Introduction Segmentation

Computer Vision

Outline

- **Introduction**
    - Couse structure
    - Computer vision applications
    - Stereovision
    - Object tracking
- **Segmentation**
    - Connected regions
    - Merging and dividing regions
    - Segmentation methods

Introduction

Syllabus

- **Teachers:**
    - Ph.D, Associate Professor Sergei Shavetov, s.shavetov@itmo.ru
    - Ph.D, Associate Professor Andrei Zhdanov, adzhdanov@itmo.ru
    - (CS) Assistant Professor, Aleksandr Belykh, vvbespalov@itmo.ru
    - (AT) Ph.D, Assistant Professor, Oleg Evstafev, oaevstafev@itmo.ru
    - (AT) Ph.D, Assistant Professor, Vladimir Bespalov, vvbespalov@itmo.ru
- **Course structure:** 16 lectures * 45 minutes each
- **Practical assignments:** 4
- **Intermediate tests:** 2
- **Final test**

Detailed Course Structure

- Introduction
- Image Segmentation
- Hough Transformation
- Image Features
- Object Description
- Feature Descriptors
- Introduction to Machine Learning
- Classification
- Images Categorization
- Object Detection
- Face Detection
- Image Search
- Neural Networks
- Deep Learning
- Convolutional Neural Networks

Practical Assignments Topics

- Image Segmentation
- Hough Transformation
- Feature Detectors
- Face Detection. Viola-Jones Approach

Files Exchange

- **DingTalk Group / Files:**
    - Lecture presentations
    - Practical assignment guidelines
    - Practical assignment templates

Course Assessments

- **4 practical assignments:** 11 points each (total 44 points max)
    - **practical task:** 5 points
    - **test:** 6 points
- **After lecture tests:** 2 points each (total 16 points max)
- **Intermediate test:** 10 points each (total 20 points max)
- **Final test:** 20 points

- **Course assessment:**
    - 60 points and more – pass,
    - 59 points and less – fail.

Course Deadlines

- Practical assignments are performed in groups of 1-3 students
- Practical assignment consists of two parts:
    - Practical task (5 points)
    - Test on the learned material (6 points)
- To get the maximum score, the group should finish all parts of the practical assignment, submit test answers, and show the practical task results to the teacher during class time
- Some tasks may have optional parts which will give the group 1 extra point

Reasons for Points Decreasing

- **Not attending the class** (all tasks should be finished in class, not attending the class results in 0 points for the in-class activity)
- **Not finishing a task** (will decrease score based on the complexity of the not finished task)
- **Not answering the questions about the finished task** (will decrease score based on answers quality, in the worst case you may be suspected in copying your classmates work)
- **Copying your classmates' works** (in this case you can't get more than 3 points for the practical assignment)

Practical Assignments Frameworks

- **MATLAB including:**
  - Digital Image Processing Toolbox,
  - Computer Vision Toolbox.
- **Python developer package with:**
  - Jupiter,
  - NumPy,
  - OpenCV.
- **Microsoft Visual Studio including:**
  - Desktop Development with C++,
  - OpenCV.

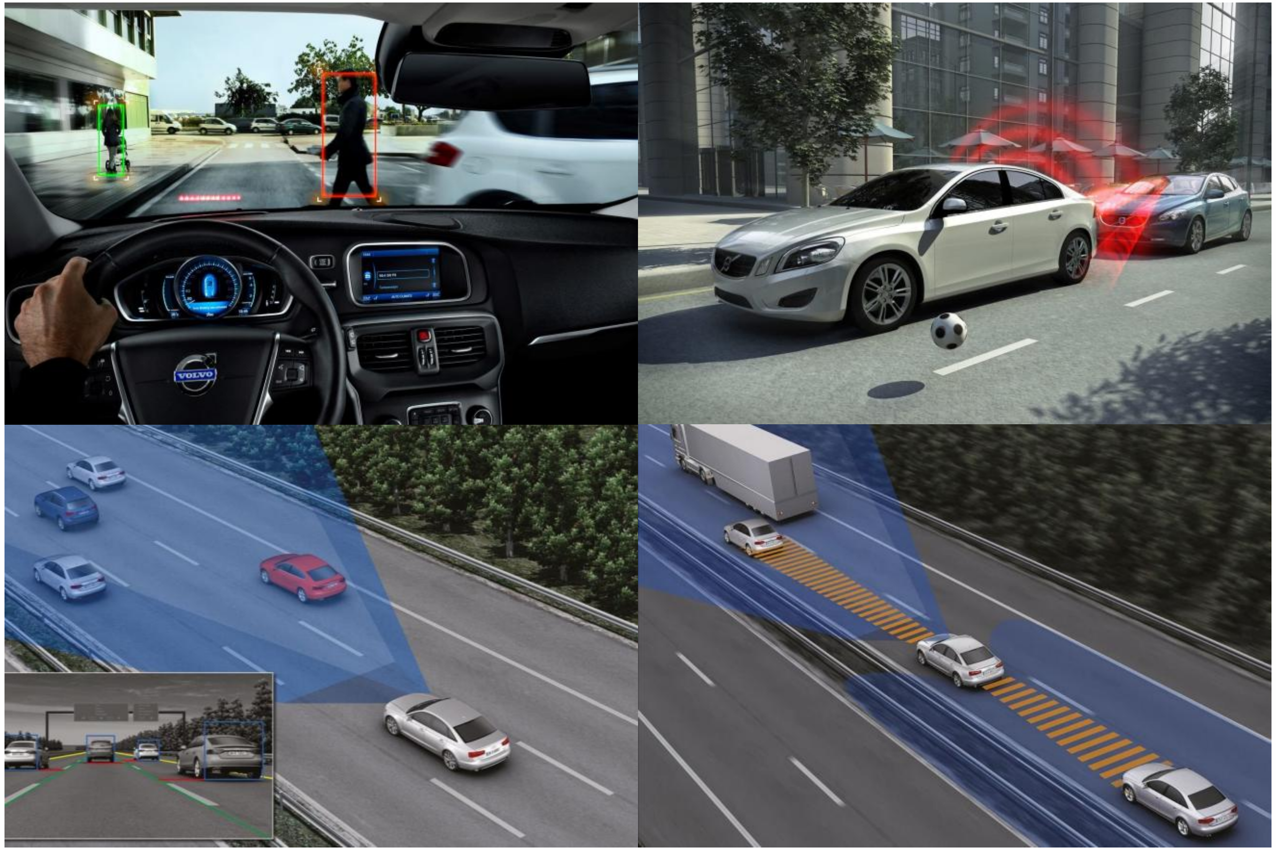
Literature

- David A. Forsyth, Jean Ponce. Computer Vision: A Modern Approach (2nd Edition) // Pearson, 2011.
- Linda G. Shapiro, George C. Stockman. Computer Vision // Pearson, 2001.
- https://www.mathworks.com/help/vision/index.html
- https://docs.opencv.org/4.x/d1/dfb/intro.html

Computer Vision Applications

Manipulator on a Conveyor
(ABB FlexPicker)

Obstacles Detection



Normal Error vs Abnormal Error

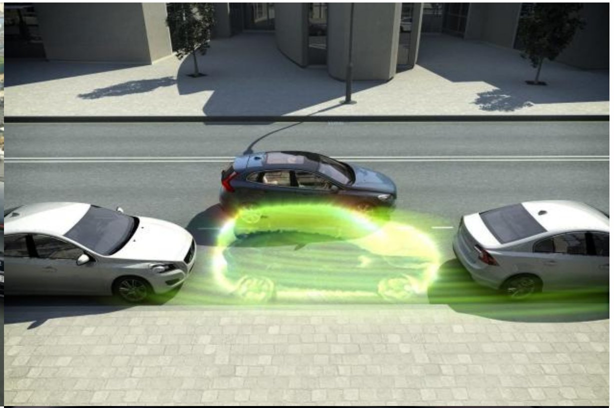
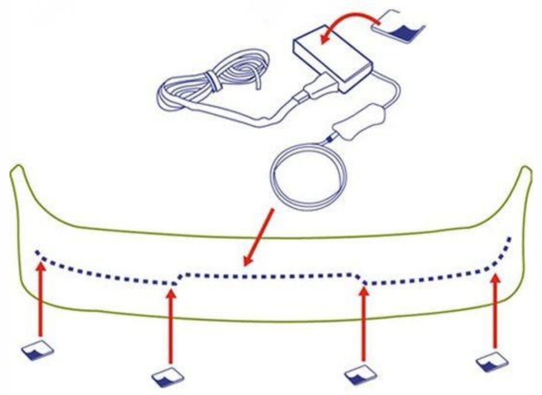
Pedestrians Detection

Pedestrians Detection

ITMO

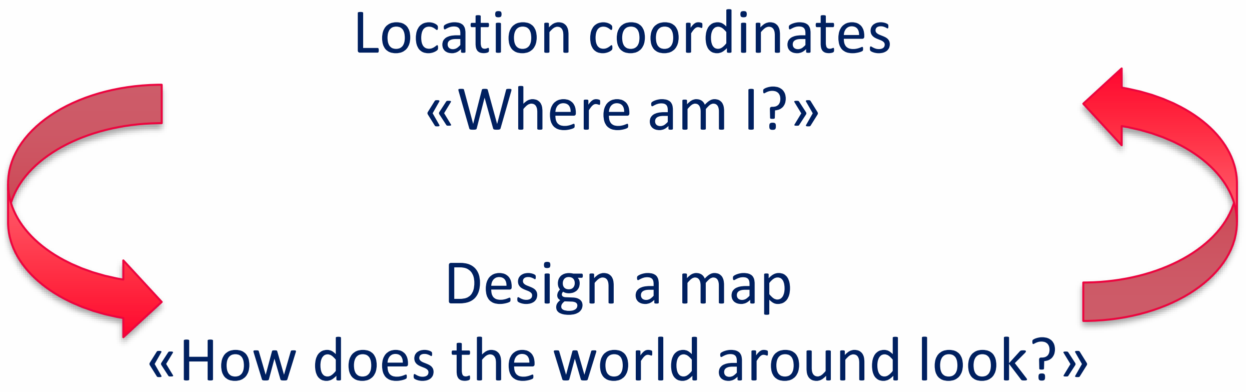
Objects Detection

Parking System

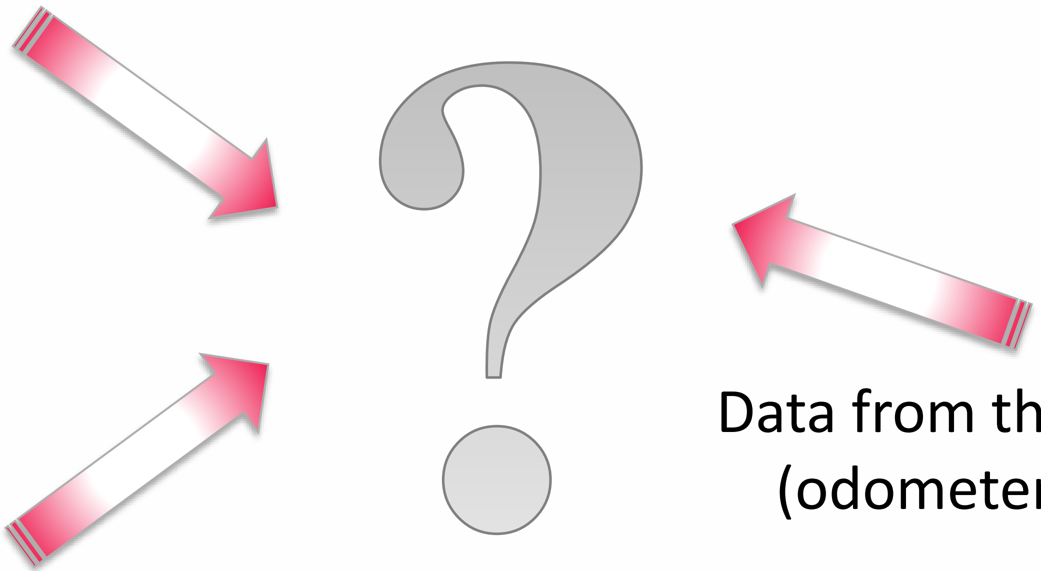
CV for Mobile Robots – vSLAM

- Visual Simultaneous Localization and Mapping (vSLAM)
- It is used:
    - to design a map in an unknown environment or
    - to update a map in a known environment
- with simultaneous keeping track of the robot location.



vSLAM

Data about the environment



Data from the robot sensors (odometers, lidars, etc.)

