Karlsruher Institut für Technologie

Hochperformante Humanoide Technologien

Robotics III: Sensors and Perception in Robotics

Chapter 05: Feature Extraction

Tamim Asfour

http://www.humanoids.kit.edu

Motivation

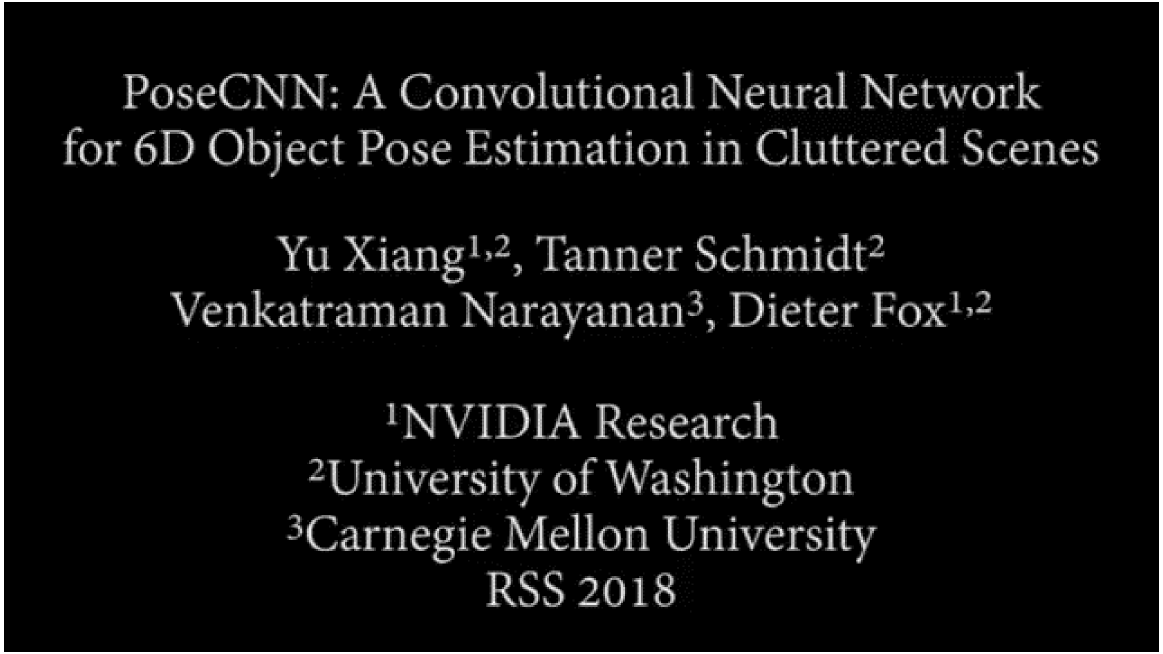
A humanoid robot designed for grasping of objects in a real-world scenario sets high requirements on visual object recognition and pose estimation

Motivation

- What is the problem?

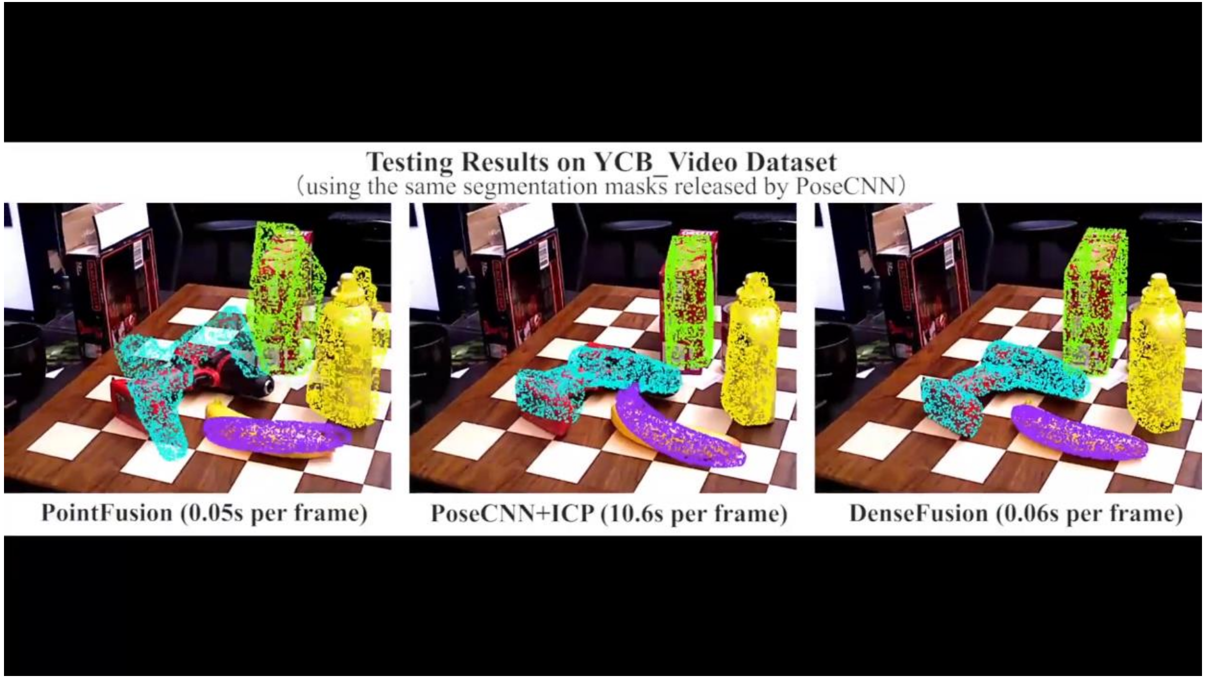
- **High dimensional visual information** from cameras has to be transferred to a **high-level description language**
  - What is an object?
  - What is the pose of the object?
- Objects have to be recognized in an arbitrary scene
  - Invariance regarding light conditions
  - Rotation
  - Scaling
  - Affine transformation
- Reasonable time

