Pattern Recognition Lab

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

# Visual Recognition for Humanities

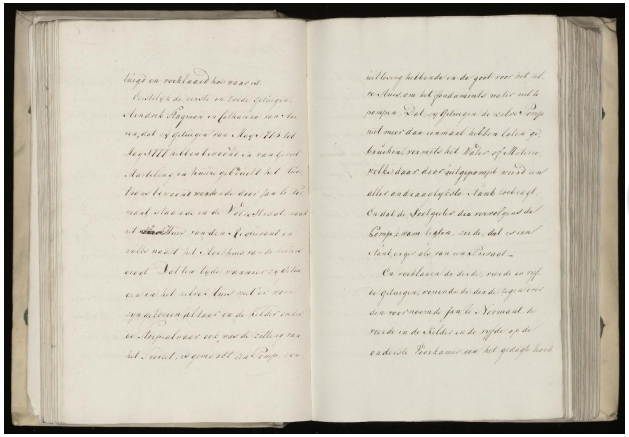
**Mathias Zinnen, M.Sc.,**

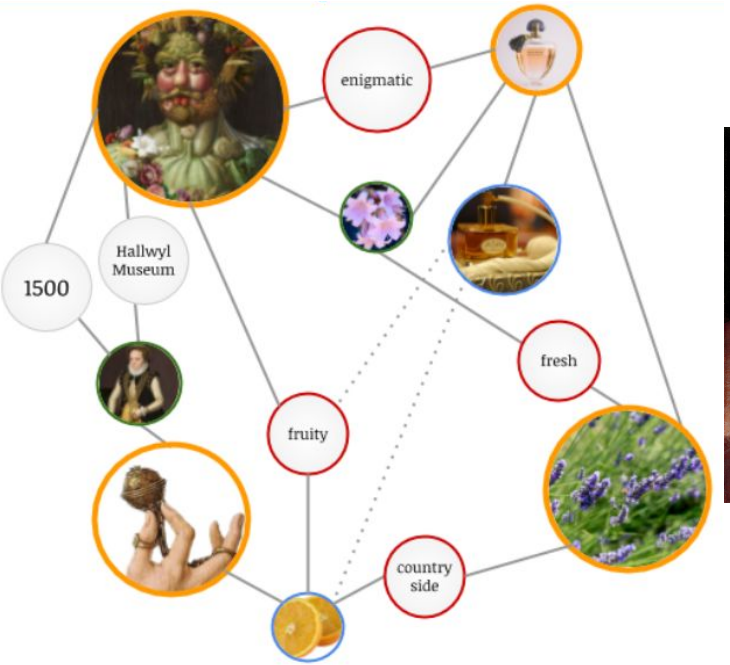
**Project CV Winter 2024/2025,**

# Outline

- Computer Vision and the Humanities
- Object Detection
- Selective search & Felzenszwalb
- Modern Object Detection
- Exercise

# Computer Vision and the Humanities






# Focus I - Text

## Text and Writer identification and OCR

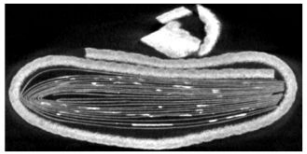
Α'πὸ τῆς Φυγῆς τοῦ Ξέρξου ἐκ τῆς Ἑλλάδος ἕως τῆς ἐν Μυκάλῃ νίκης.

Ἡ πρώτη δὲ ἄμεσος τῶν Ἑλλήνων φροντὶς μετὰ
πτ. κ. τὴν ἐν Σαλαμῖνι μάχην, ἦτο τὸ νὰ ἐκδιώξωσιν εἰς τὰς Δελ-
3524. φοὺς τὰς ἀπαρχὰς τῶν πλουσίων λαφύρων, ὅσα ἔλαβον ἀπὸ τὰς Πέρσας. Ὡς σύμμαχοι θεωρούμενοι οἱ Ἕλληνες ἦσαν πάντοτε προσεκτικοὶ εἰς τὰ καθήκοντα τῆς θρησκείας, ἐξ οὗ ὅλον ὅτι αἱ αἱρέσεις καὶ τὰ δόγματα τῶν φιλοσόφων ἐδίδασκον τὸ ἀνθρώπινον γένος ὀλίγου νὰ τιμᾷ τὰς δημοσίας τελετάς, ὡς τόσον ἡ θρησκεία ἦτον ὁ πρῶτος καὶ μόνος σύνδεσμος τῆς ἑνώσεως αὐτῶν, καὶ ἐκράτει αὐτὰς πρὸς ὥραν ἀδυνάτως ἡνωμένους· ὅτε δὲ ἐλύθη αὐτὸς ὁ δεσμός, καὶ ἡ Ἀμφικτυονικὴ σύνοδος ἔγινε πολιτικὴ μᾶλλον ἢ θεοκρατικὴ σύνοδος, διελύθη καὶ ἡ γενικὴ ἕνωσις, καὶ πολλαὶ ἐπαρχίαι ἔγιναν θῦμα τῶν ἐμφυλίων πολέμων.

## Handwriting imitation

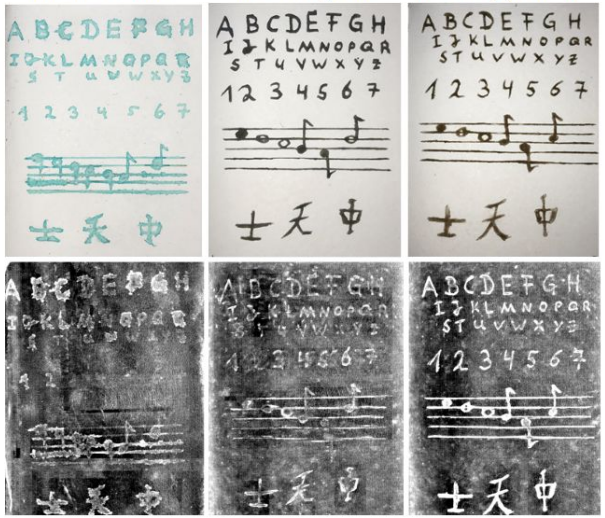
# Focus II - Book CT [1]

Daniel Stromer



## Results

Malachit | Iron gall | Iron gall +

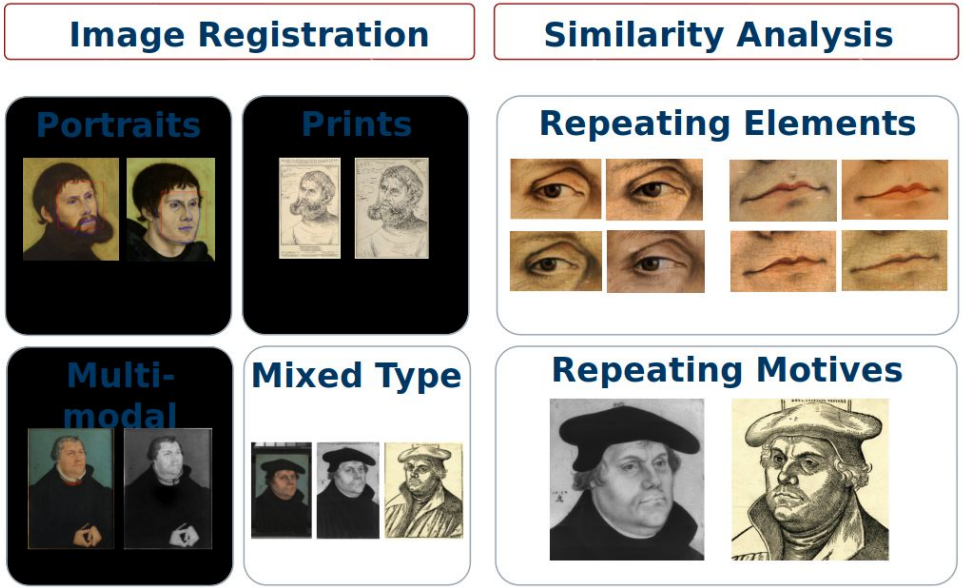


[1] Stromer, Daniel; Christlein, Vincent; Huang, Xiaolin; Zippert, Patrick; Hausotte, Tino; Maier, Andreas Virtual cleaning and unwrapping of non-invasively digitized soiled bamboo scrolls Scientific Reports, vol. 9, no. 1, pp. 2311, 2019

# Focus III - Similarity [2]

Aline Sindel

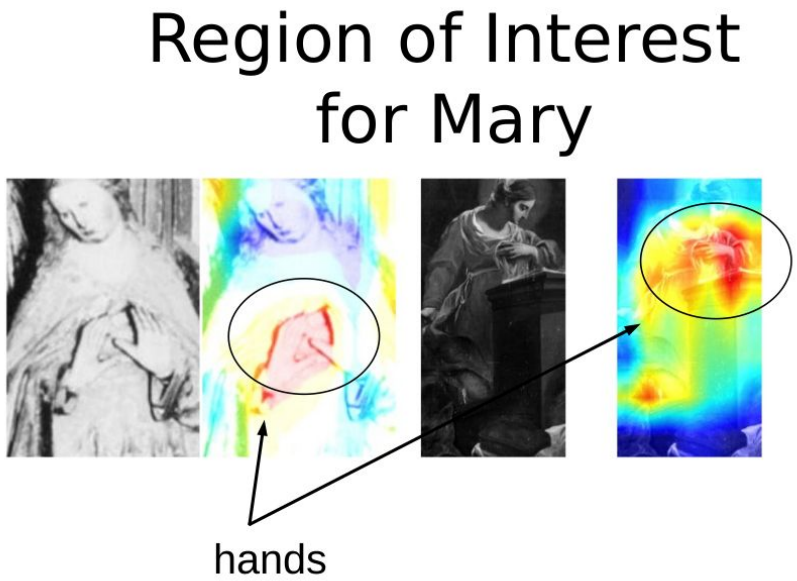
GERMANISCHES NATIONAL MUSEUM




[2] Project page : https://www.gnm.de/forschung/archiv/luther-bildnisse/

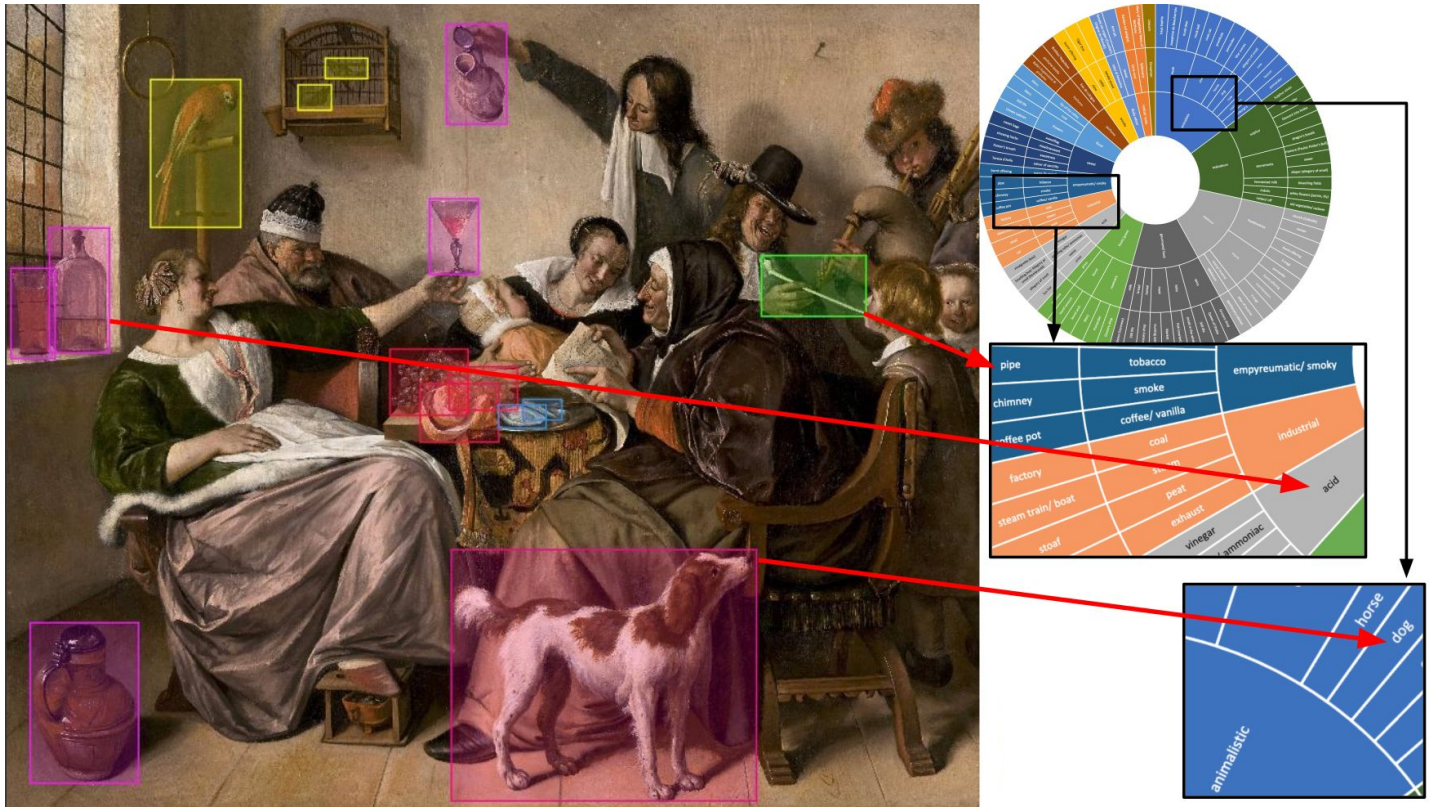
# Focus IV - Understanding [3]

Ronak Kosti




[3] Madhu, Prathmesh, et al. "Recognizing Characters in Art History Using Deep Learning." *Proceedings of the 1st Workshop on Structuring and Understanding of Multimedia heritAge Contents*. 2019.