Map data

vSLAM in ROS (Robot Operating System)

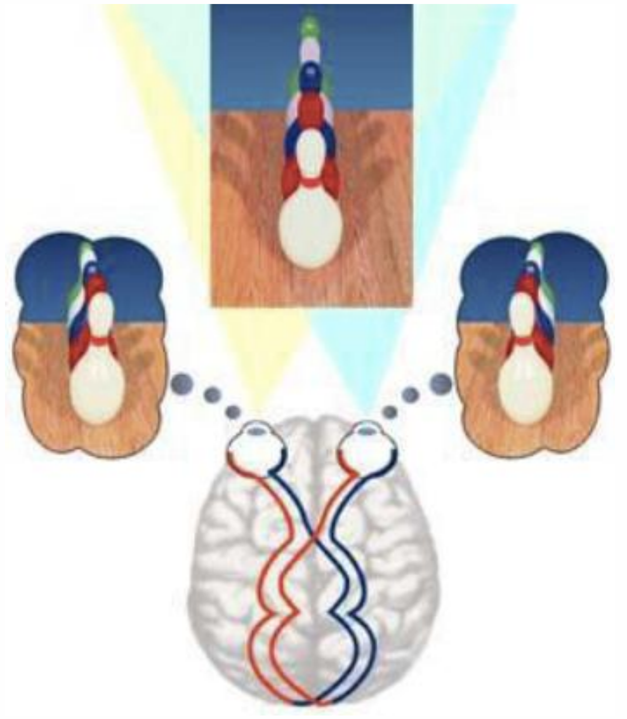
rgbdslam

gmapping

Stereovision

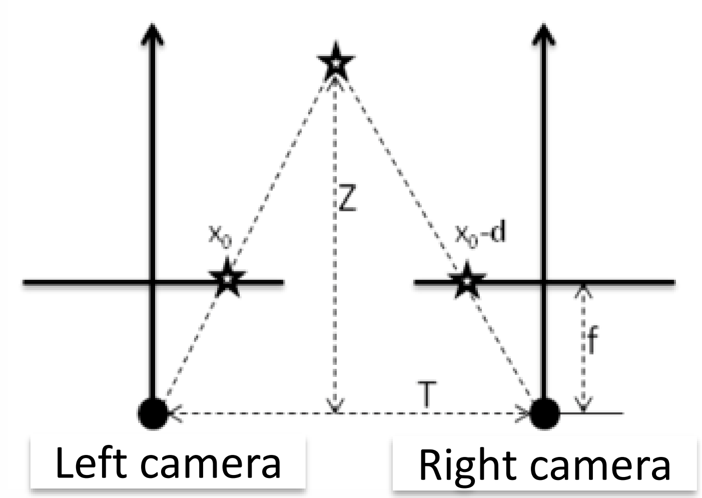
(distance measurement)

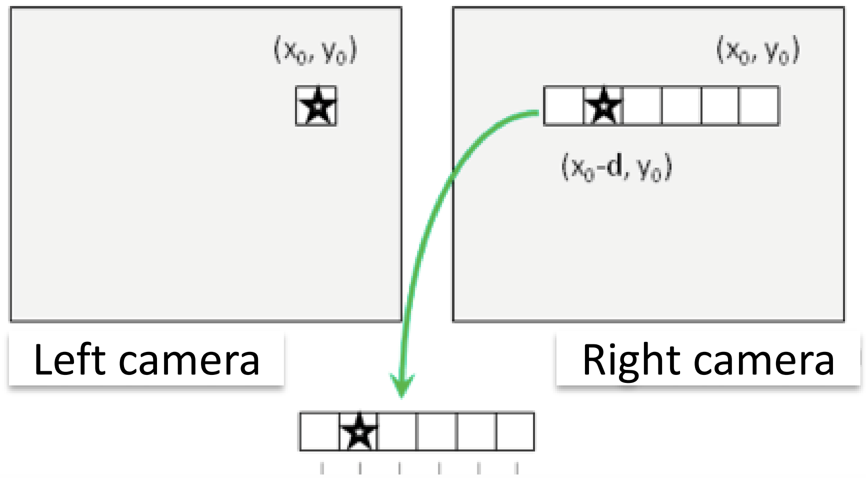
- **Binocular disparity** is the difference in the position of objects in terms of each eye.
- **Stereoscopy** is a sense of depth derived from binocular disparity.



Distance Measurement

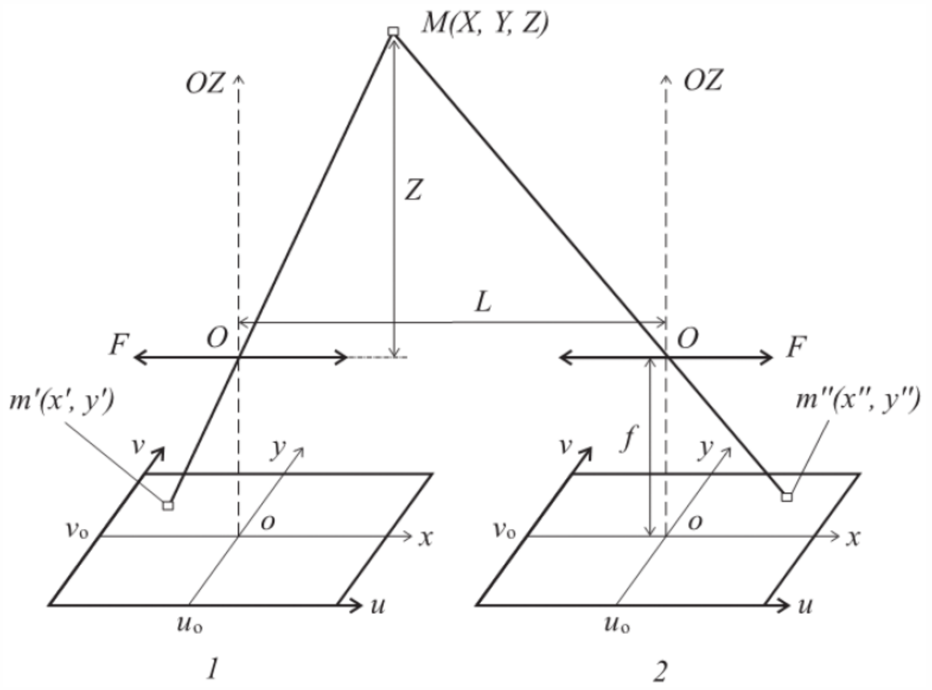
- **Disparity map:**
  - for each pixel in the *left picture* with coordinates (x_0, y_0),
  - a pixel is searched for in the right picture with coordinates $(x_0 - d, y_0)$.





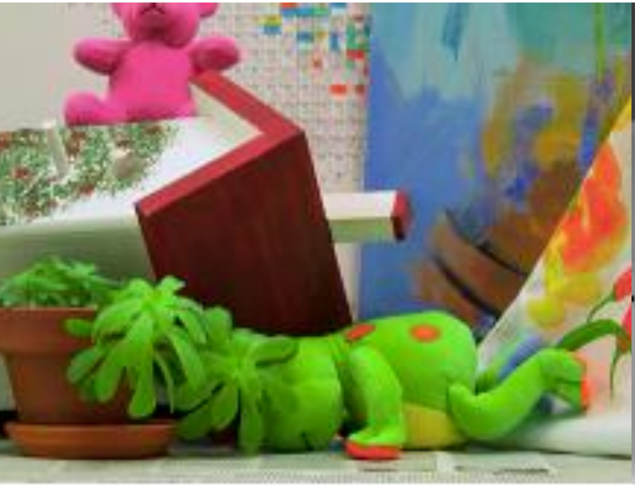
Distance Measurement

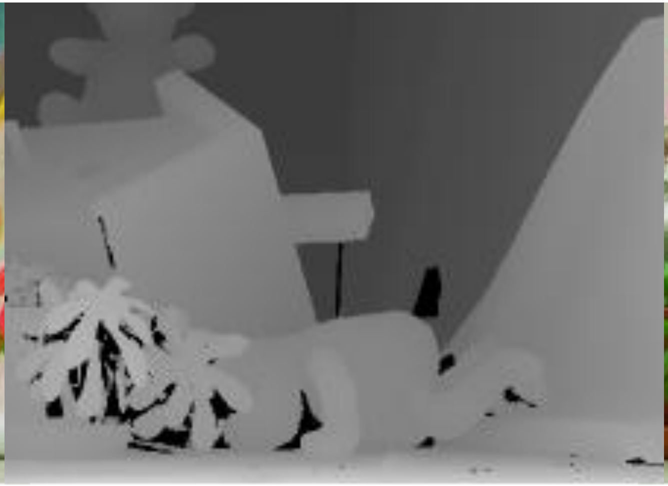
- Z – distance,
- f – focal length,
- L – base length,
- D – disparity,
- x' and x'' – coordinates of the object in the image plane in the right and left pictures of the stereo pair, respectively.

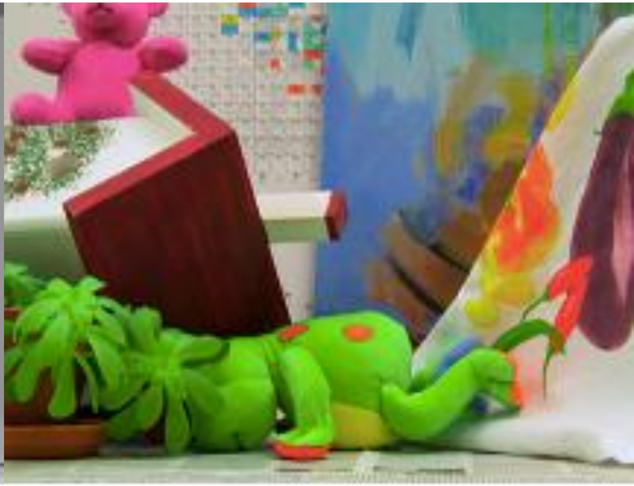


$$\boxed{Z = \frac{f \cdot L}{x' - x''} = \frac{f \cdot L}{d}}$$

Distance Measurement

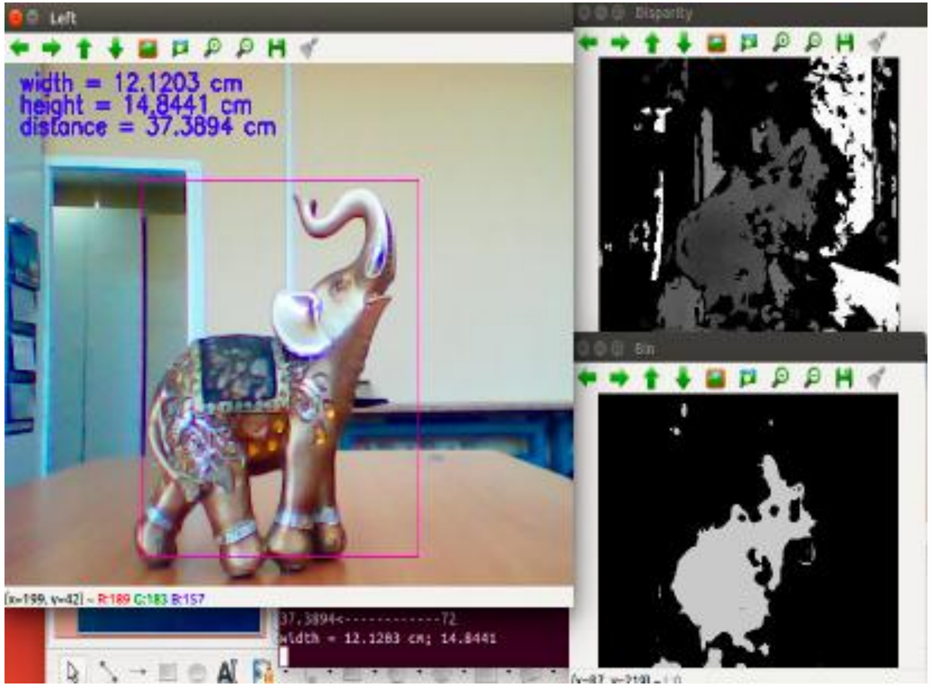

Left camera


Disparity map


Right camera

Distance Measurement

- StereoSGBM – OpenCV class for calculation stereo matching.



