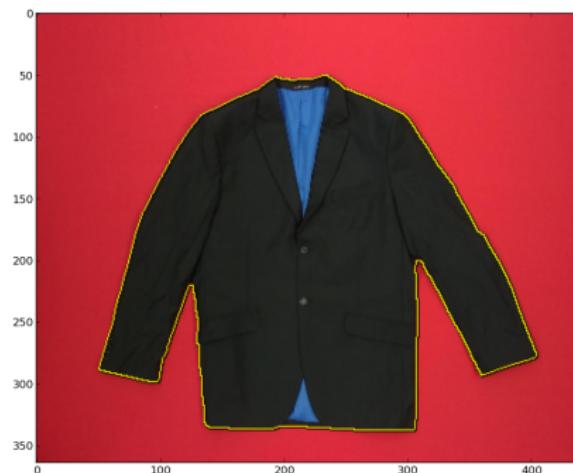
# Less frequent samples



23

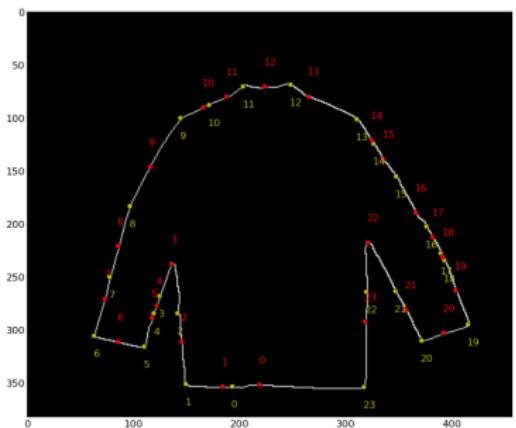
# Method

1. Segmentation
2. Extract contour

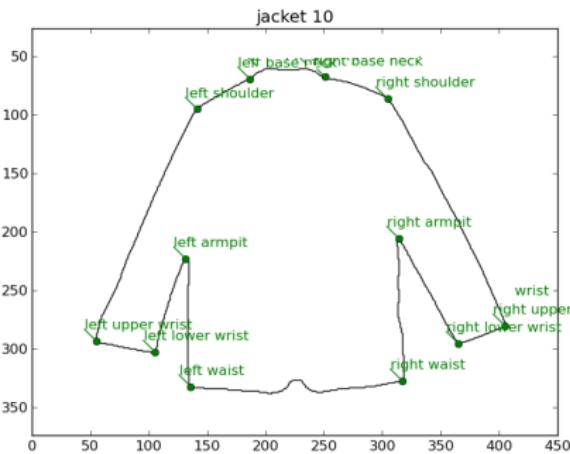


# Method

3. contour → curvature → extrema points → keypoints



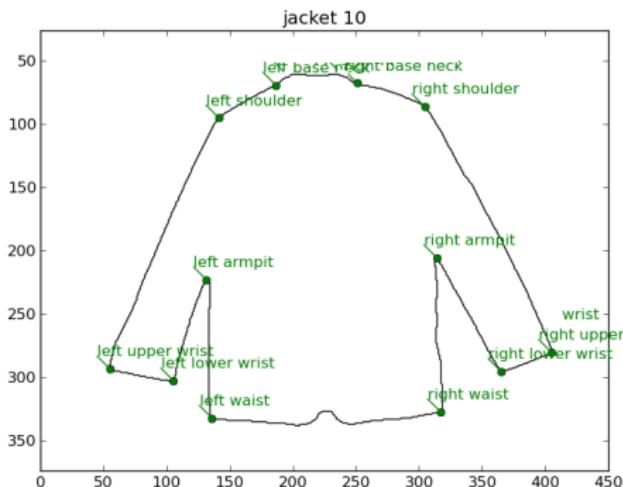
yellow = maxima / convex  
red = minima / concave



# Method

## 3. contour → curvature → extrema points → keypoints

```
labels_extrema = {  
    0: 'not a keypoint',  
    1: 'left base neck',  
    2: 'left shoulder',  
    3: 'left upper wrist',  
    4: 'left lower wrist',  
    5: 'left armpit',  
    6: 'left waist',  
    7: 'right waist',  
    8: 'right armpit',  
    9: 'right lower wrist',  
    10: 'right upper wrist',  
    11: 'right shoulder',  
    12: 'right base neck'  
}
```