# Focus V - Digital Heritage




Odeuropa

## Art History

## Classical Arch.

## Christian Arch.

# Visual recognition?

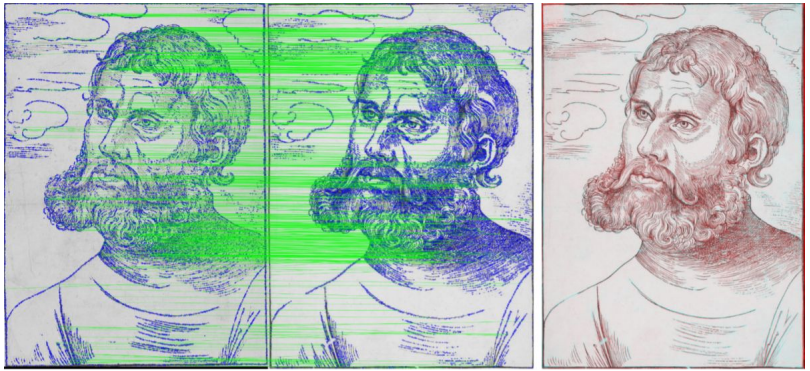
# Visual recognition?

# Visual recognition?

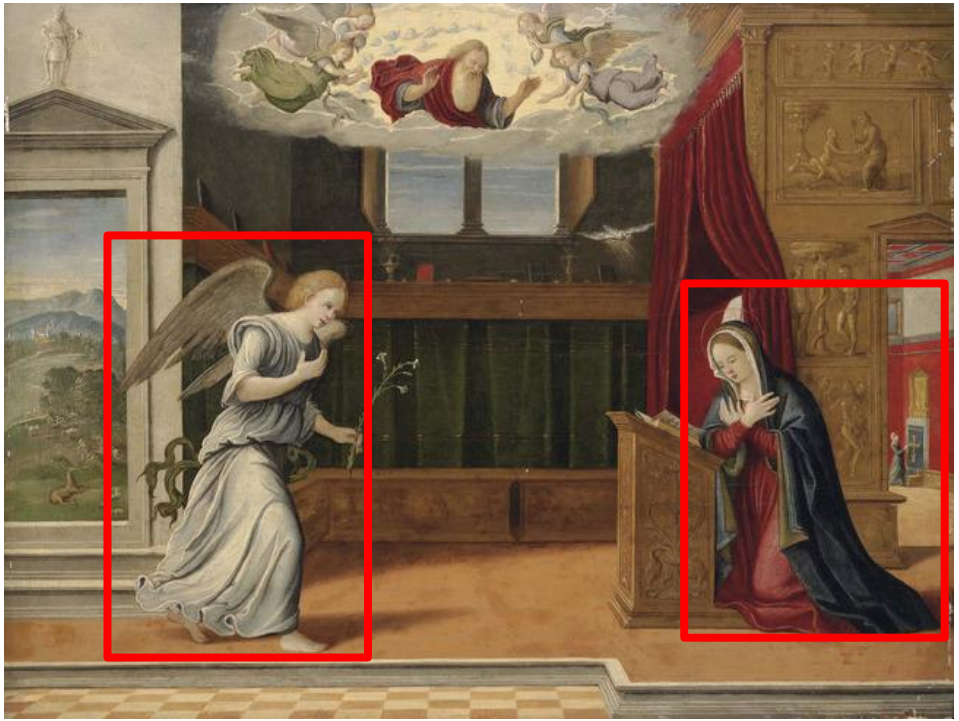
# Visual recognition?

# Visual recognition?

- Classification
- Object Detection
- Semantic Segmentation
- Instance Segmentation
- Key Point Detection
- VQA



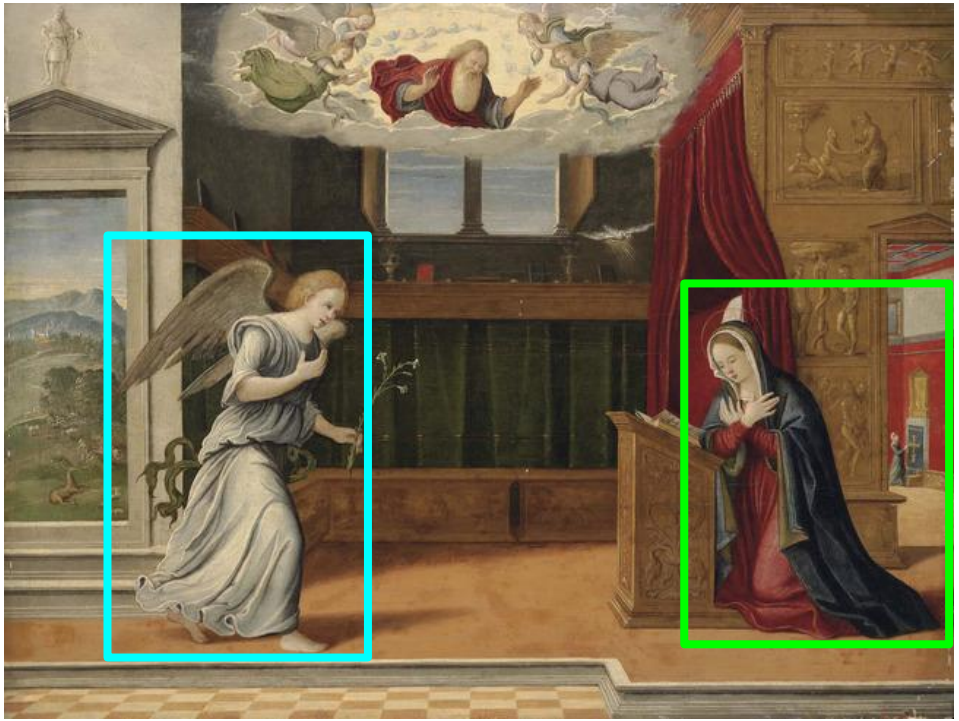
# Visual recognition?



**Category-level recognition**

**Person**

# Visual recognition?



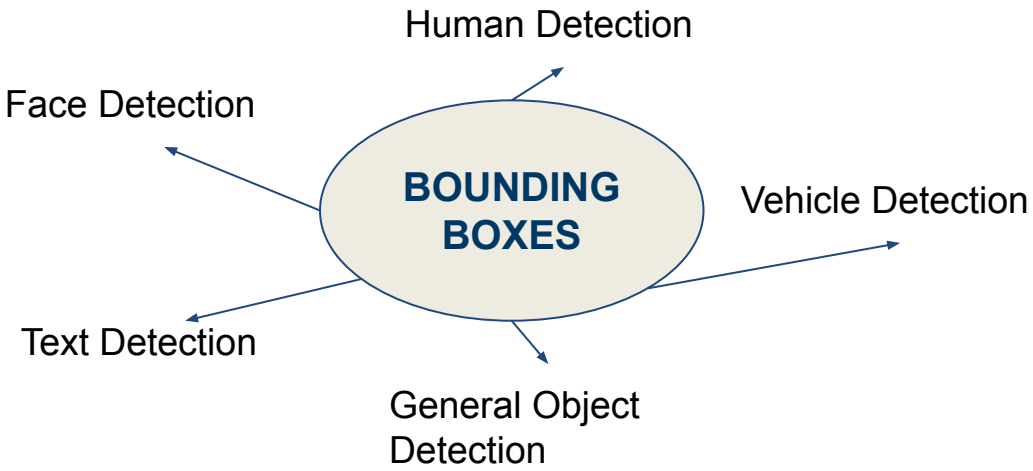
**Instance-level recognition**

**Mary**

**Gabriel**

# How to quantify objects?

- Encapsulating objects → a common representation



# Quantifying objects - Applications

- Cancer detection in radiology-based images [Healthcare]
- Autonomous Driving [Automation]
- Detection of manufacturing defects, factory floor surveillance [Manufacturing]
- Pedestrian Detection [Public safety and surveillance]

… and many more ...

# Modern Object Detection

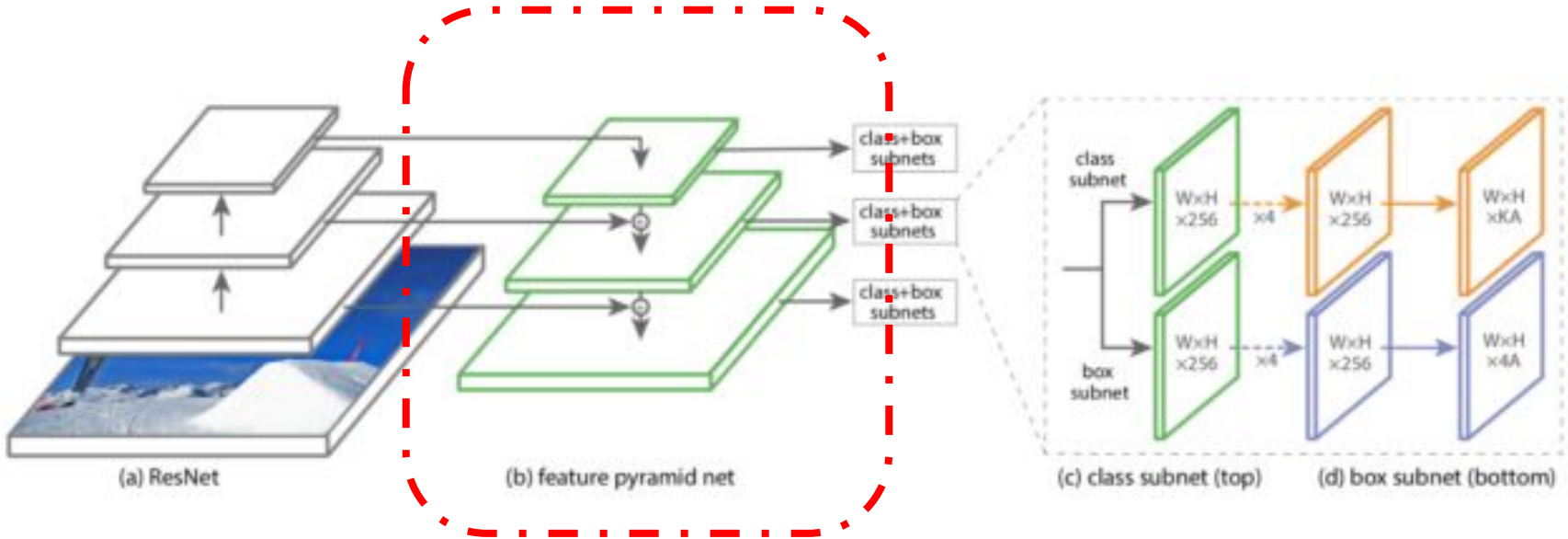



Figure - RetinaNet Architecture

# Challenges

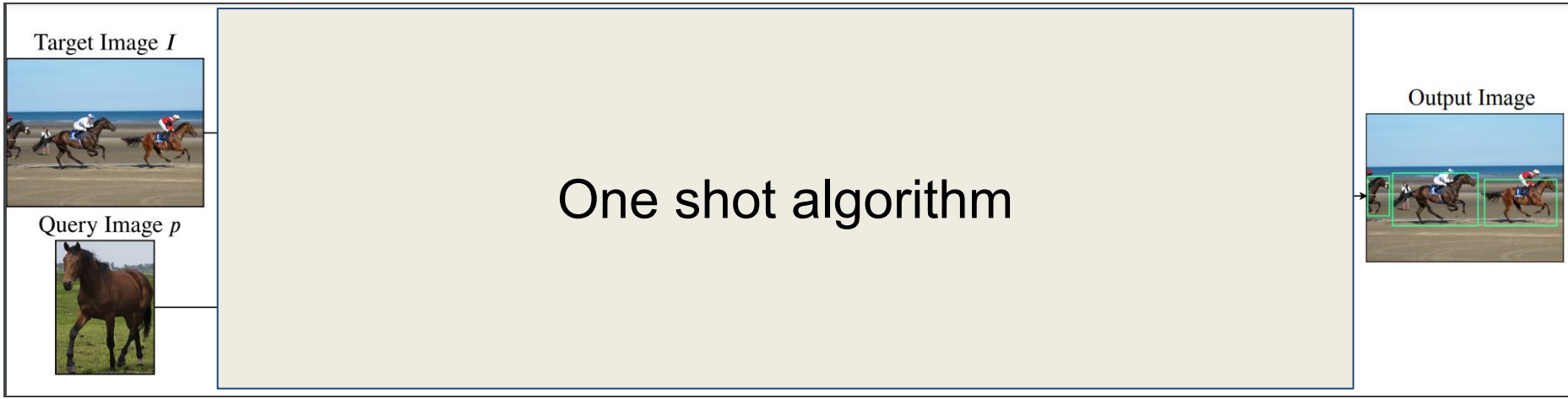
- Existing methods. Domain shift?
- Amount of Annotations
- What about new classes?