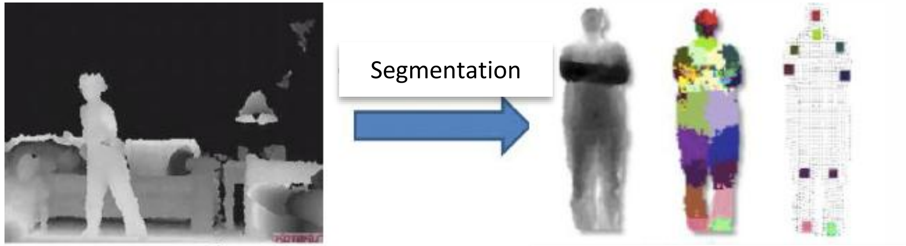
```
width = 12.1203 cm
height = 14.8441 cm
distance = 37.3894 cm
```

```
(x=199, y=42) ~ R:189 G:183 B:157
```

```
37.3894<------------72
width = 12.1283 cm; 14.8441
```

Skeleton Tracking

- ROS openni_tracker



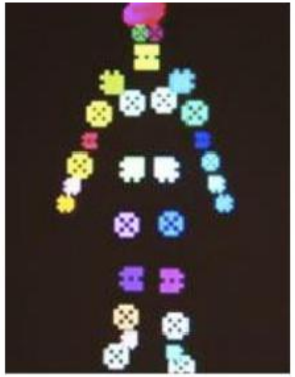
Depth map

Torso position

RGB

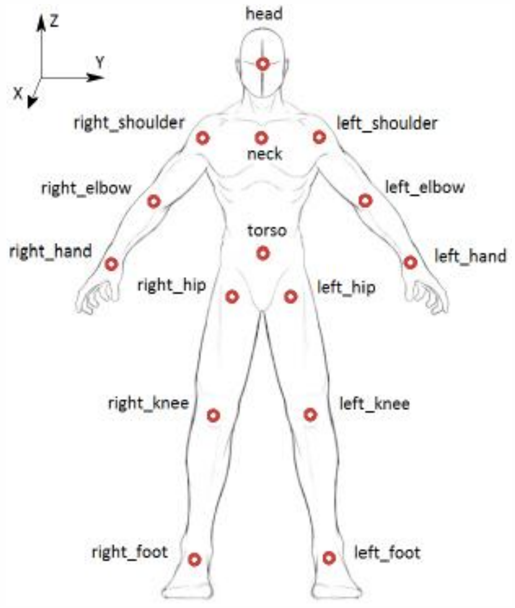
Depth map

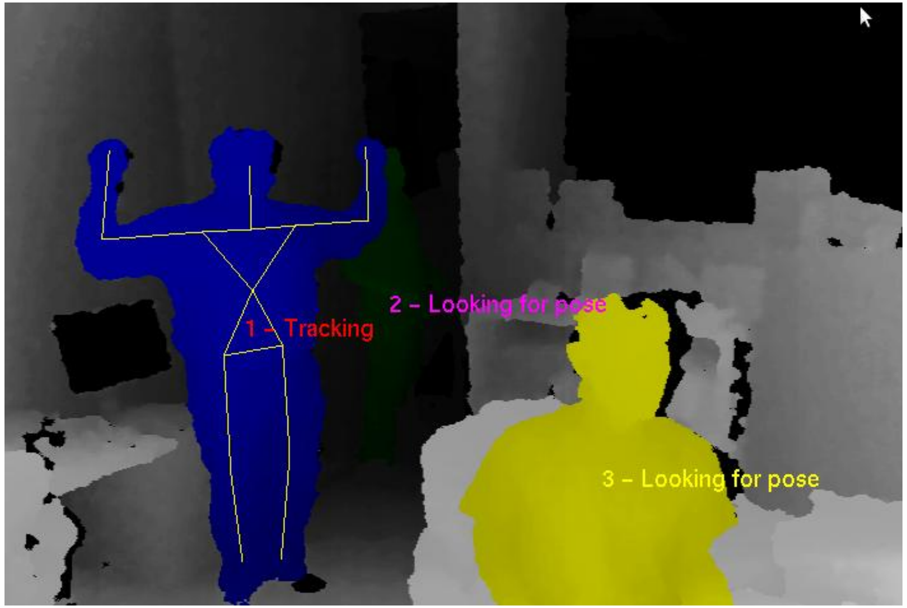
Body parts


Joints positions

Skeleton Tracking

- ROS openni_tracker



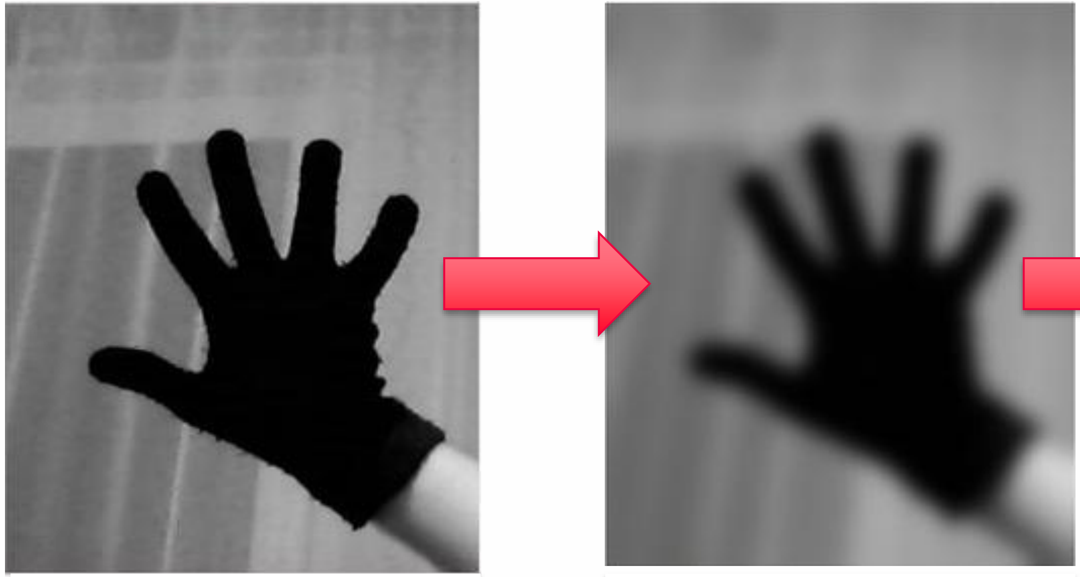


Fingers Counting

$Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$

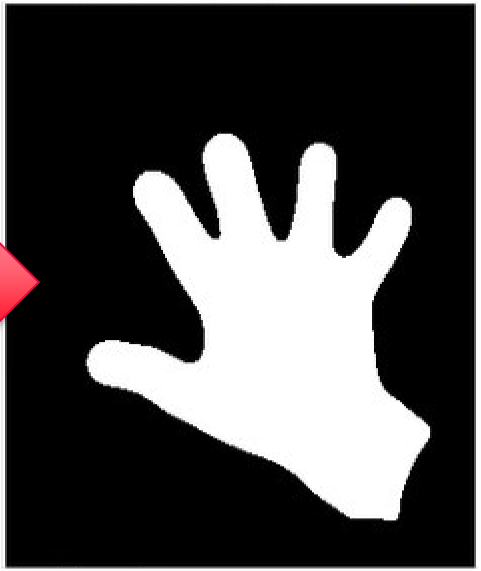
$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$

$\sigma_w^2 = w_1\sigma_1^2 + w_2\sigma_2^2$



RGB to Grayscale

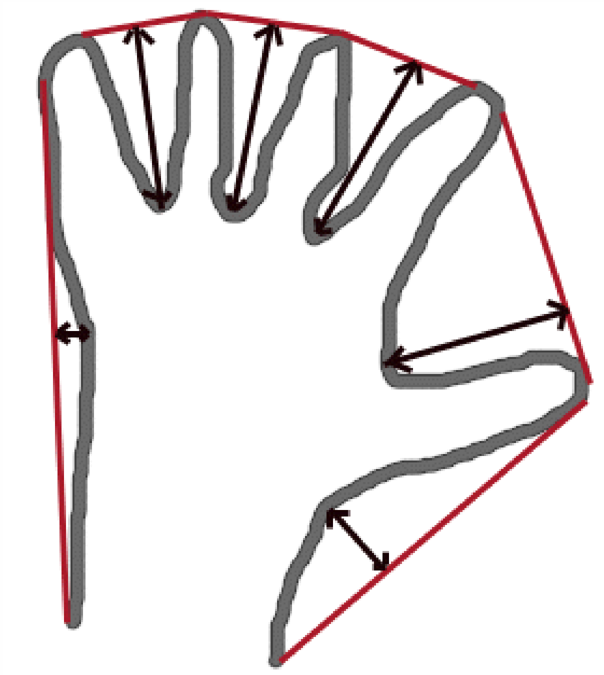
Gauss blurring



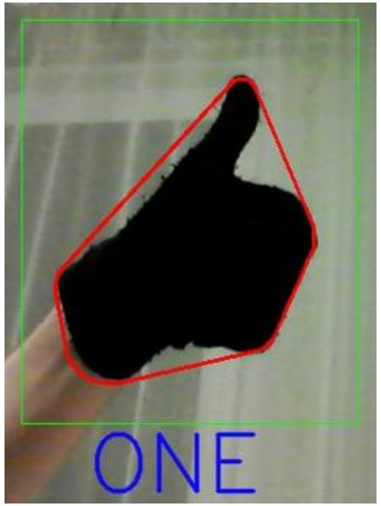
Binarization

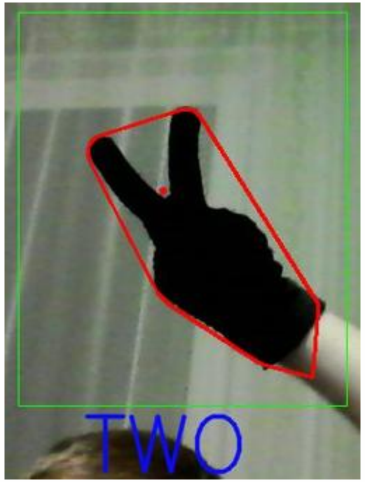
Fingers Counting

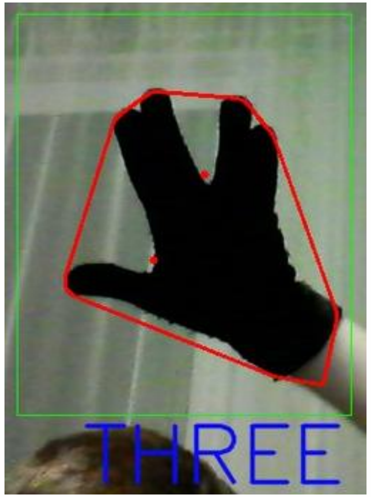
- Computation of convex hull defects.