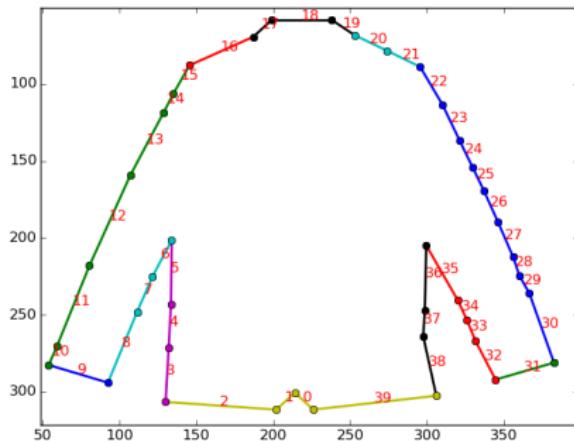


# Method

or

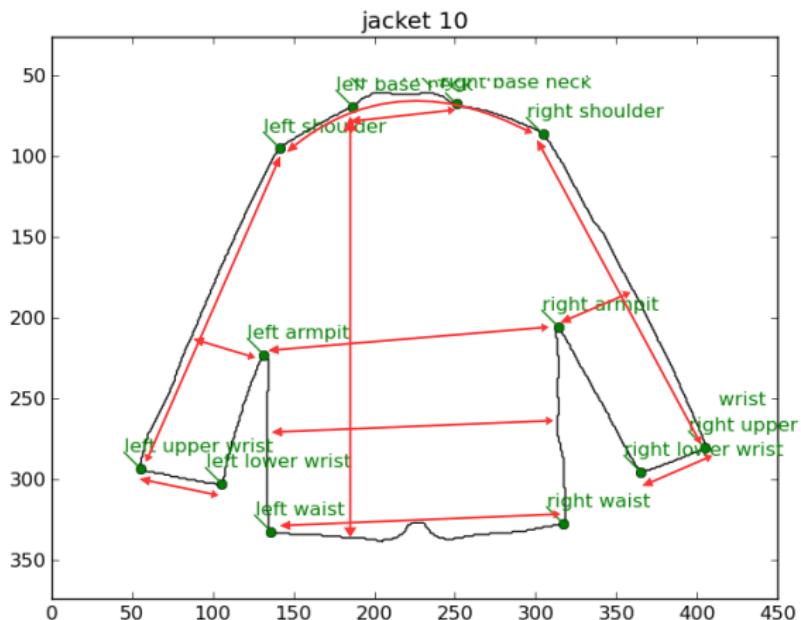
3. contour → curvature → extrema points → segments → keypoints

```
labels_segments = {  
    0: 'neck',  
    1: 'left shoulder',  
    2: 'outer left sleeve',  
    3: 'left wrist',  
    4: 'inner left sleeve',  
    5: 'left chest',  
    6: 'waist',  
    7: 'right chest',  
    8: 'inner right sleeve',  
    9: 'right wrist',  
    10: 'outer right sleeve',  
    11: 'right shoulder',  
}
```



# Method

## 4. keypoints → measures



# Method

What should we label = classify to then get the keypoints ?

- ▶ curvature extrema, or
- ▶ segments between curvature extrema

What has a more constraining context ? (see the labels in each case)

# What to do

Given

- ▶ segments plus some precomputed features
- ▶ groundtruth = type of each segment

for each jacket, **learn a segment classifier** (no need to obtain keypoints, it's direct)

# What to do

1. design a **graphical model** to infer the class of each segment, making the most of the structured nature of the problem
2. train it = **learn** its parameters, trying a few different ssvm learning algorithms
3. perform structured prediction as **inference** on it
4. **evaluation** with kfold=5 on jackets (23 for learning, 4 for testing)
5. compare performance with classification by standard linear SVM
6. **explain** results

# Tools, data & code

- ▶ Python + some IDE (for example Windows: Python( $x, y$ ),  
Windows/Linux: Anaconda)
- ▶ scikit-learn for linear SVM and cross-validation
- ▶ Panda or cvs to load Excel spreadsheets
- ▶ Pystruct + dependencies (CVXOPT, qpbo)
- ▶ Pystruct OCR letter recognition example !
- ▶ segments for 23 jackets, groundtruth and template script

# Grading

- ▶ C = 0 points : code does not work or performs bad, doesn't learn a GM
- ▶ B = 0.5 points : it works and performs well (like >95% accuracy), good discussion of results
- ▶ A = 1 point : B + comparison with linear SVM, adding noise and different sets of features, good discussion of results

# Deliverables

Each group sends to [joans@cvc.uab.es](mailto:joans@cvc.uab.es)

- ▶ the source code, that can or not be based on the template we provide, plus a 4 or 5 slides in PDF with
- ▶ quantitative + qualitative results
- ▶ discuss results : when does it work ? when not ? comparison
- ▶ all in a single zip file

Include names of group members in the email