# Challenges

- Existing methods. Domain shift?
- Amount of Annotations
- What about new classes?

***N-Way K-Shot Object Detection***

# One-Shot Object Detection




- Train/Seen classes – [Horses, hats, cats, donkeys]
- Test/Unseen classes – [Persons, dogs]

# What do these deep features encode?

OR

How do these networks define “***objectness***”?

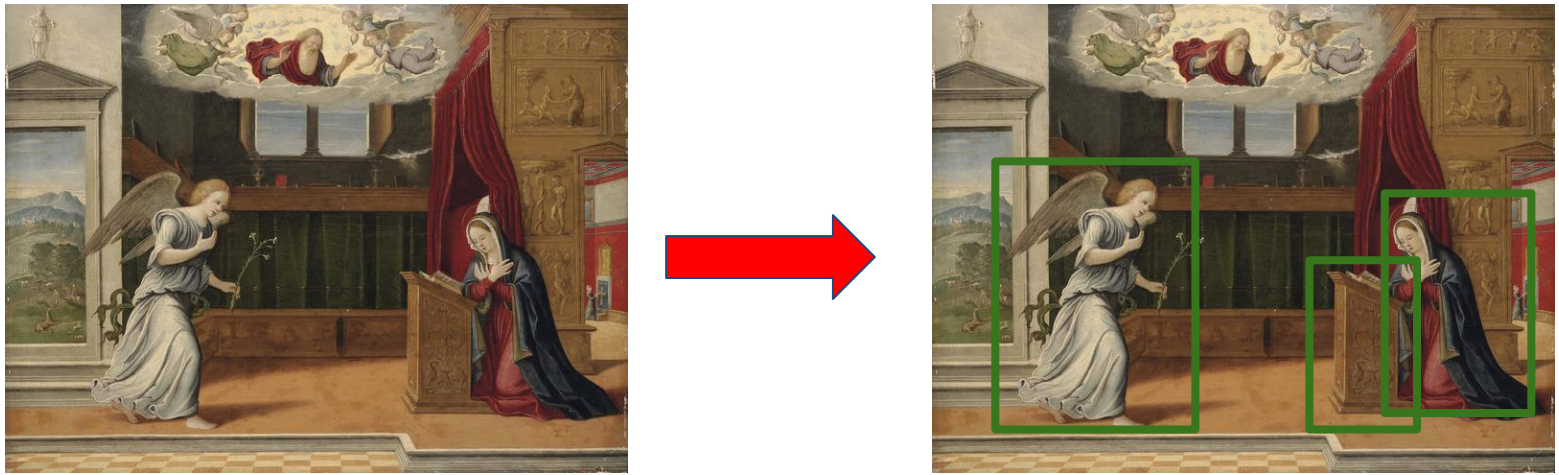
**“Selective Search for object recognition”**

# Object recognition

Find objects and recognize them

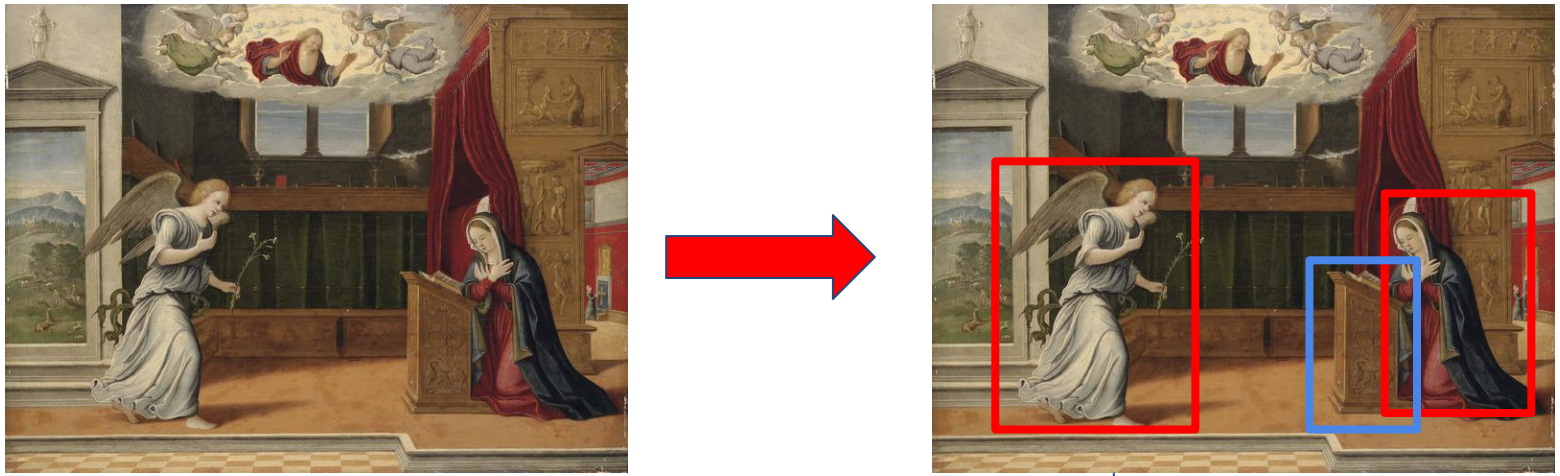
# Object recognition

## Find objects and recognize them

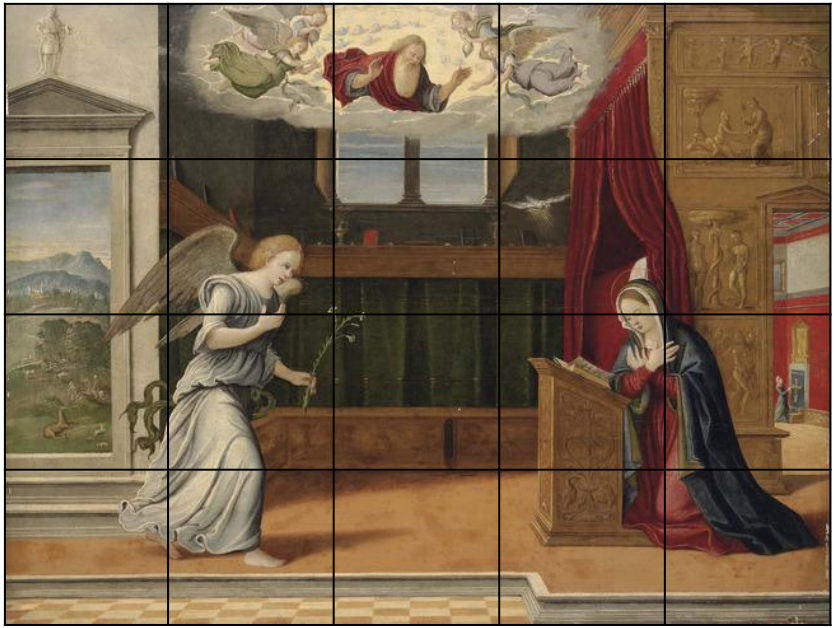


# Object recognition

**Find** objects and *recognize* them

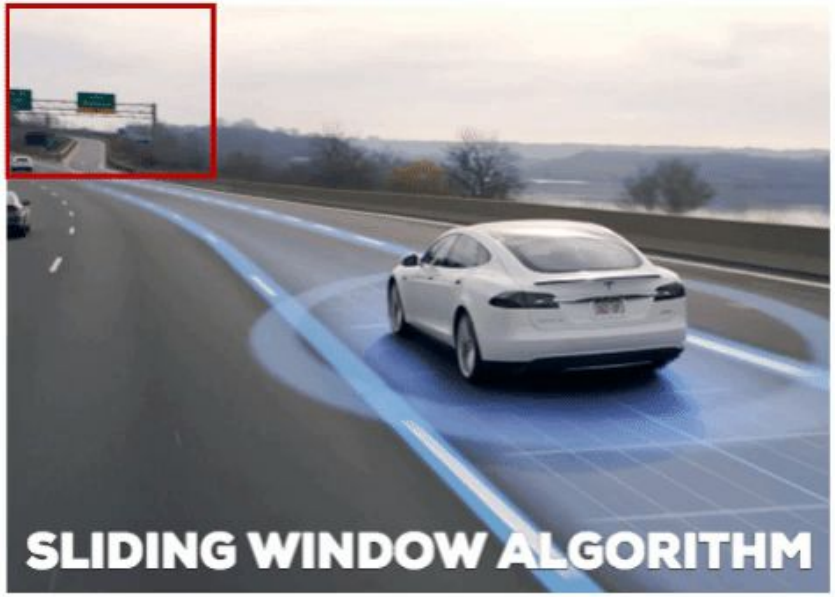


# Exhaustive search



# Sliding Window Object Detector

- Localizing → Identifying objects
- $F_i$ : fixed size windows
- $f(F_i) \rightarrow$ ? (which class), $f$ : classifier




https://medium.com/@rajshekhar_k/object-detection-series-3-0-basic-of-object-detection-196d91550671

# Computationally effective?

\+ Trained classification network can be used for object detection directly

− Computationally inefficient: One pass through the network for every patch!

Solution - Improve speed by using ***Fully Convolutional Kernels***

Still we would need ***regions of interest*** right and that too in an automated fashion maybe?

# Selective Search

- Goals:
  - Capture all scales - How could we know the size of object?
  - Diversifications - Different criteria for segmentation
  - Fast to compute
- use of the powerful ***Bag-of-Words model for recognition*** [Further Reading]

# Approach

- Hierarchical segmentation
  - Apply existing algorithms to find sub-segmentations
    - Small segmentations
  - Recursively combine small segmentations into big segmentations
    - Big segmentations

# Algorithm

**Algorithm 1:** Hierarchical Grouping Algorithm

**Input**: (colour) image
**Output**: Set of object location hypotheses $L$

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]
Initialise similarity set $S = \emptyset$
**foreach** *Neighbouring region pair* $(r_i, r_j)$ **do**
    Calculate similarity $s(r_i, r_j)$
    $S = S \cup s(r_i, r_j)$

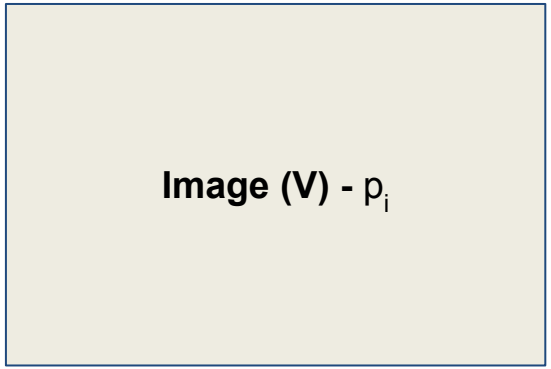
**while** $S \neq \emptyset$ **do**
    Get highest similarity $s(r_i, r_j) = \max(S)$
    Merge corresponding regions $r_t = r_i \cup r_j$
    Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$
    Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$
    Calculate similarity set $S_t$ between $r_t$ and its neighbours
    $S = S \cup S_t$
    $R = R \cup r_t$

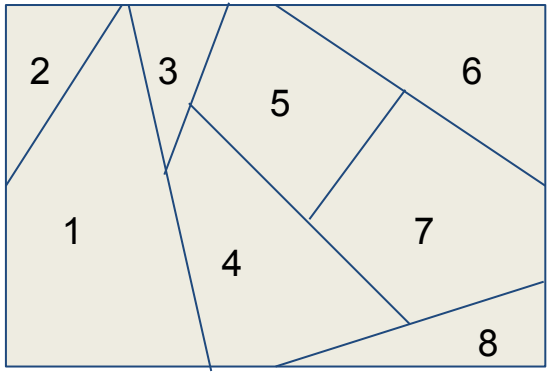
Extract object location boxes $L$ from all regions in $R$