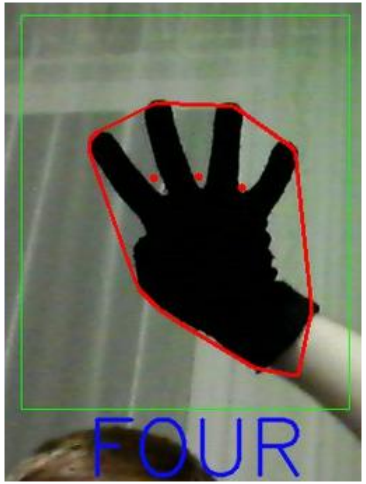


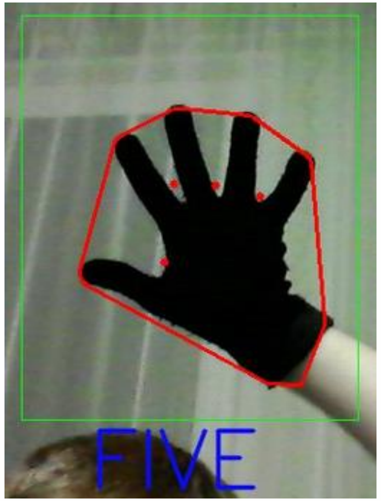
Fingers Counting



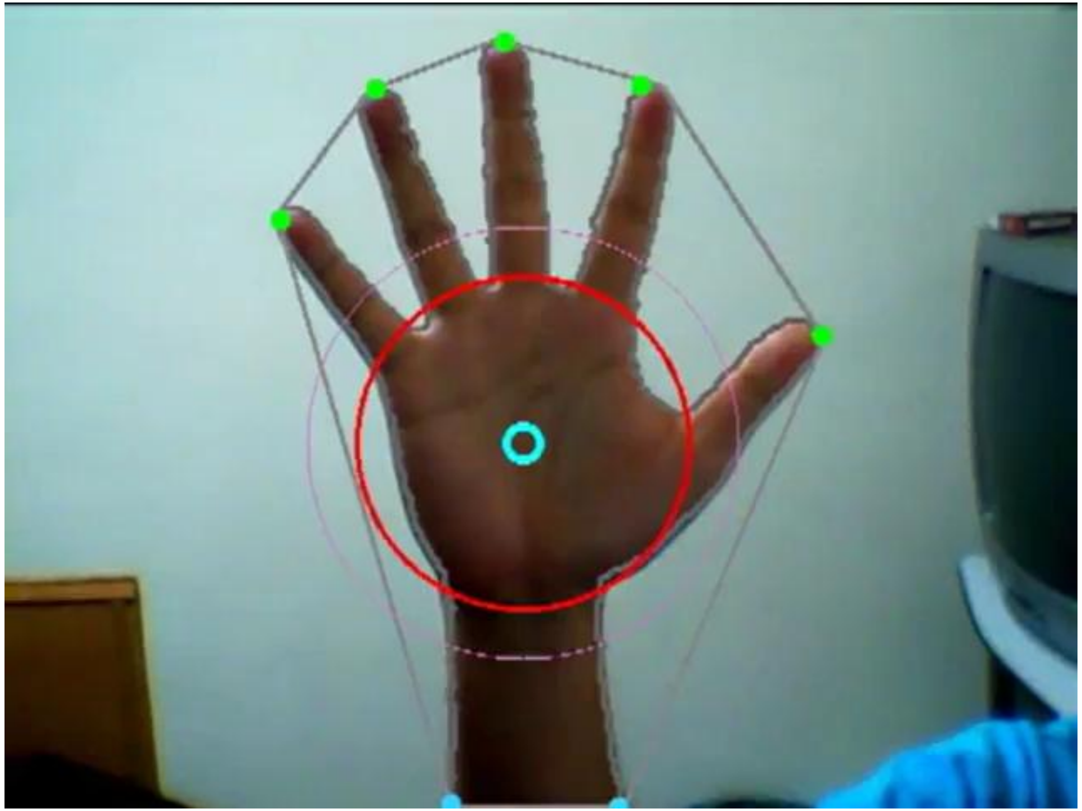






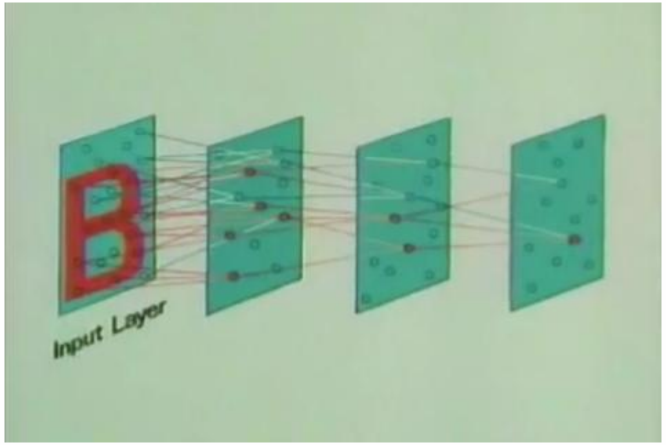


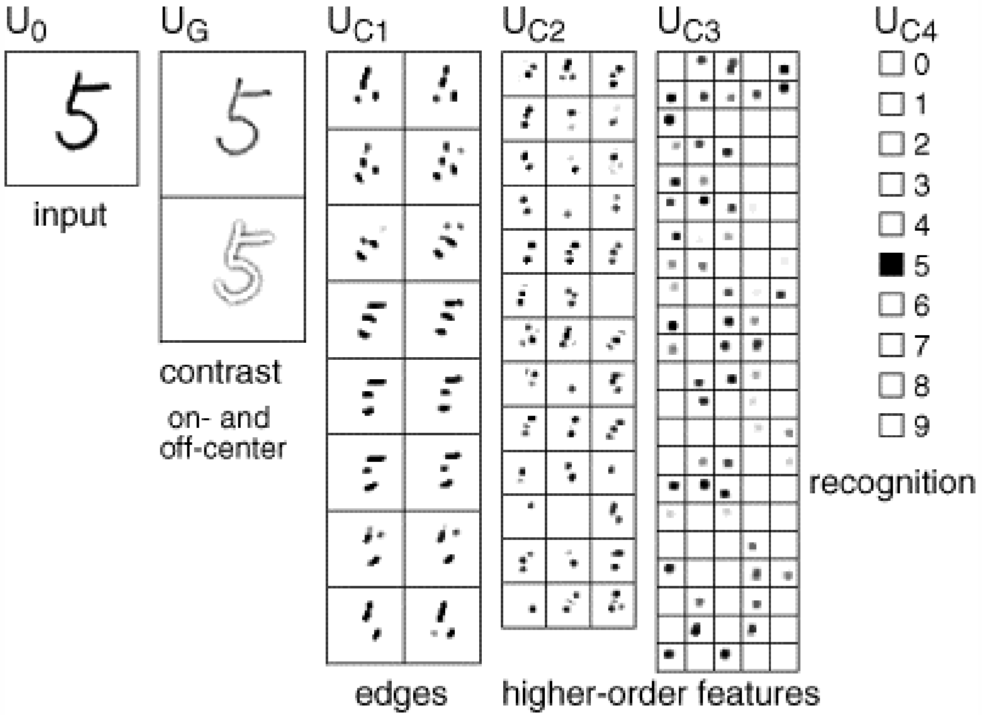
Fingers Counting



Object Detection

(using neural networks)





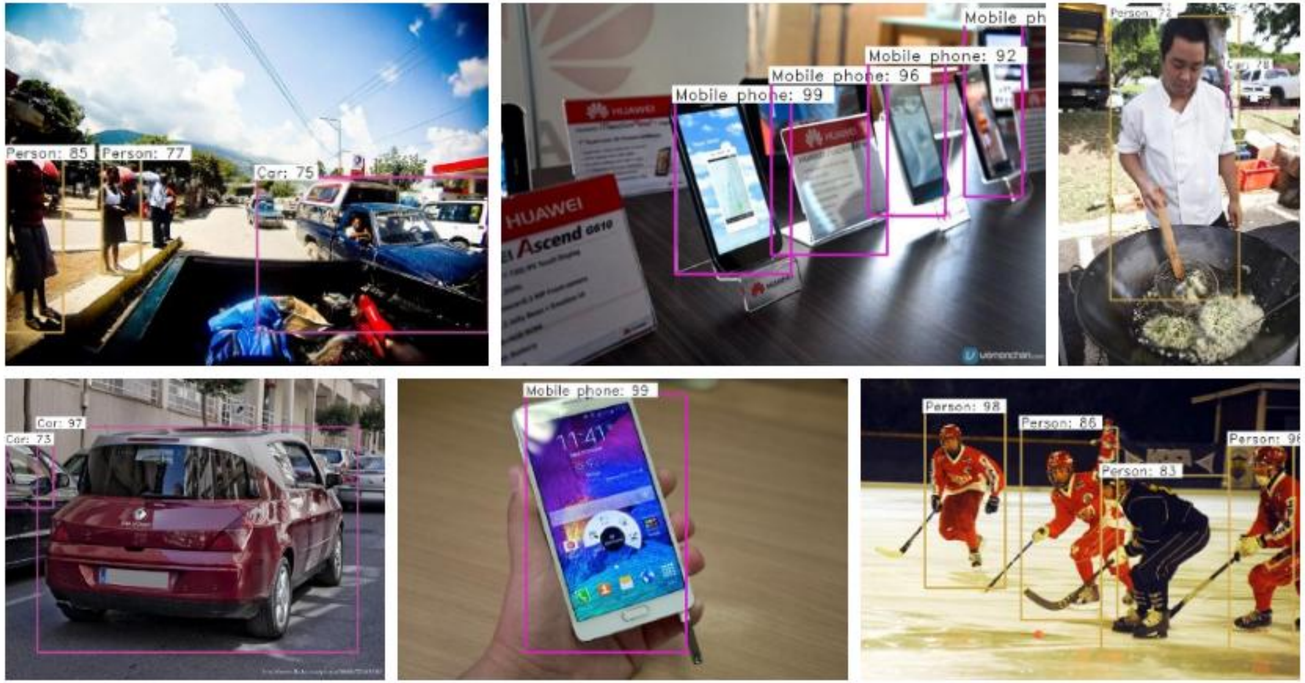
Object Detection
(using neural networks)

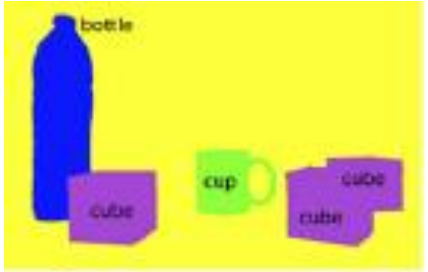
Similar images

Caffe – C++ Framework for implementing deep learning algorithms
http://caffe.berkeleyvision.org/

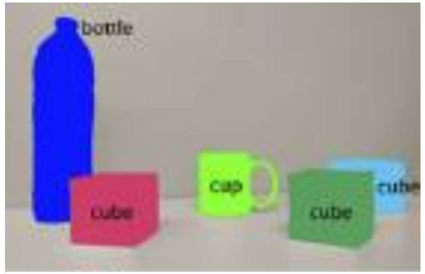
ImageNet – labeled dataset
http://www.image-net.org/