Felzenszwalb paper

# Felzenszwalb - in a nutshell

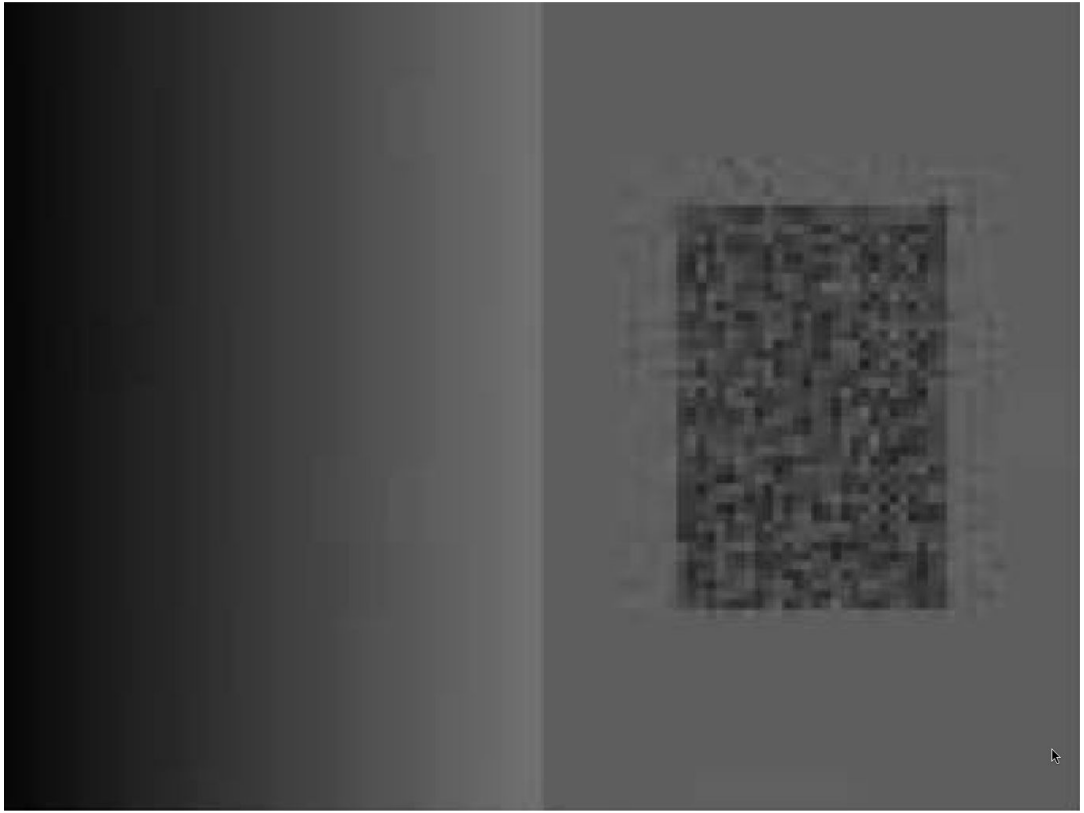
- Divide an image into regions
- Defining boundaries by graph-representation satisfying global properties
- Perceptual grouping - psychology
- Idea
  - Capture perceptually important groupings
  - Time efficient - real time applications
- Method - **Pairwise Region Comparison Predicate**



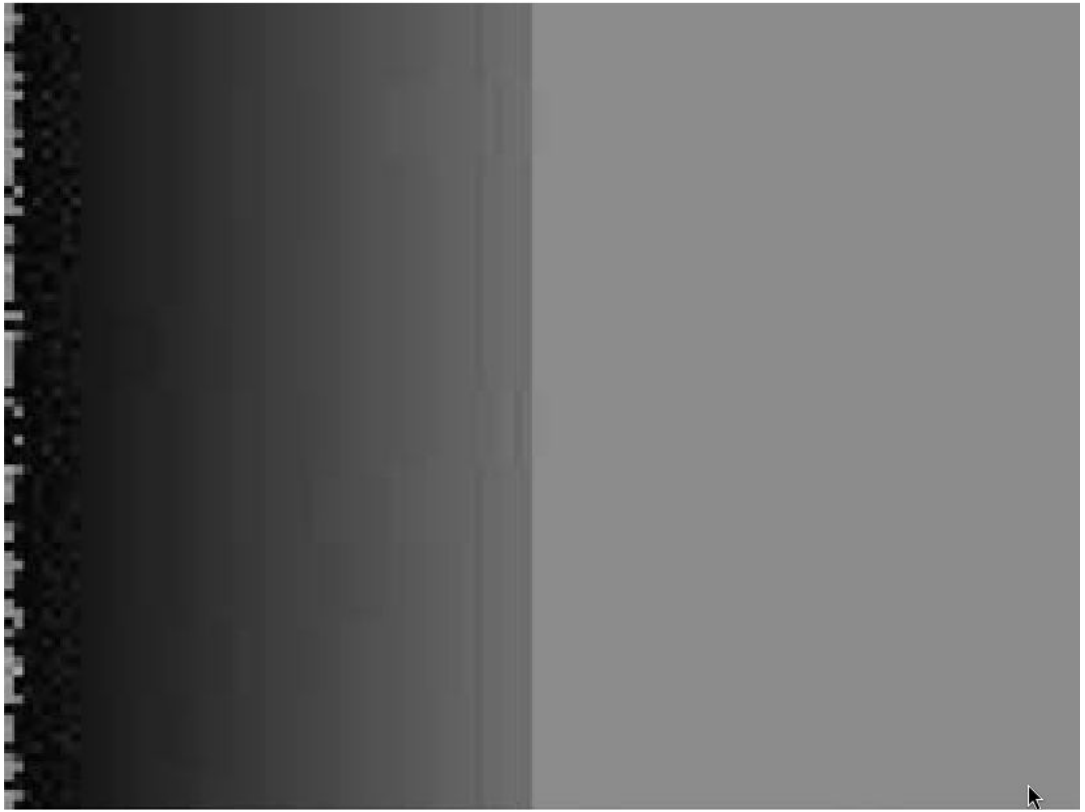
# Understanding segmentation

- How many regions?
- How to distinguish?



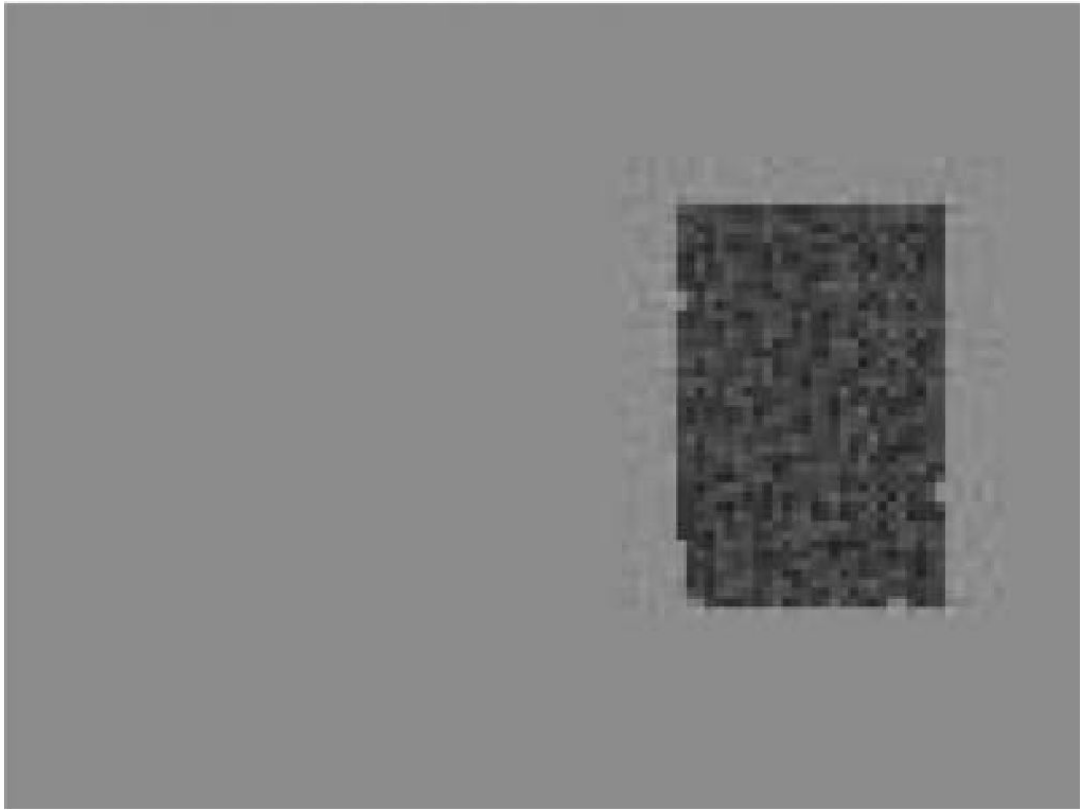
# Understanding segmentation

- Possible Criteria
  - Constant Intensities



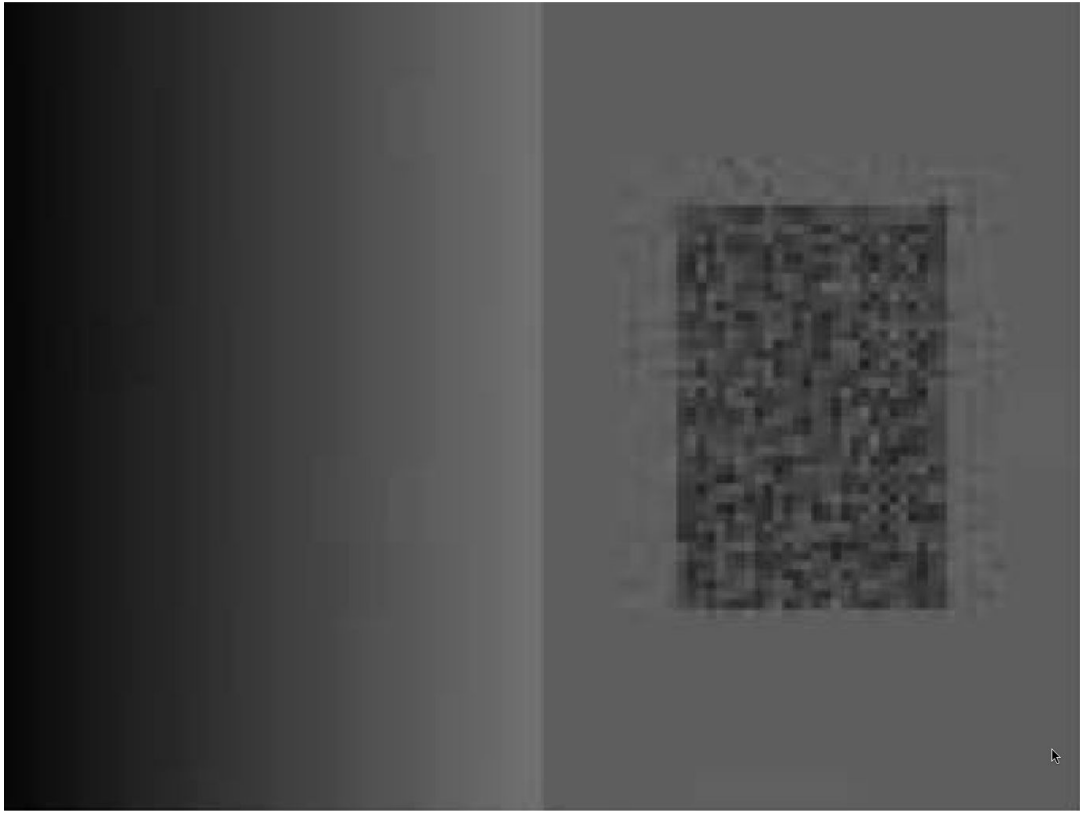
# Understanding segmentation

- Possible Criteria
  - Constant Intensities
  - Slowly Varying Intensities



# Understanding segmentation

- Possible Criteria
  - Constant Intensities
  - Slowly Varying Intensities
  - We need a non-local criterion



# Pairwise Region Comparison Predicate

- Compares inter-component difference to within component differences
- Or formally:

$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{false} & \text{otherwise} \end{cases}$$

# Inter-Class Difference

- Minimum difference on the border between two components
- Or formally:

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w((v_i, v_j)) \ .$$

# Intra-Class Difference

- Weighted maximum of internal differences
- Require stronger evidence for boundary for small components
- Or formally:

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1),\ Int(C_2) + \tau(C_2)).$$

$$Int(C) = \max_{e \in MST(C,E)} w(e)\ . \qquad \tau(C) = k/|C|$$

# Felzenszwalb - Algorithm

The input is a graph G = (V, E), with n vertices and m edges. The output is a segmentation of V into components S = ($C_1$,... , $C_r$).