Segmentation





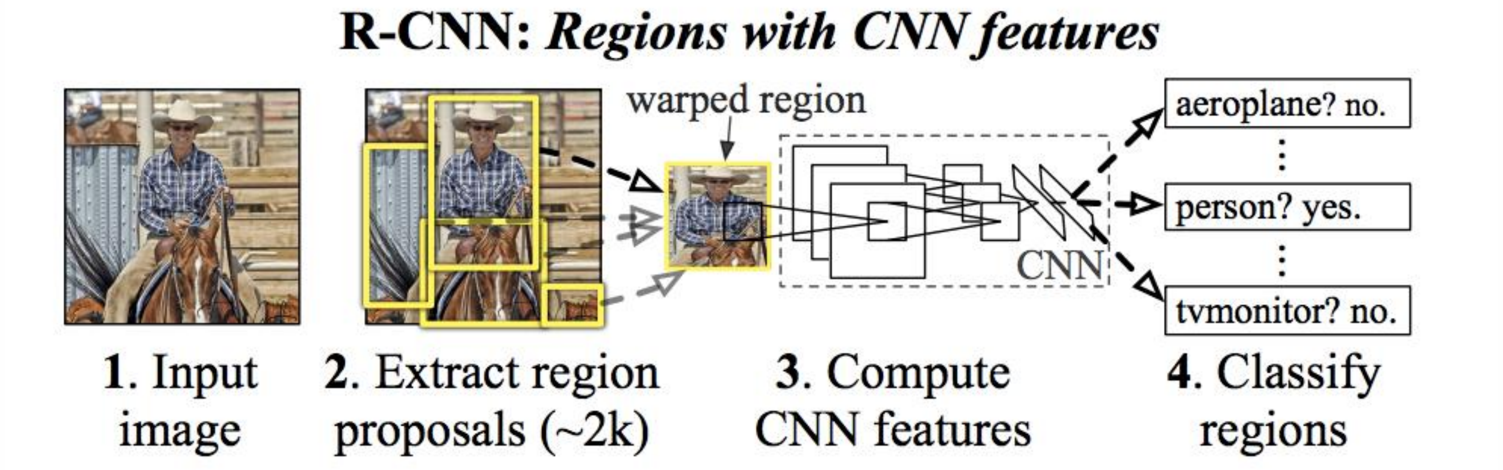
Semantic segmentation



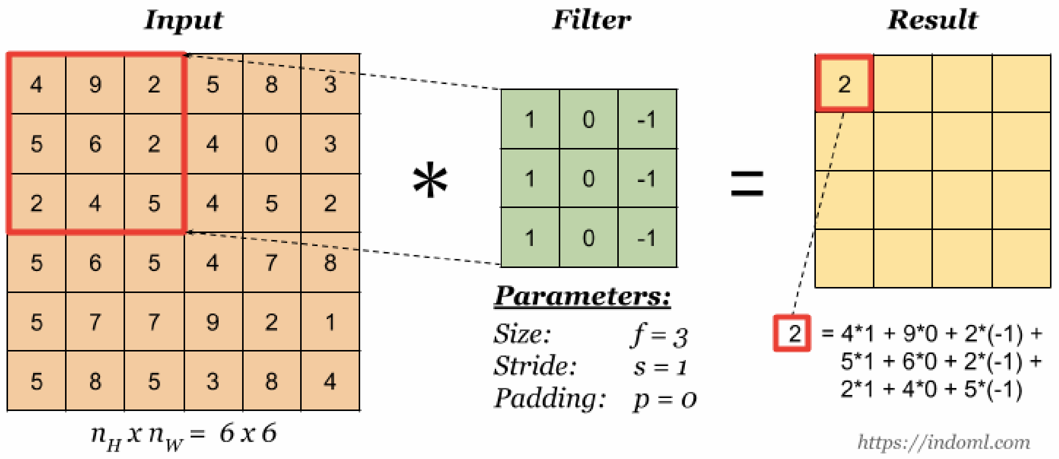
Object segmentation

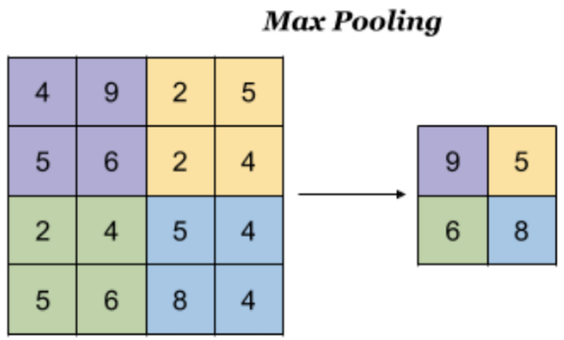
Regions with Convolutional Neural Networks

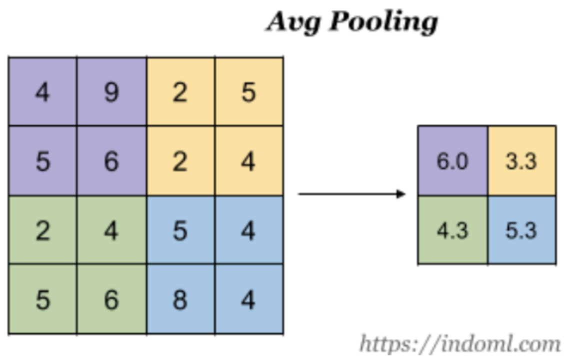
- The R-CNN architecture identifies 2,000 candidate regions in the image and works with them:



Convolution







Mask R-CNN

R-CNN performance

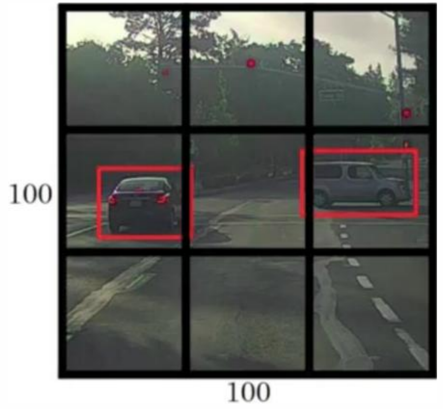
- Performance with the same number of classes and equal compute resources:
  - R-CNN: 40-50 s;
  - Fast R-CNN: 2 s;
  - Faster R-CNN: 0.2 s;
  - Mask R-CNN: 0.2 s.
- **The R-CNN, Fast R-CNN**, and **Faster R-CNN** are fit for object detection and *semantic segmentation*;
- **Mask R-CNN** is fit for object detection and *object segmentation*.

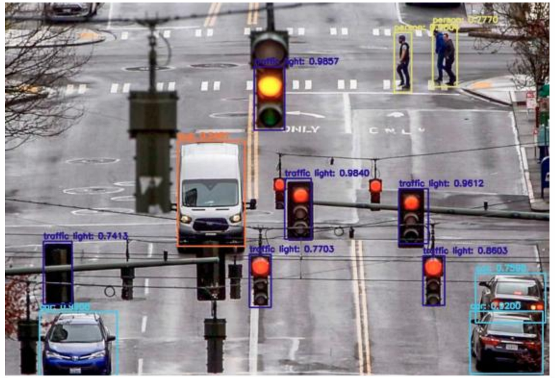
YOLO – You Only Look Once

- The image is divided by a grid to cells.
- Classification and localization algorithms are applied to each cell.
- In each cell the location of the bounding rectangles and the corresponding probabilities are estimated.

YOLO – You Only Look Once

- Modifications:
  - TinyYOLO;
  - YOLOv2;
  - YOLOv3…
  - 4, 5, 6, 7, 8, 9, …
- Performance: ~45 FPS



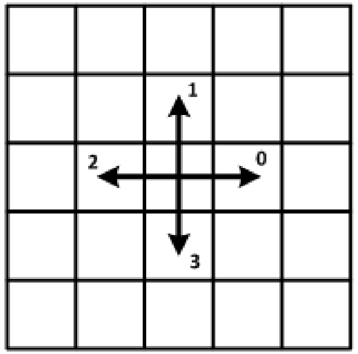


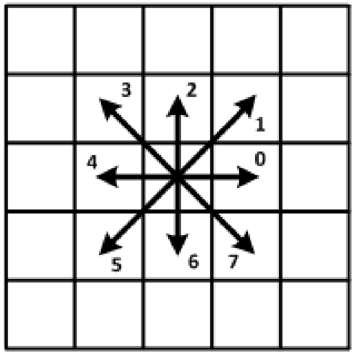
YOLOv3 & dataset COCO

Segmentation

Connected Regions in Binary Images

- **Neighborhood**
    - "cross" neighborhood and 4-connectivity
    - "square" neighborhood and 8-connectivity
- H**orizontal and vertical** *neighbors* are at a distance of 1 pixel from the central pixel of the neighborhood
- **Diagonal** *neighbors* are at a distance of $\sqrt{2}$ pixels from the central pixel of the neighborhood





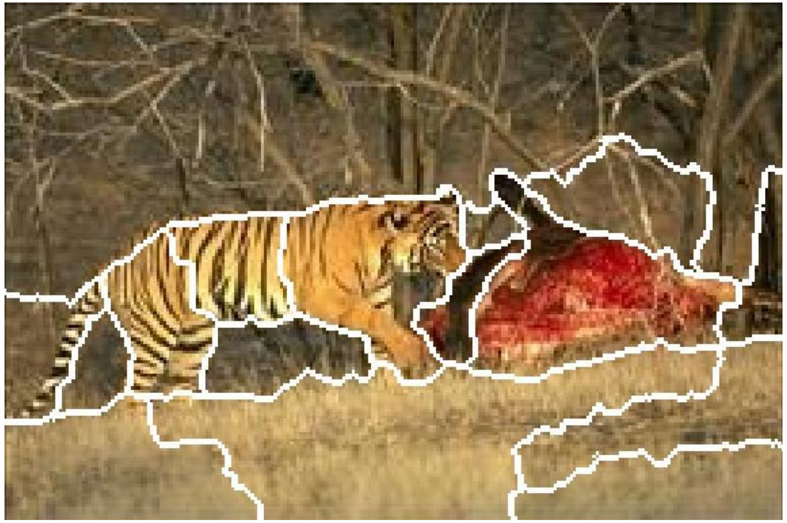
Connected Regions in Binary Images

- The connected region of the image is the region (a set of points) if
  - all points have the same value
  - there is a continuous path between any two points from the given region consisting of points that also belong to the given region and are *neighbors* at the same time

- **Algorithms for selecting connected regions**
  1. “Forest fire” method (Y and U collision forms possible)
  2. Two pass algorithm (algorithm avoids collision)

Segmentation

- Splitting an image into non-overlapping regions, each of which is represented by a color or texture of the same type.
- The purpose of segmentation in the "*broad sense*": split an image into semantic regions that have a strong correlation with objects or regions of the observed three-dimensional scene.



Segmentation

- R – is the entire region of the image.
- Segmentation is the process of splitting R into a such set of connected regions $\{R_i\}, i = 1, \dots, n,$ that the following basic conditions are met for them:
  - $R = \cup R_{i=1,\dots,n}$– regions completely cover the image;
  - $R_i \cap R_j = \emptyset, \forall i \neq j$ –regions do not intersect with each other;
  - $\mathrm{Pred}(R_i) = TRUE, i = 1, \dots, n,$ where $\mathrm{Pred}(R)$ is the Boolean homogeneity predicate of the region;
  - $Pred\left(R_i \cup R_j\right) = FALSE, \forall\, i \neq j$ – the pairwise union of any two regions does not satisfy the same homogeneity condition.