0. Sort *E* into $\pi = (o_1, \ldots, o_m)$, by non-decreasing edge weight.

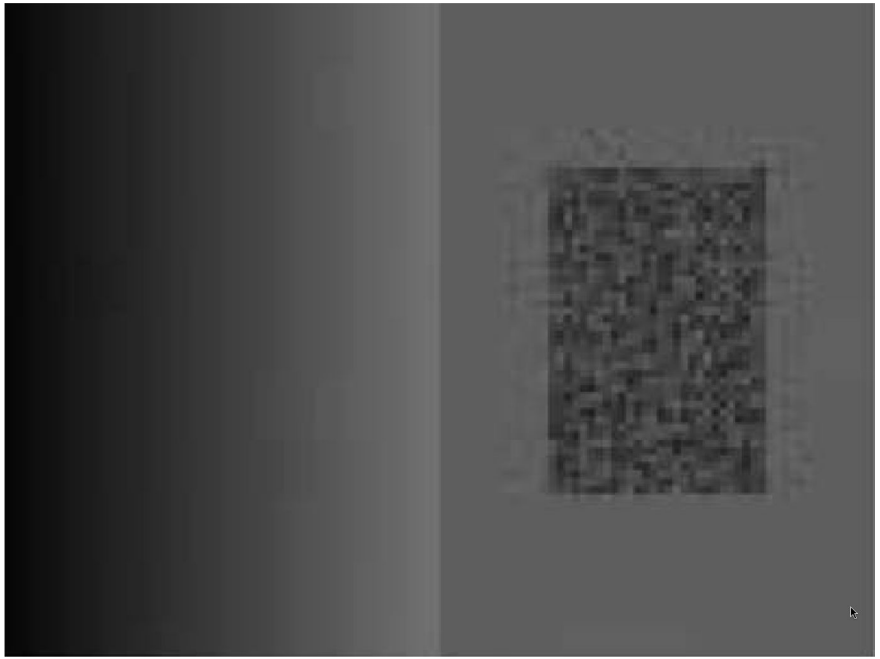
1. Start with a segmentation $S^0$ , where each vertex $v_i$ is in its own component.

2. Repeat step 3 for q = 1, . . . , *m*.

3. Construct $S^q$ given $S^{q-1}$ as follows. Let $v_i$ and $v_j$ denote the vertices connected by the q-th edge in the ordering, i.e., $o_q = (v_i, v_j)$. If $v_i$ and $v_j$ are in disjoint components of $S_{q-1}$ and $w(o_q)$ is small compared to the internal difference of both those components, then merge the two components otherwise do nothing.
More formally, let $C_i^{q-1}$ be the component of $S^{q-1}$ containing $v_i$ and $C_j^{q-1}$ the component containing $v_j$. If $C_i^{q-1}$ != $C_j^{q-1}$ and $w(o_q) \leq MInt(C_i^{q-1}, C_j^{q-1})$ then $S^q$ is obtained from $S^{q-1}$ by merging $C_i^{q-1}$ and $C_j^{q-1}$. Otherwise $S^q = S^{q-1}$.

4. Return $S = S^m$.

# Felzenszwalb - Results I



# Felzenszwalb - Results II

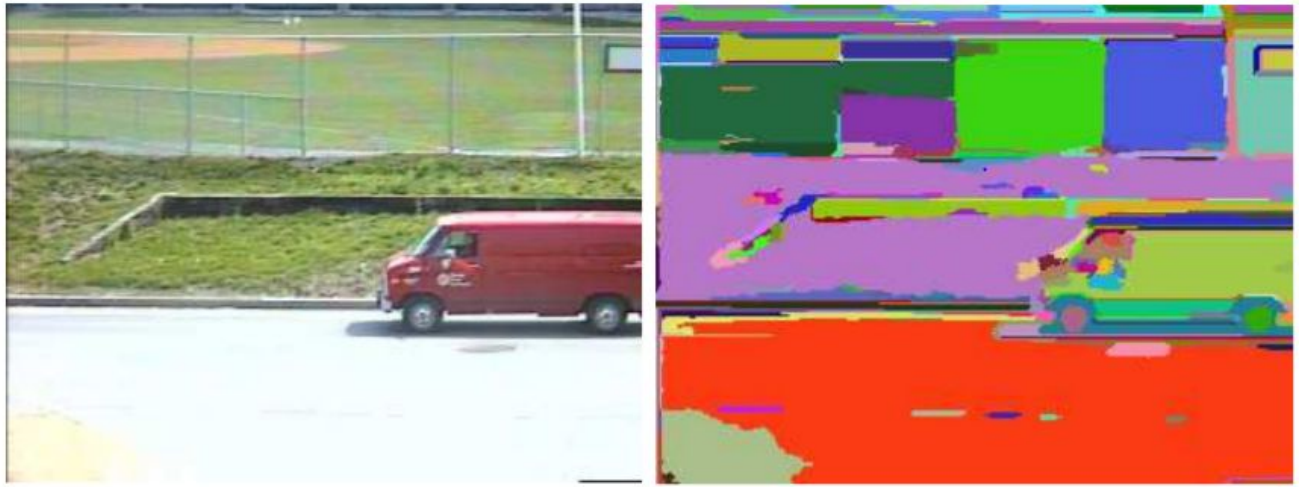
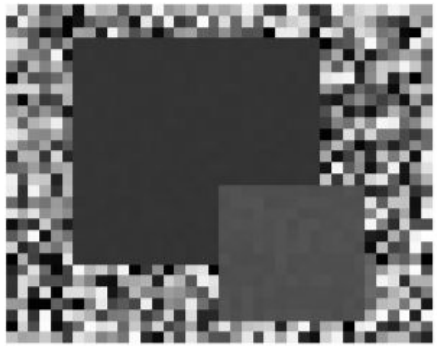
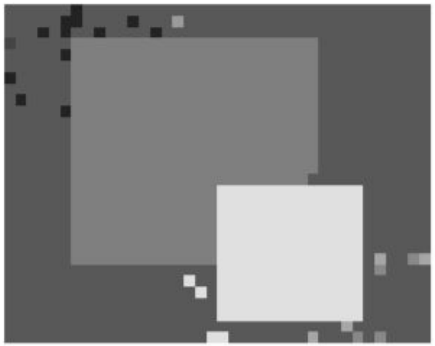


Figure 2: A street scene ($320 \times 240$ color image), and the segmentation results produced by our algorithm ($\sigma = 0.8$, $k = 300$).

# Felzenszwalb - Result III







# Algorithm

**Algorithm 1:** Hierarchical Grouping Algorithm

**Input**: (colour) image
**Output**: Set of object location hypotheses $L$

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]
Initialise similarity set $S = \emptyset$
**foreach** *Neighbouring region pair* $(r_i, r_j)$ **do**
    Calculate similarity $s(r_i, r_j)$
    $S = S \cup s(r_i, r_j)$

**while** $S \neq \emptyset$ **do**
    Get highest similarity $s(r_i, r_j) = \max(S)$
    Merge corresponding regions $r_t = r_i \cup r_j$
    Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$
    Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$
    Calculate similarity set $S_t$ between $r_t$ and its neighbours
    $S = S \cup S_t$
    $R = R \cup r_t$

Extract object location boxes $L$ from all regions in $R$

# Diversification

- ***How*** to model different criteria into the algorithm?
- ***What criteria*** for combining the segmentations?

(a) (b)

(c) (d)

# Diversification

- Complementary Color Spaces

(a) (b)

(c) (d)

# Diversification

- Complementary Color Spaces

- Complementary Similarity Measures
  - Color similarity
  - Texture similarity
  - Size similarity
  - Shape compatibility - shape fill

(a) (b)

(c) (d)

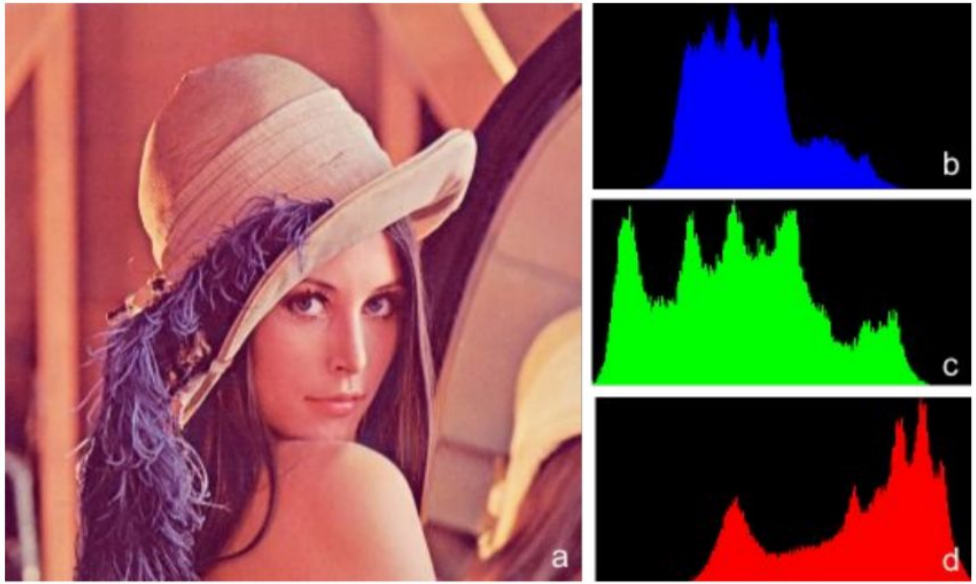
# Color similarity

$$C_i = \{c_i^1, \cdots, c_i^n\}$$

$$s_{colour}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k)$$

$$C_t = \frac{\text{size}(r_i) \times C_i + \text{size}(r_j) \times C_j}{\text{size}(r_i) + \text{size}(\text{r}_\text{j})}$$

$$\text{size}(r_t) = \text{size}(r_i) + \text{size}(r_j)$$



# Texture similarity

- Extract derivatives in 8 directions for 3 channels
- 10 bins for each, 240 bins in total
- L1 Normalize
- Histogram Intersection

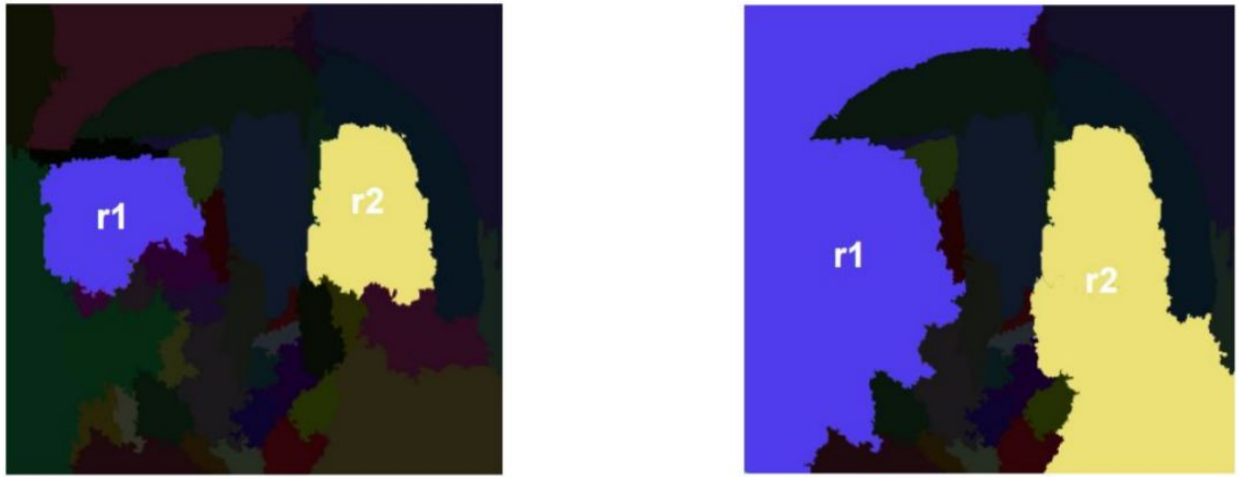
$$T_i = \{t_i^1, \cdots, t_i^n\}$$

$$s_{texture}(r_i, r_j) = \sum_{k=1}^{n} \min(t_i^k, t_j^k)$$

# Size similarity

- We hope to **merge two small region** into a large segmentation

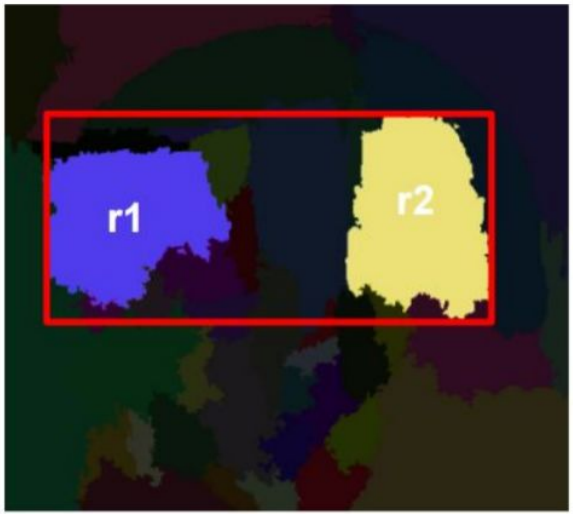
$$s_{size}(r_i, r_j) = 1 - \frac{\text{size}(r_i) + \text{size}(\text{r}_\text{j})}{\text{size}(im)}$$

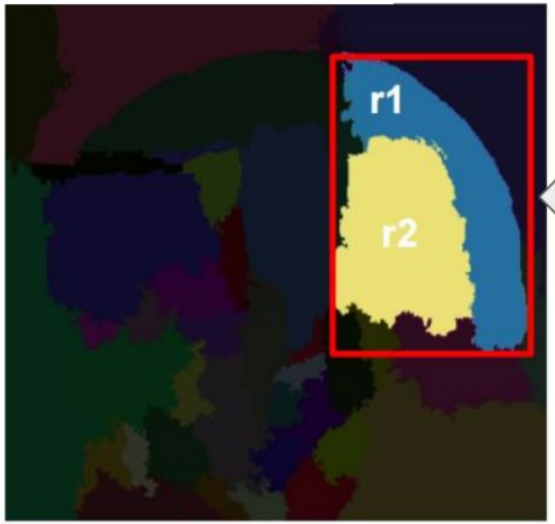


# Shape compatibility

- Whether two segmentations fit each other?

$$fill(r_i, r_j) = 1 - \frac{\text{size}(BB_{ij}) - \text{size}(r_i) - \text{size}(r_j)}{\text{size}(im)}$$




**B**ounding
**B**ox

# Diversification (contd.)

- Weighted mixture approach

$$s(r_i, r_j) = a_1 s_{colour}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j),$$

# Diversification

- Complementary Color Spaces
- Complementary Similarity Measures
  - Color similarity
  - Texture similarity
  - Size similarity
  - Shape compatibility - shape fill
- Complementary Starting Regions

# Evaluation

- **Average Best Overlap (ABO)**

$$\text{ABO} = \frac{1}{|G^c|} \sum_{g_i^c \in G^c} \max_{l_j \in L} \text{Overlap}(g_i^c, l_j)$$

Overlap between ground truth and best selected box.

Average of "best overlaps" across all images.

$$\text{Overlap}(g_i^c, l_j) = \frac{\text{area}(g_i^c) \cap \text{area}(\text{l}_\text{j})}{\text{area}(g_i^c) \cup \text{area}(\text{l}_\text{j})}$$

# Evaluation

| Similarities | MABO | # box |
|---|---|---|
| C | 0.635 | 356 |
| T | 0.581 | 303 |
| S | 0.640 | 466 |
| F | 0.634 | 449 |
| C+T | 0.635 | 346 |
| C+S | 0.660 | 383 |
| C+F | 0.660 | 389 |
| T+S | 0.650 | 406 |
| T+F | 0.638 | 400 |
| S+F | 0.638 | 449 |
| C+T+S | 0.662 | 377 |
| C+T+F | 0.659 | 381 |
| C+S+F | 0.674 | 401 |
| T+S+F | 0.655 | 427 |
| C+T+S+F | 0.676 | 395 |

| Colours | MABO | # box |
|---|---|---|
| HSV | 0.693 | 463 |
| I | 0.670 | 399 |
| RGB | 0.676 | 395 |
| rgI | 0.693 | 362 |
| Lab | 0.690 | 328 |
| H | 0.644 | 322 |
| rgb | 0.647 | 207 |
| C | 0.615 | 125 |

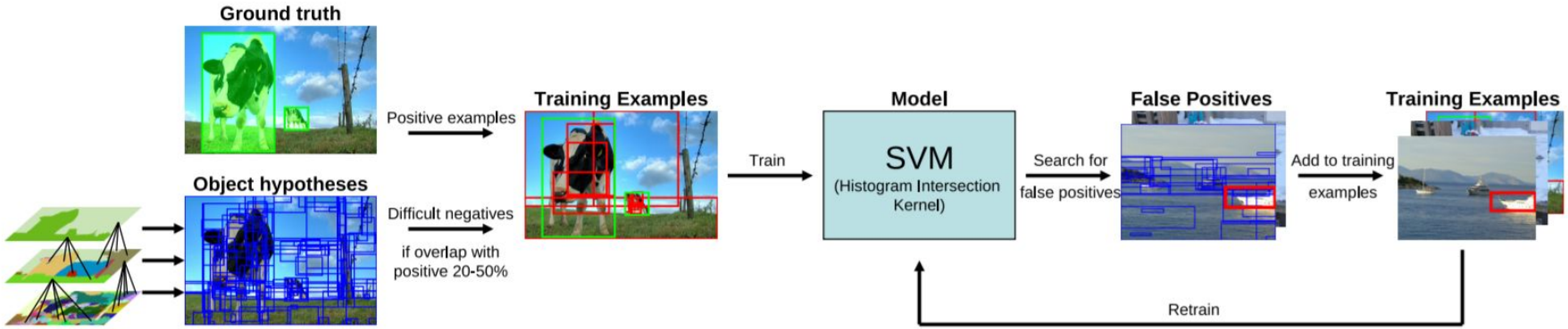
| Thresholds | MABO | # box |
|---|---|---|
| 50 | 0.676 | 395 |
| 100 | 0.671 | 239 |
| 150 | 0.668 | 168 |
| 250 | 0.647 | 102 |
| 500 | 0.585 | 46 |
| 1000 | 0.477 | 19 |

# Evaluation

| Version | Diversification Strategies | MABO | # win | # strategies | time (s) |
|---|---|---|---|---|---|
| Single Strategy | HSV<br>C+T+S+F<br>$k = 100$ | 0.693 | 362 | 1 | 0.71 |
| Selective Search Fast | HSV, Lab<br>C+T+S+F, T+S+F<br>$k = 50, 100$ | 0.799 | 2147 | 8 | 3.79 |
| Selective Search Quality | HSV, Lab, rgI, H, I<br>C+T+S+F, T+S+F, F, S<br>$k = 50, 100, 150, 300$ | 0.878 | 10,108 | 80 | 17.15 |