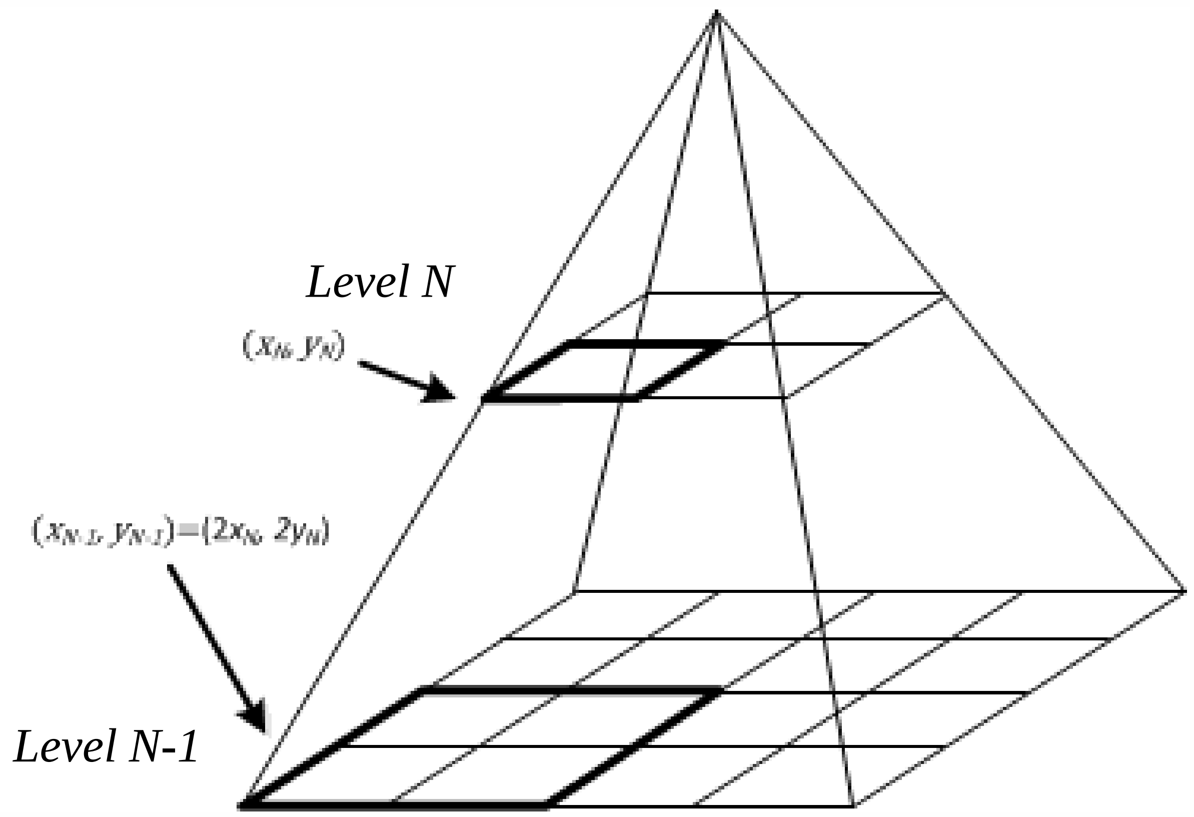
Merging Regions

- Perform presegmentation of the image into "starting" regions using a non-iterative (single) method.
- Determine the criterion for the merging of two neighboring regions.
- Iteratively find and merge all pairs of neighboring regions that satisfy the merge criterion.
- If no pair of candidates for merging is found, stop and exit the algorithm.

Splitting Regions

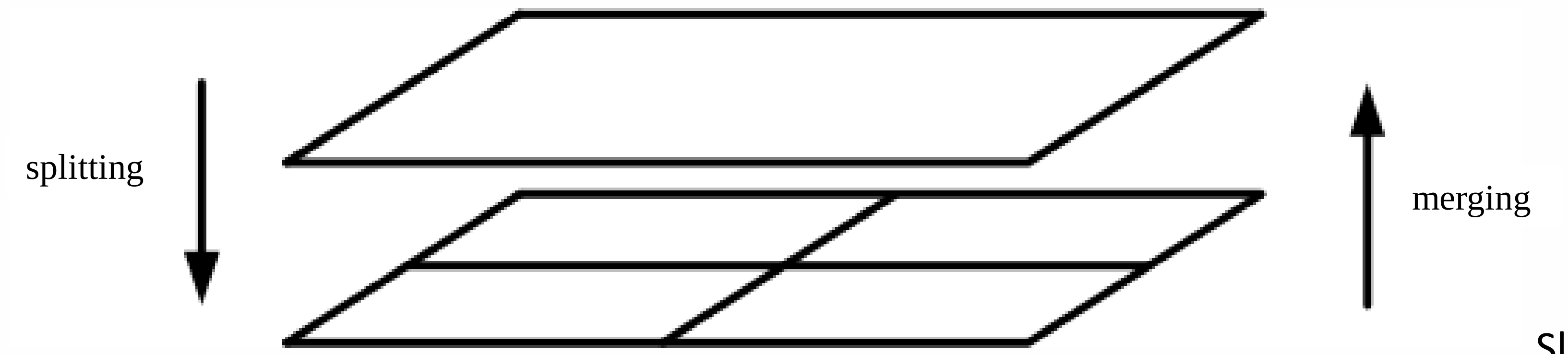
- The partitioning begins with the representation of the entire image as a simple region, which does not always meet the uniformity condition.
- During the segmentation process, the current regions of the image are sequentially split in accordance with the specified uniformity conditions.
- The methods of merging and splitting regions do not always lead to the same segmentation results, even if they use the same homogeneity criterion.

Splitting and Merging Regions



Splitting and Merging Regions

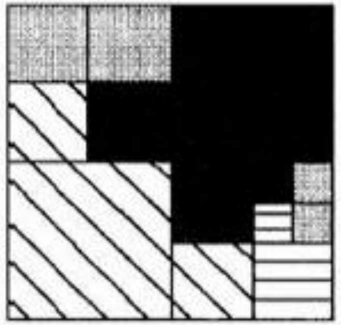
- The processes of splitting and merging regions are carried out alternately at each iteration.
- If any region at any pyramidal level is heterogeneous, it is split into four sub-regions.
- On the contrary, if at any level of the pyramid there are four adjacent regions with approximately the same amount of uniformity, they are merged into a single region at a higher level of the pyramid.

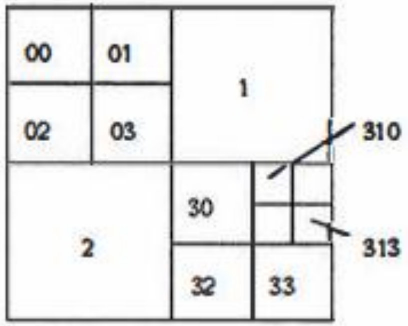


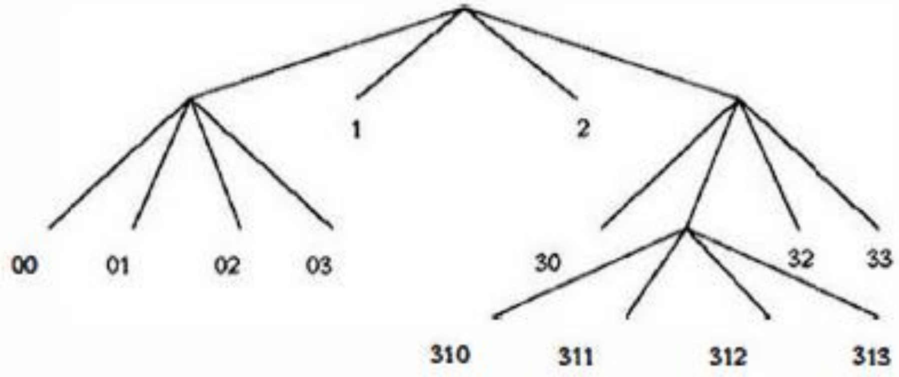
Splitting and Merging Regions

1. Carry out the initial segmentation of regions, determine the criterion of homogeneity and the pyramid of the data structure.
2. If:
   - any region R in the pyramid of the data structure is not homogeneous ($Pred(R) = FALSE$), we split it into four child regions,
   - any four regions having the same parents can be merged into a simple homogeneous region, then the regions are merged,
   - there are no more regions that could be divided or merged at this step, go to step 3.
3. If there are any two adjacent regions R_i, R_j, that can be merged into a homogeneous region, merge them.
4. We merge small regions with the largest similar neighboring region.

Splitting and Merging Regions







Basic Image Segmentation Methods

- **Threshold image segmentation by brightness levels.**
- Formula:

$$\begin{cases} g(i,j) = 1, \text{for } f(i,j) \geq T, \\ g(i,j) = 0, \text{for } f(i,j) < T, \end{cases}$$

where

$g(i,j)$ – the element of the resulting binary image,

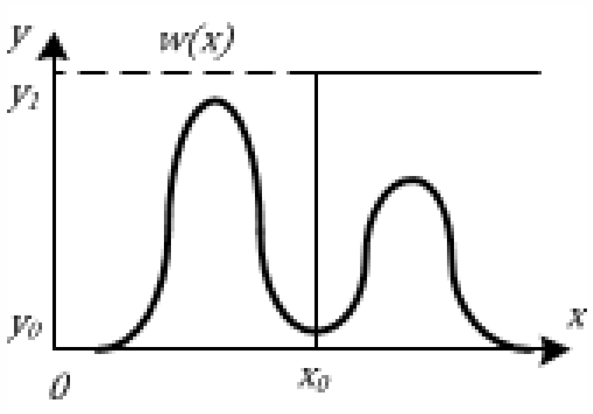
$f(i,j)$ – the element of the original image,

T – the brightness threshold value.

- **The main issue** is the definition of the segmentation threshold.

Segmentation Threshold Calculation

- $w(x)$ – image histogram, where $0 \leq x \leq 255$.



Threshold by the histogram

Basic Image Segmentation Methods

- **Range Threshold Segmentation**

$$\begin{cases} g(i,j) = 1, \text{for } f(i,j) \in D \\ g(i,j) = 0, \text{otherwise}, \end{cases}$$

where D is the range of values.

- **Multithreshold segmentation**

$$\begin{cases} g(i,j) = 1, \text{for } f(i,j) \in D_1 \\ g(i,j) = 2, \text{for } f(i,j) \in D_2 \\ \qquad\qquad \dots \\ g(i,j) = 0, \text{otherwise} \end{cases}$$

k-means Segmentation Algorithm

1. **Specify the number of classes** k into which the image should be divided. All pixels are considered as a set of vectors $\{x_i | \; i = 1, \dots, p\}$.
2. **Determine** k**-vectors** $\{m_j | \; j = 1, \dots, k\}$, which are declared as initial centers of clusters. Choose the values $\{m_j | \; j = 1, \dots, k\}$ (for example, randomly).
3. **Update the values** of the mean vectors $\{m_j | \; j = 1, \dots, k\}$, (cluster centers). For this:
   - calculate the distance from each $\{x_i | \; i = 1, \dots, p\}$ to each $\{m_j | \; j = 1, \dots, k\}$;
   - assign each x_i to the cluster j^*, the distance to the center of which m_{j^*} is minimal;
   - recalculate the average values m_j for all clusters.
4. **Repeat steps 2, 3** until the cluster centers stop changing.

Weber Segmentation Algorithm

- Formula:

$$W(I) = \begin{cases} 20 - \dfrac{12I}{88} & if\ 0 \le I \le 88 \\ 0{,}002(I-88)^2 & if\ 88 < I \le 138 \\ \dfrac{7(I-138)}{255-138} + 13 & if\ 138 < I \le 255 \end{cases}$$

where $W(I)$ is the Weber function**,**
I is the brightness value.

- **Weber principle:** a person does not distinguish between gray levels between $\left[I(n), I(n) + W\left(I(n)\right)\right]$.

Weber Segmentation Algorithm

1. Set first class number $n = 1$ and initial gray level $I(n) = 0$.
2. Calculate the value $W\big(I(n)\big)$ corresponding to the brightness $I(n)$ using the Weber formula.
3. In the original image I, set the brightness values $I(n)$ for all pixels whose brightness is in the range $\big[I(n), I(n) + W\big(I(n)\big)\big]$.
4. Find pixels whose brightness value is higher than $G = I(n) + W\big(I(n)\big) + 1$. If there are such pixels, increase the class number $n = n + 1$, $I(n) = G$, go to step 2. If there are none, finish the job.

- The image is segmented into n classes, each class is shown by the brightness $W\big(I(n)\big)$. It is convenient to implement this segmentation method by building a LUT table.

Iterative algorithm of Vezhnevets

1. Traversing the image from the top left pixel, which is the class C_1.
   - For the pixels of the first row, calculate the deviation from the class of the left pixel and compare with the specified threshold. If less than the threshold, add a pixel to the class of the neighbor, otherwise, create a new class C_{1+i}.
2. Compare the first pixel of each next row with the classes of the two neighbors: left and top.
   - If the deviation from both compared classes is greater than the threshold, then start a new class.
   - If the deviation is greater for only one class, then add a pixel to that class, the deviation from which is less than the threshold.
   - If the deviation is acceptable for both classes, two options are possible:
     1. $L\left(g(C_i) - g(C_j)\right) < \delta$ – combine these two classes (if they are not the same class) and add the current pixel to the combined class;
     2. $L\left(g(C_i) - g(C_j)\right) > \delta$ – add a pixel to the one of the two classes from which the deviation is minimal.

- As a measure of $\boldsymbol{L}$, you can use any distance function, for example, the difference in RGB space.

Segmentation by Skin Color

- **Advantages:** skin color is independent of face orientation; pixel color analysis is computationally efficient.

- **Task:** Choose a criterion for evaluating the proximity of the color of each pixel to the skin tone.

- **Design of skin color model**:
  1. Accumulate the training data using images that indicate "skin" and "non-skin" areas. Accumulate the skin tone statistics based on the training data.
  2. Process the obtained statistics and select the skin color model parameters for subsequent use. Select the criteria for evaluating whether pixels belong to the “skin” area.
  3. Process images with the obtained criteria.

Segmentation by Skin Color

- Threshold criteria, i.e. context-independent segmentation: the color of a pixel (R,G,B) is assigned to the "skin" area if the following conditions are met:

$$\begin{cases} R > 95 \\ G > 40 \\ B < 20 \\ R > G \\ R > B \\ max\{R, G, B\} - min\{R, G, B\} > 15 \\ |R - G| > 15 \end{cases}$$

Segmentation by Skin Color

- Threshold criteria, i.e. context-independent segmentation: the color of a pixel (R,G,B) is assigned to the "skin" area if the following conditions are met:

  or in flashlight conditions

$$\begin{cases} R > 220 \\ G > 210 \\ B > 170 \\ |R - G| \le 15 \\ G > B \\ R > B \end{cases}$$

Segmentation by Skin Color

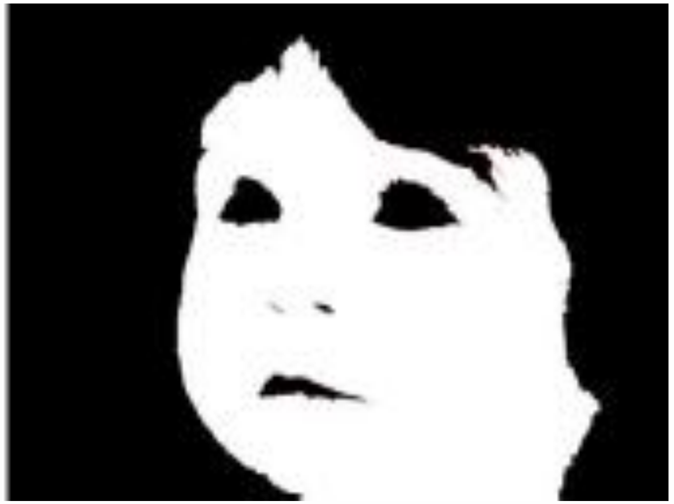
- Threshold criteria, i.e. context-independent segmentation: the color of a pixel (R,G,B) is assigned to the "skin" area if the following conditions are met:

  or with a normalized color

$$\begin{cases} r = \dfrac{R}{R+G+B} \\ g = \dfrac{G}{R+G+B} \\ b = \dfrac{B}{R+G+B} \\ \dfrac{r}{g} > 1.185 \\ \dfrac{rb}{(r+g+b)^2} > 0.107 \\ \dfrac{rg}{(r+g+b)^2} > 0.112 \end{cases}$$

Segmentation by Skin Color

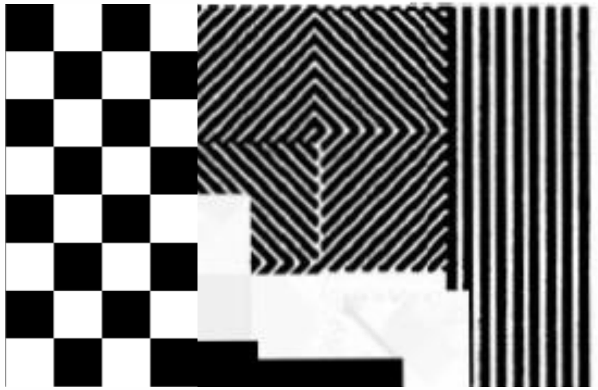
- It is advised to apply the median filter after segmenting by skin color.



Texture Segmentation

Approaches to texture segmentation:

1. **Statistical** – allows you to characterize the texture of the area as smooth, rough and grainy.
2. **Structural** - define and describe the relative position of the simplest repeating image elements, for example, segments of parallel lines passing at a constant step, cells on a chessboard.
3. **Spectral.**

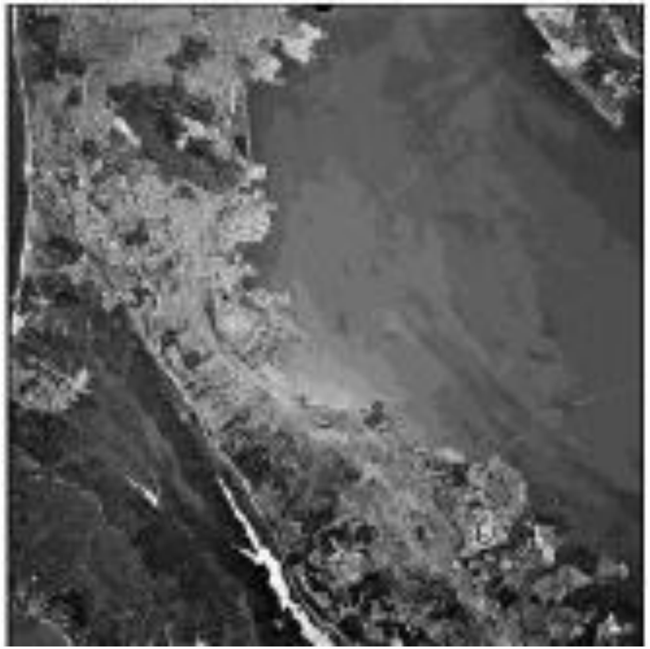


Texture Segmentation

Example

- Split an image containing two types of regions represented by different textures.
- It can be carried out with a statistical texture segmentation approach.
- As a result, the image will be split into water surface and land.

- This cannot be done by binarization methods, only by analyzing the texture parameters in the area around each pixel.


An image with different types of texture areas corresponding to land and water.

Texture Segmentation

- Texture Analysis Segmentation Algorithm:
  1. Read image.
  2. Define texture parameters. Assuming that the brightness in the pixels of the image is a random variable z, it corresponds to the distribution probability $p(z_i)$, taken from the histogram (L is the number of brightness levels).
- **The central moment** of order n of the random variable z_i is equal to:

$$\mu_n(z) = \sum_{i=0}^{L-1} (z_i - m)^n p(z_i),$$

where *m* is the average value of z (average brightness of the image),

$$m = \sum_{i=0}^{L-1} z_i p(z_i),$$

$\mu_0 = 1$ and $\mu_1 = 0$.

Texture Segmentation

- To describe the texture, the second point is important, i.e. variance: $\sigma^2(z) = \mu_2(z)$. It is a measure of luminance contrast, which can be used to calculate features of relative smoothness:

$$R = 1 - \frac{1}{1+\sigma^2(z)},$$

  - R is zero for areas of constant brightness (where the variance is zero),
  - R approaches unity for large values of $\sigma^2(z)$.

- In grayscale images, it is advised to normalize the dispersion to the interval [0,1]. To do this, you need to divide $\sigma^2(z)$ by $(L-1)^2$.

- **The standard deviation** is used as a characteristic of the texture: $s = \sigma(z)$

Texture Segmentation

- The third point is a characteristic of the symmetry of the histogram:

$$\mu_3(z) = \sum_{i=0}^{L-1} (z_i - m)^3 p(z_i).$$

- To estimate the spread in brightness of neighboring pixels, the entropy function is used:

$$e = -\sum_{i=0}^{L-1} p(z)\, log_2 p(z_i).$$

where $p(z_i)$ is the probability of the current brightness in the vicinity of the point;

L is the number of brightness levels; *e* is the entropy value at the current point.

Texture Segmentation

- To describe the texture, a uniformity measure is also used, which evaluates the uniformity of the histogram:

$$U = \sum_{i=0}^{L-1} p^2(z_i)\,.$$

- The table shows the values of the described characteristics, selected for smooth, rough and periodic textures.

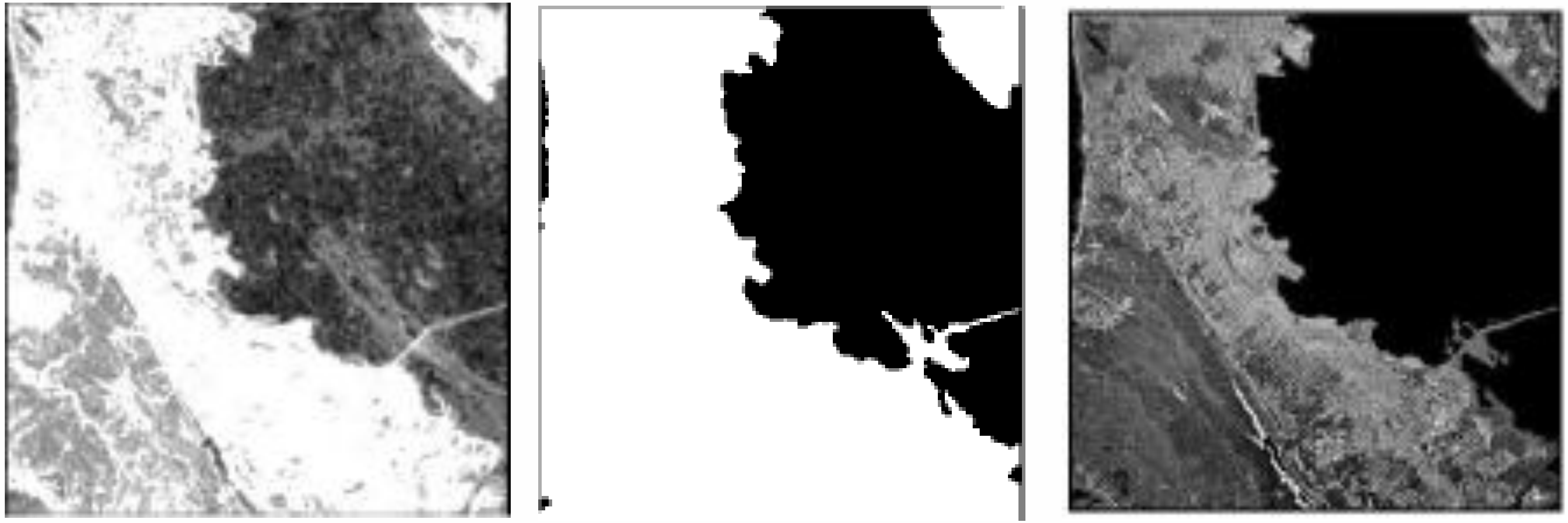
Texture	Average	Standard deviation	R (normalized)	Third moment	Uniformity	Entropy
Smooth	82.64	11.79	0.002	-0.105	0.026	5.434
Rough	143.56	74.63	0.0079	-0.151	0.005	7.783
Periodic	99.72	33.73	0.017	0.750	0.013	6.674

3. Create a mask to highlight the larger texture.

Texture Segmentation

- Let the image have textures of two types: large and small (grainy). Grainy corresponds to the water zone.
- To separate one area from another, create a mask that removes small objects.
  - To do this, use the function of determining a connected set of pixels in a binary image and calculate the areas of the resulting objects. Use connection type eight.
- If the color of the neighbors is the same, then they belong to the same object, otherwise they belong to different ones.
- All objects with an area less than a given value *S* are deleted.