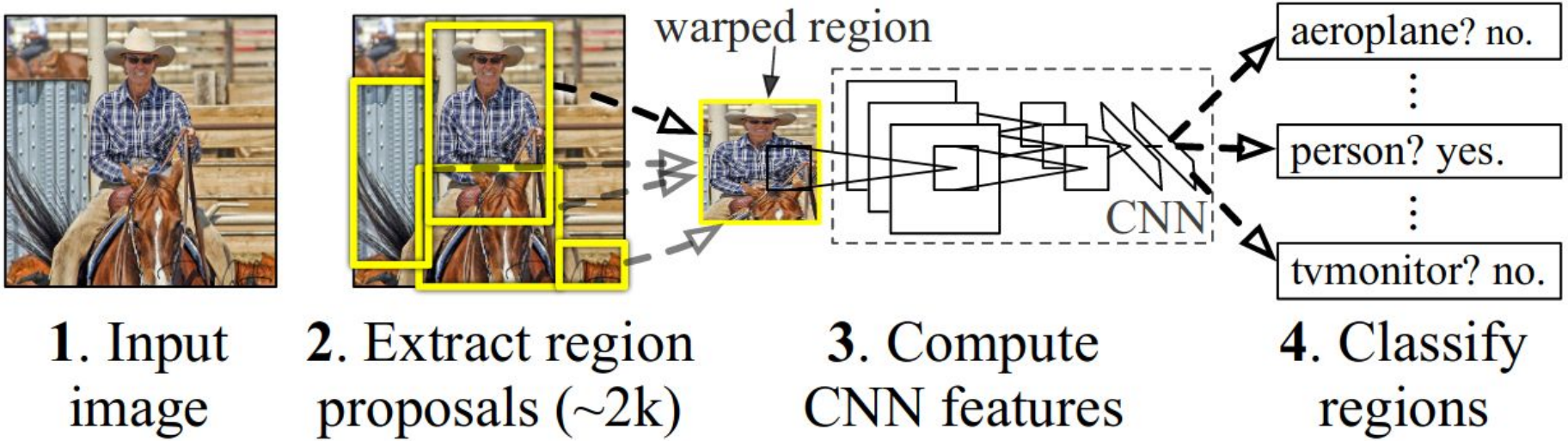
# Object Recognition

Approach : Selective Search + SIFT + SVM



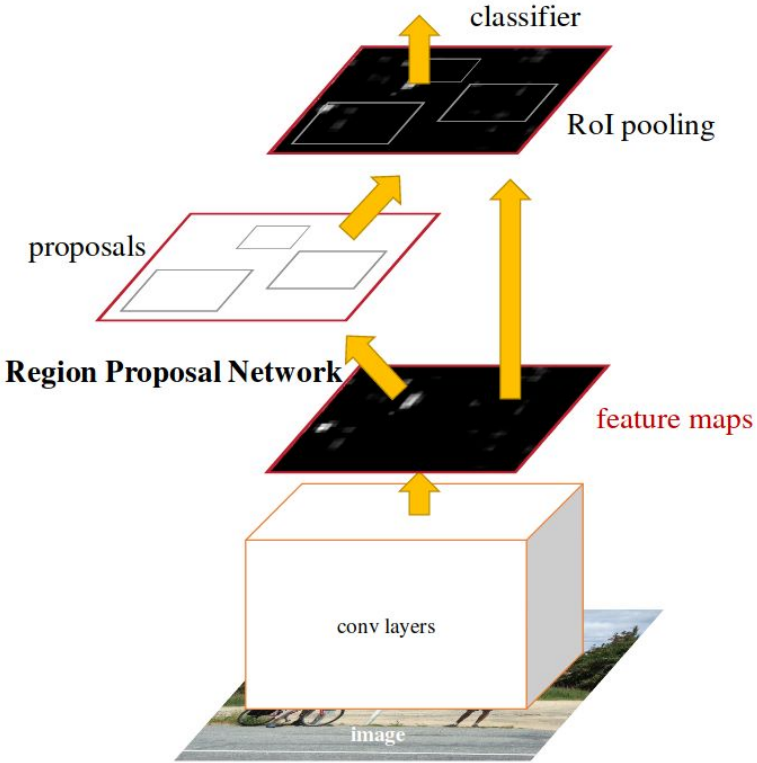

Uijings et al. Selective Search for Object Recognition

# Modern Object Detection



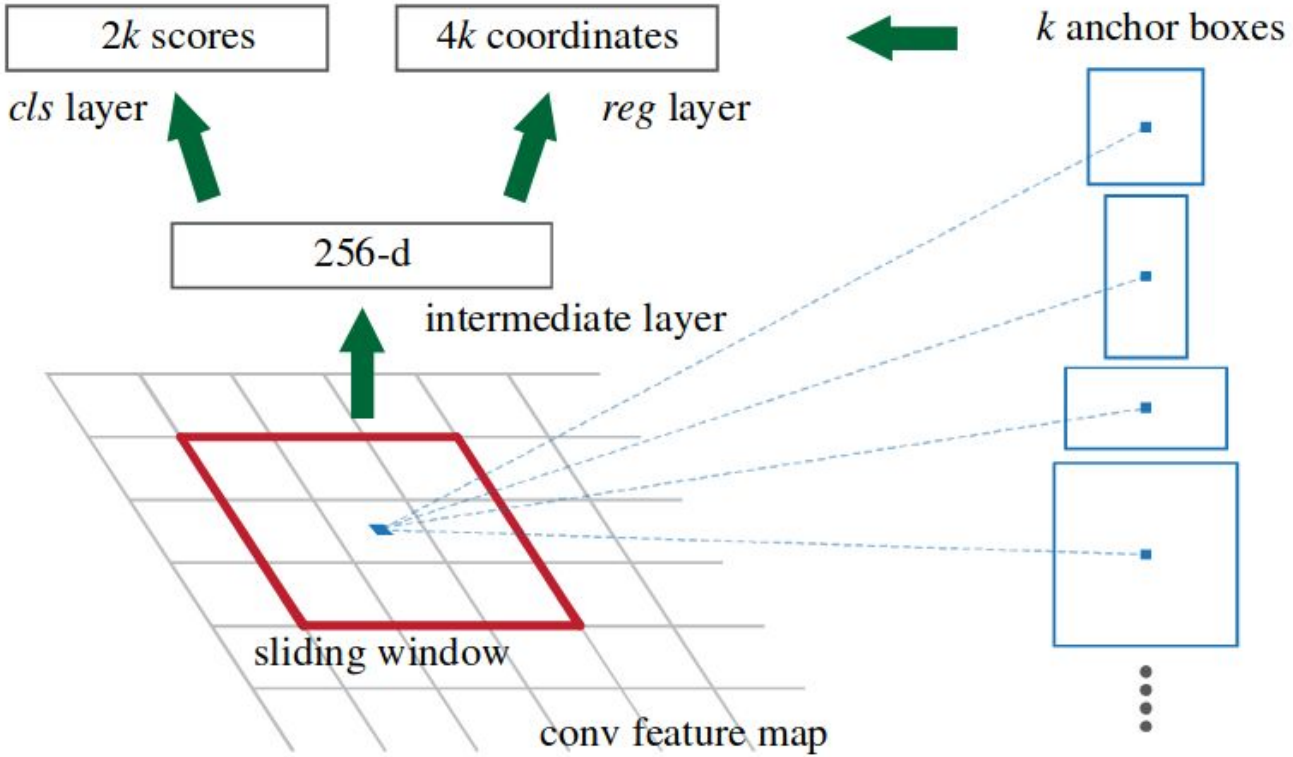

R-CNN Architecture

# From Selective Search to Deep Learning: RPNs [1]



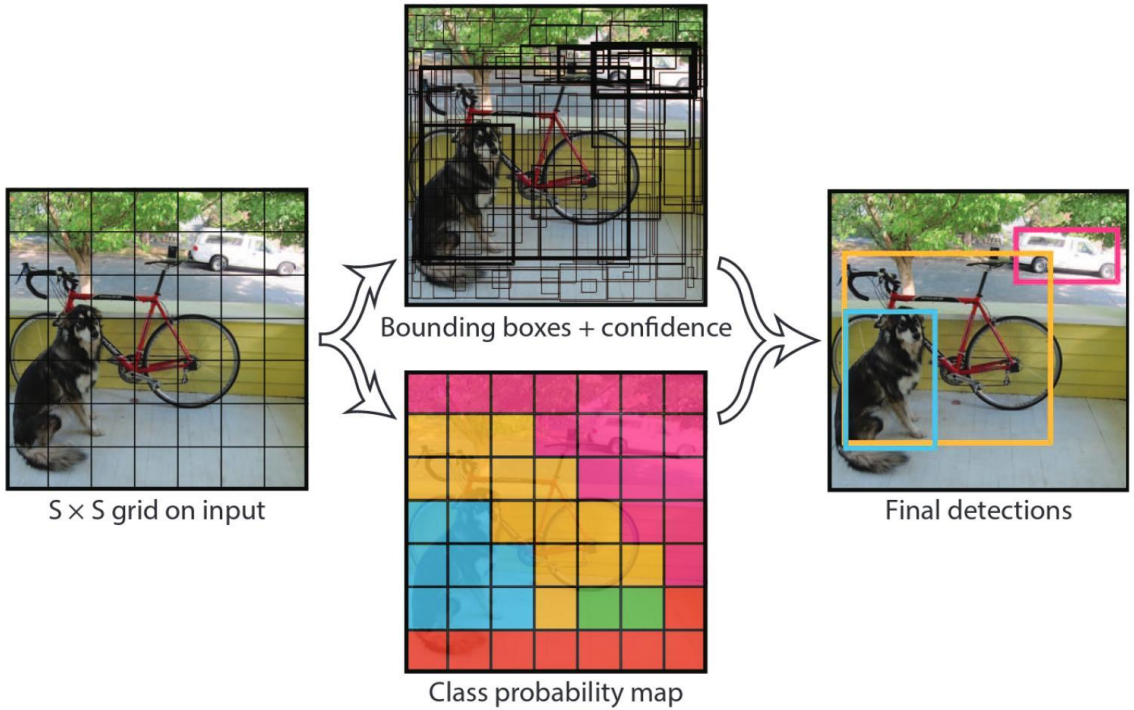

[1] Ren et al. Towards Real-Time Object Detection with Region Proposal Networks

# From Selective Search to Deep Learning: RPNs



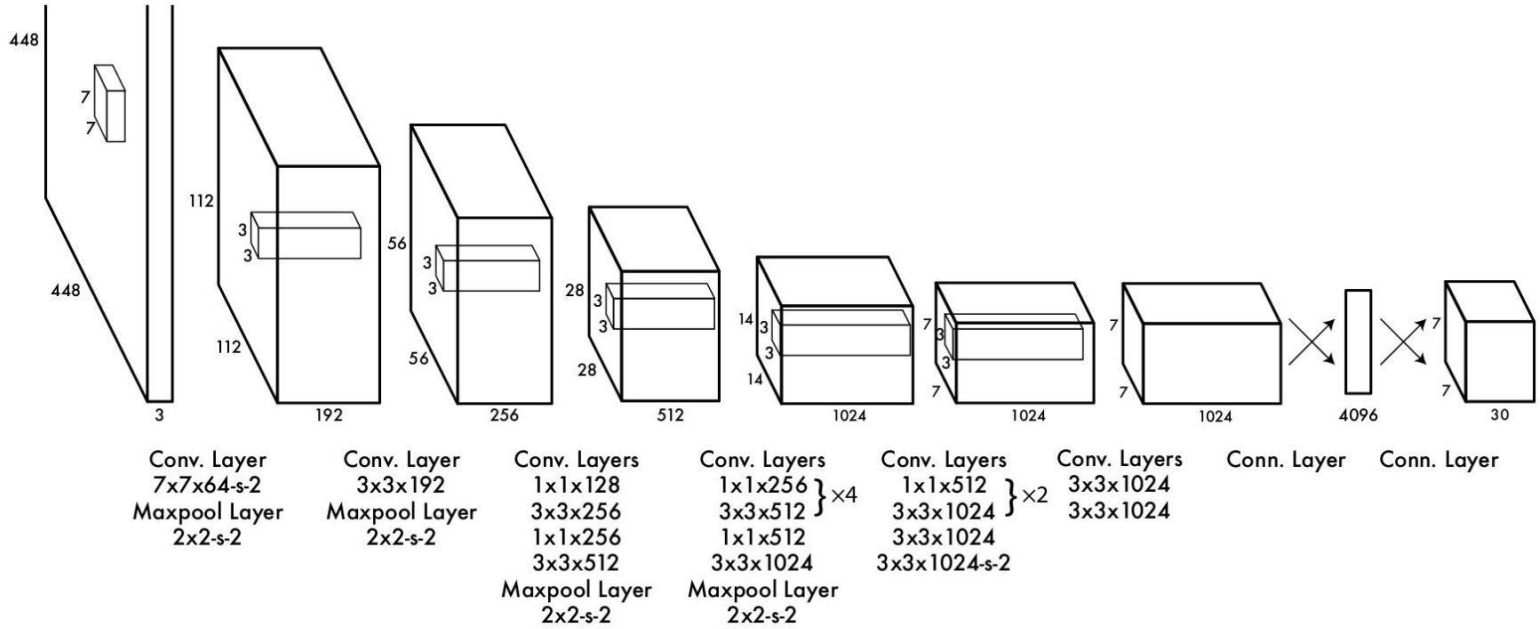
# Do we need object proposals?

- Single-shot detector YOLO [1]
- Split image into SxS grid
- Propose B boxes per grid cell
- Box := (x,y,w,h,**c**)
- C additional class probabilities per box
- S x S x (B * 5 + C) output predictions



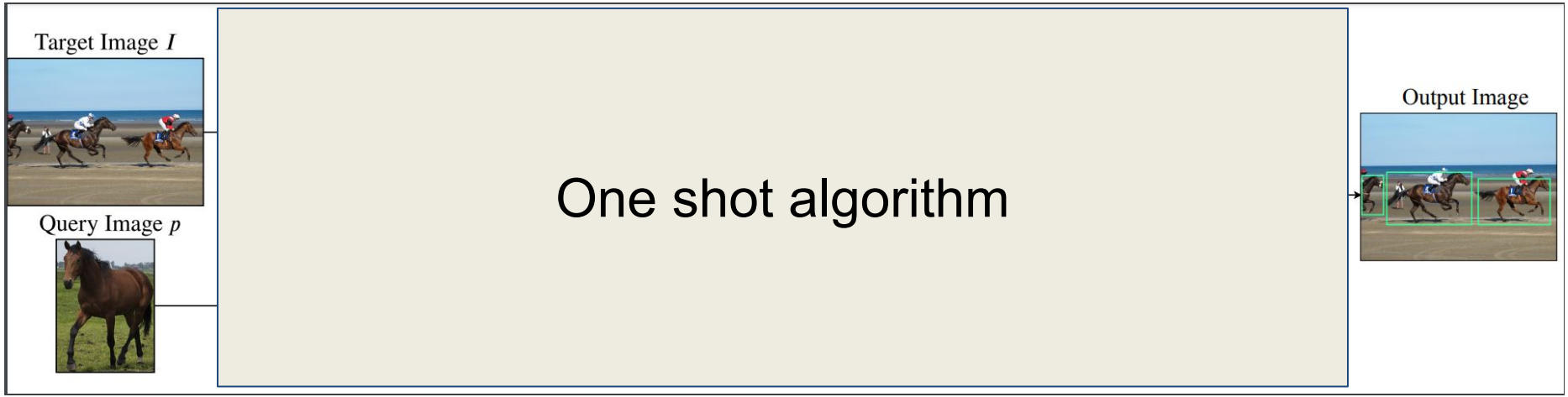

[1] Redmon et al., You Only Look Once: Unified, Real-Time Object Detection. CVPR 2016

# YOLO architecture

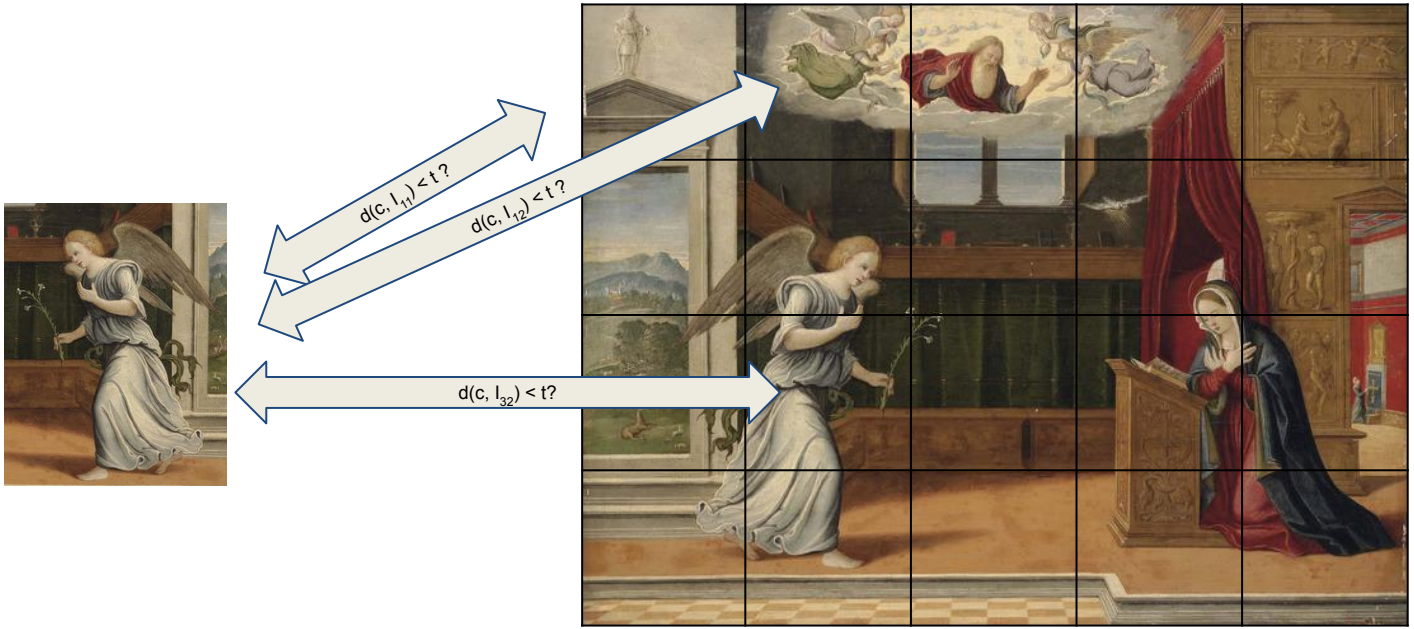



[1] Redmon et al., You Only Look Once: Unified, Real-Time Object Detection. CVPR 2016