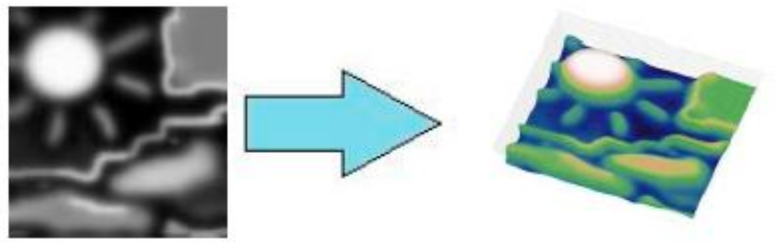
Texture Segmentation



In the left is the result of texture filtering based on entropy calculation in a 9x9 window. Regions of the two textures are shown as dark (water) and light (land) hues; in the center – water surface mask after removal of small region objects; on the right is the result of land segmentation.

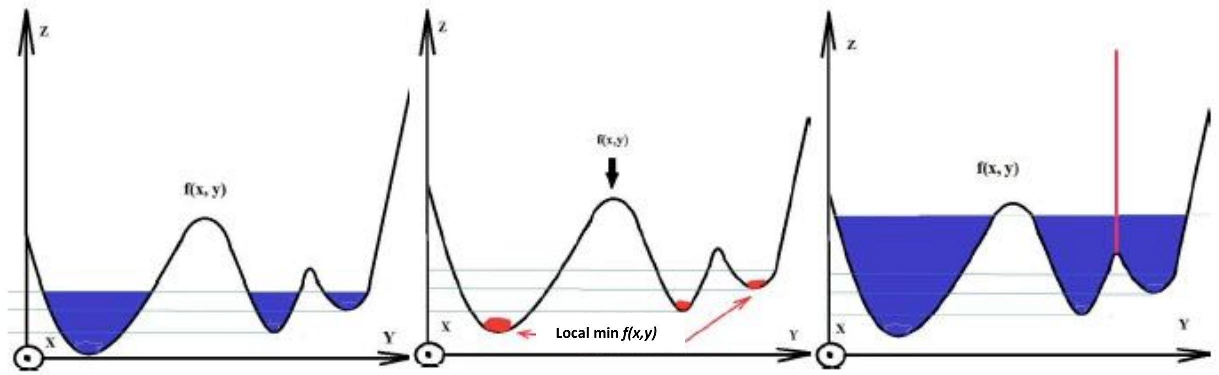
Morphological Watershed Method

- A grayscale image is a digital terrain model, where the brightness values are heights relative to a certain level, i.e. image is a matrix of heights.



Morphological Watershed Method

- If it rains on such an area, many pools are formed. Water fills small pools, then overflows from overflowing pools and the pools combine into larger pools according to the heights of the water level.
- The places where pools merge are marked as watershed lines. As a result, the entire area may be flooded.
- The result of segmentation depends on the moment when the water supply stops. If the process is stopped early, the image will be segmented into small areas, if it is stopped late, into very large ones.



Morphological Watershed Method

- All pixels are divided into three types:
  1. **local minima**;
  2. **located on a slope,** i.e. those from which water rolls into the same local minimum;
  3. **local maxima,** i.e. those from which water rolls into more than one minimum.
- When segmenting using this method, it is necessary to determine the watersheds and watershed lines in the image by processing local areas depending on their brightness characteristics**.**

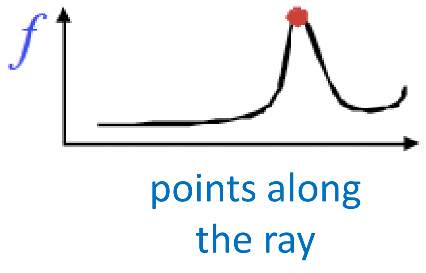
Morphological Watershed Method

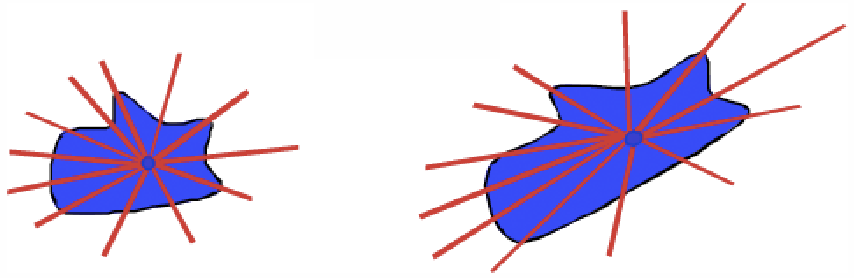
- Algorithm for implementing the watershed method:
  1. **the segmentation function is calculated** (this applies to images where objects are placed in dark regions and are difficult to distinguish);
  2. **foreground markers of the image are calculated** based on the analysis of pixel connectivity of each object;
  3. **background markers are calculated,** which are pixels that are not part of objects;
  4. **the segmentation function is modified** based on the location values of background markers and foreground markers.
  5. **selection against the background of the image of objects of uniform brightness** (in the form of spots).
- Regions characterized by small brightness variations have small gradient values. Therefore, in practice, the watershed segmentation method is usually applied not to the image itself, but to its gradient representation.

Region Detectors

- **IBR detector (Intensity-extrema based regions)**
- It is necessary to go from the points of the local brightness extremum I_0 along the rays, calculating some value f.
- As soon as the peak of the value f is found, it is necessary to stop. This point will be the boundary of the region.

$$f(t) = \frac{|I(t) - I_0|}{\frac{1}{t}\int_0^t |I(t) - I_0| dt}.$$

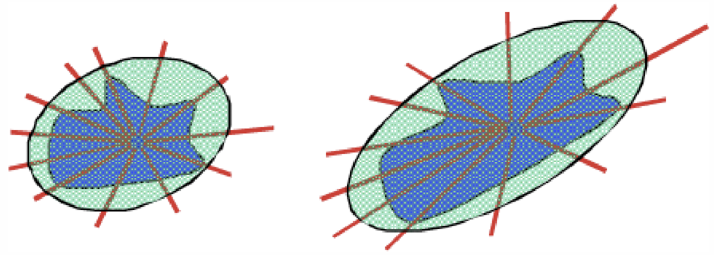


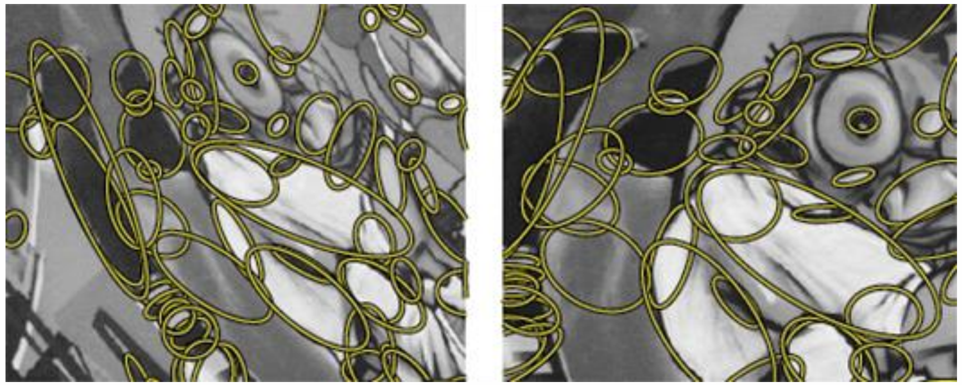


An example of the IBR detector

IBR Detector

- The areas on a pair of similar images may differ, so we describe ellipses around them.
- If the ellipses are turned into circles, then we get complete similarity up to rotation.

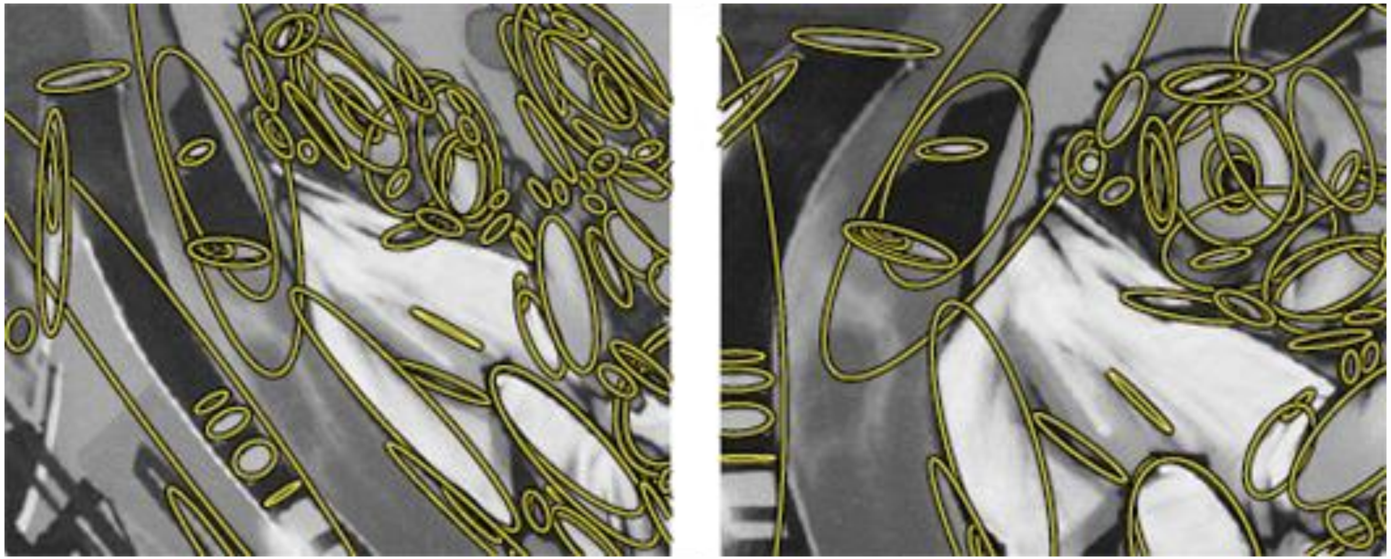

Circumscribed ellipses around objects


An example of the IBR detector

MSER Detector

- **Maximally Stable Extreme Regions**
- Solves the problem of invariance of keypoints when scaling.
- **MSER detector algorithm**:
    1. Sort the set of all image pixels in ascending/descending order of intensity.
    2. Construction of a pyramid of connected components. For each pixel of the sorted set, perform the following sequence of actions:
        - updating the list of points included in the component;
        - updating the areas of the next components, as a result of which the pixels of the previous level will be a subset of the pixels of the next level.
    3. For all components, search for local minima (we find pixels that are present in this component, but are not part of the previous ones). The set of local level minima corresponds to the extreme region in the image.

MSER Detector



An example of the MSER detector

Test

Lecture 1-2 Test

Please scan the code to start the test

THANK YOU
FOR YOUR TIME!

ITsMOre than a UNIVERSITY

Andrei Zhdanov
adzhdanov@itmo.ru