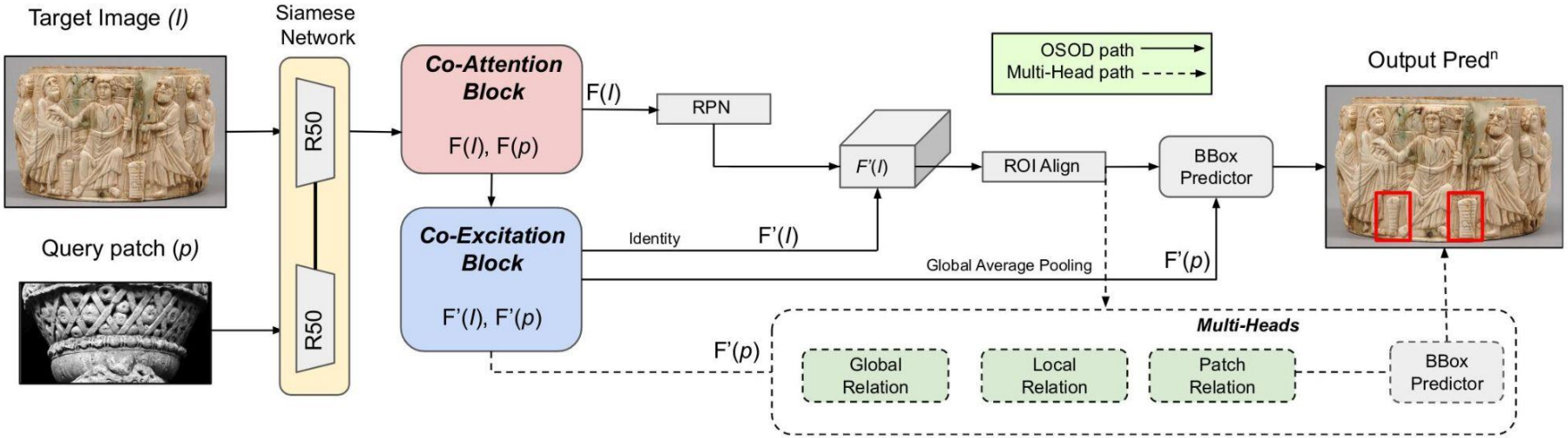
# One-Shot Learning for Humanities



# One-Shot Learning for Humanities



# One-Shot Learning for Humanities




[1] Madhu et al., One-Shot Object Detection in Heterogeneous Artwork Datasets, IPTA 2022

# Exercise (group)

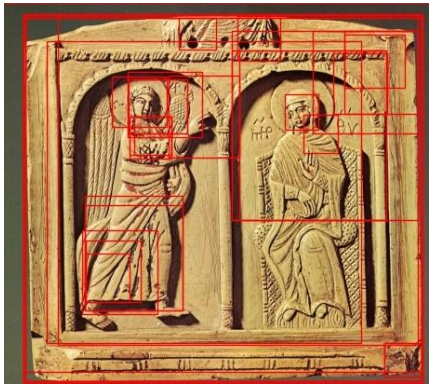
What you get?

- 9 images (3 from each of Art History, Class. Arch and Chris. Arch)

Your goal:

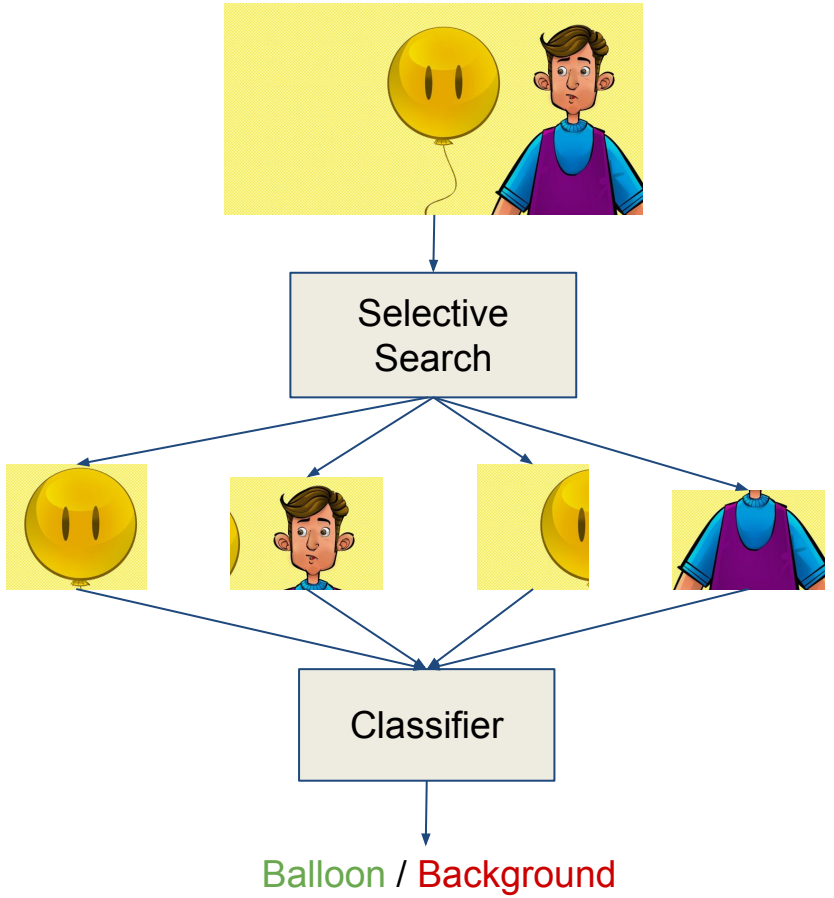
- Implement selective search
- Function stubs are given
- Submit generated region proposals
- Answer questions

Detailed description in exercise.pdf



# Exercise (individual)

- Implement a full detection pipeline using your selective search implementation
- Balloon Dataset (yay!)
- Crop candidate regions
- Generate global feature vector (per crop)
- Classify crop features
- Visualize classified boxes on the original canvas
- How to train the classifier?



# Further Reading

1. Felzenszwalb paper - http://cs.brown.edu/people/pfelzens/papers/seg-ijcv.pdf

2. Selective Search paper - http://huppelen.nl/publications/selectiveSearchDraft.pdf

3. Project on bag of words: https://cs.brown.edu/courses/csci1430/proj3/

# Felzenszwalb Details

# GRAPH - based segmentation

- **G** = (**V**, **E**) - undirected graph
  - $v_i \in \mathbf{V}$ - vertices (set of elements to be segmented, i.e. *pixels*)
  - $(v_i, v_j) \in \mathbf{E}$ - edges (pairs of neighboring vertices)
  - $w((v_i, v_j))$ - non negative dissimilarity
    - Based on intensity difference, color, motion etc.

- **S : Segmentation** (partition of V into components or region)
  - $\mathbf{C} \in \mathbf{S}$ - connected component in a graph **G'** = (**V**, **E'**), where $\mathbf{E'} \subseteq \mathbf{E}$

**Intuition:**

- Any segmentation induced by subset of its edges in E
- Edges between 2 vertices in same component/region → low weight

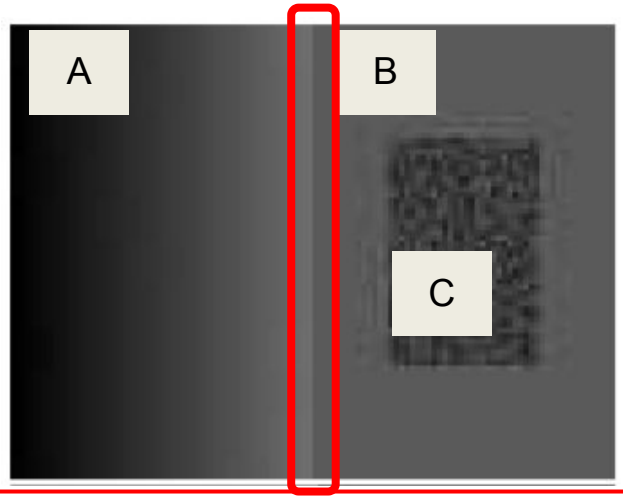
# Pairwise Region Comparison Predicate

- **D**: predicate to evaluate whether or not there is an evidence for a boundary between 2 regions

"Measuring dissimilarity *between elements along the boundary of 2 components* relative to a measure of the dissimilarity *among neighboring elements within each of 2 components*"

# Pairwise Region Comparison Predicate (PRCP)

- **D**: predicate to evaluate whether or not there is an evidence for a boundary between 2 regions




"Measuring dissimilarity *between elements along the boundary of 2 components* relative to a measure of the dissimilarity *among neighboring elements within each of 2 components*"

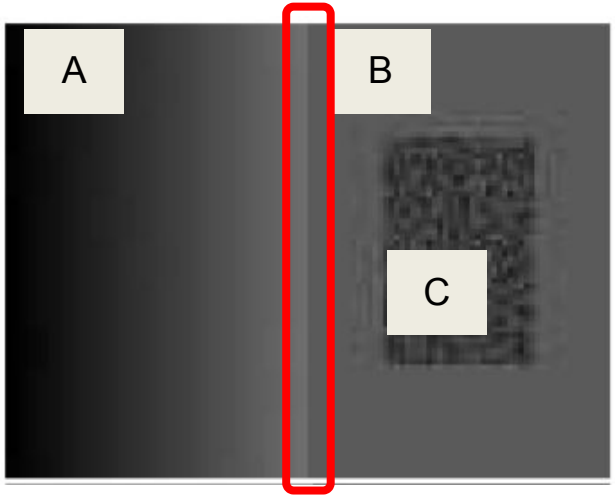
# Pairwise Region Comparison Predicate (PRCP)

- Internal difference of a component

$$Int(C) = \max_{e \in MST(C,E)} w(e)$$

- Difference between 2 components

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w((v_i, v_j))$$



# Pairwise Region Comparison Predicate (PRCP)

- PRCP

$$D(C_1, C_2) = \begin{cases} \text{true} & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ \text{false} & \text{otherwise} \end{cases}$$

where,

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1),\ Int(C_2) + \tau(C_2))$$

# Algorithm

Input: **G**(**V**, **E**) with *n* vertices and *m* edges
Output: **S** = ($C_1$, …, $C_r$) i.e. segmentation of **V**

1. Sort **E** into $\pi$ = ($o_1$, $o_2$, …, $o_m$) by non-decreasing edge weights

# Algorithm

Input: **G**(**V**, **E**) with *n* vertices and *m* edges
Output: **S** = ($C_1$, …, $C_r$) i.e. segmentation of **V**

1. Sort **E** into $\pi$ = ($o_1$, $o_2$, …, $o_m$) by non-decreasing edge weights
2. Start with seg. $\mathbf{S_0}$ where $v_i$ is in its own component

# Algorithm

Input: **G**(**V**, **E**) with *n* vertices and *m* edges
Output: **S** = ($C_1$, …, $C_r$) i.e. segmentation of **V**

1. Sort **E** into $\boldsymbol{\pi}$ = ($o_1$, $o_2$, …, $o_m$) by non-decreasing edge weights
2. Start with seg. $\mathbf{S_0}$ where $v_i$ is in its own component
3. Repeat Step 4 for q = 1, 2, …, m
4. Construct $\mathbf{S_q}$ from $\mathbf{S_{q-1}}$ as follows:
   a. $v_i$ and $v_j$ → connected by qth edge → $o_q$ = ($v_i$, $v_j$)

# Algorithm

Input: **G**(**V**, **E**) with *n* vertices and *m* edges
Output: **S** = ($C_1$, …, $C_r$) i.e. segmentation of **V**

1. Sort **E** into $\pi$ = ($o_1$, $o_2$, …, $o_m$) by non-decreasing edge weights
2. Start with seg. $\mathbf{S_0}$ where $v_i$ is in its own component
3. Repeat Step 4 for q = 1, 2, …, m
4. Construct $\mathbf{S_q}$ from $\mathbf{S_{q-1}}$ as follows:
   a. $v_i$ and $v_j$ → connected by qth edge → $o_q$ = ($v_i$, $v_j$)
   b. If $v_i$ and $v_j$ are in disjoint components of $\mathbf{S_{q-1}}$ and w($o_q$) is small compared to internal diff. Of both these components then merge these two components, else do nothing

# Algorithm

Input: **G**(**V**, **E**) with *n* vertices and *m* edges
Output: **S** = ($C_1$, …, $C_r$) i.e. segmentation of **V**

1. Sort **E** into $\pi$ = ($o_1$, $o_2$, …, $o_m$) by non-decreasing edge weights
2. Start with seg. $\mathbf{S_0}$ where $v_i$ is in its own component
3. Repeat Step 4 for q = 1, 2, …, m
4. Construct $\mathbf{S_q}$ from $\mathbf{S_{q-1}}$ as follows:
   a. $v_i$ and $v_j$ → connected by qth edge → $o_q$ = ($v_i$, $v_j$)
   b. If $v_i$ and $v_j$ are in disjoint components of $\mathbf{S_{q-1}}$ and w($o_q$) is small compared to internal diff. Of both these components then merge these two components, else do nothing
5. Return **S** = $\mathbf{S_m}$