Motivation – PoseCNN



Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, Dieter Fox; PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes; https://arxiv.org/abs/1711.00199

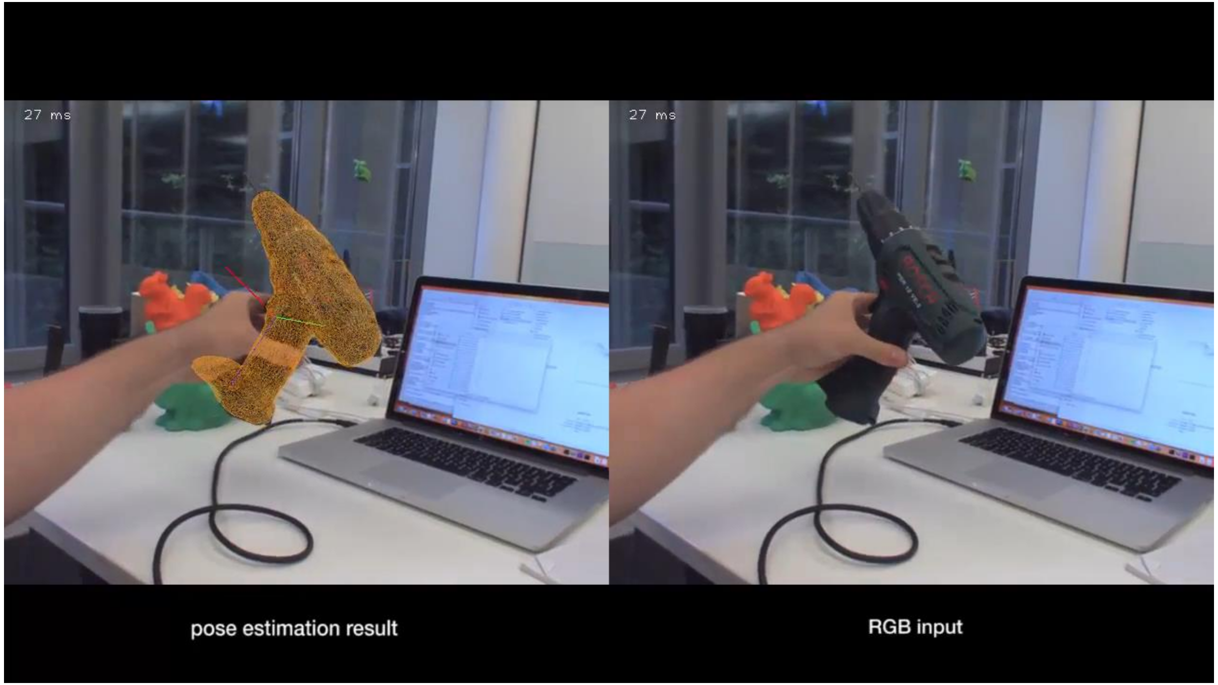
Motivation – Dense Fusion



Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, Silvio Savarese "DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.

Motivation – DenseFusion for Grasping

Motivation – Region-based Object Tracking



H. Tjaden, U. Schwanecke and E. Schömer, "Real-Time Monocular Pose Estimation of 3D Objects Using Temporally Consistent Local Color Histograms," *2017 IEEE International Conference on Computer Vision (ICCV)*

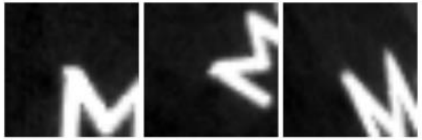
Feature Extraction

- Image processing operations
  - Input: one or several images
  - Output: image

- Feature extraction
  - Input: Image
  - Output: one or several image features (scalars or “short” vectors)
  - Examples of image features
    - 6D pose of an object
    - Parameter of a line
  - Classes of features
    - Region features (redness)
    - Line features (doors, buildings, roads)
    - Interest points, salient points, corner points (point features)

Outline

- **Correlation Functions**
- Corner Detectors
  - Moravec operator
  - Harris Corner detector
  - Good Features to Track
  - Machine-learned features
- Feature Descriptors
  - Simple descriptors
  - SIFT
  - SURF
  - MSER
- Pose Estimation
  - Monocular
  - Stereo images
  - Depth images
  - Neural networks



Correlation Methods

- Determine correspondences between images or image patches I_1 and I_2
- Used for:
  - Solving of correspondence problem in stereo vision
  - Object recognition
  - Image-based localization

See lecture Robotics-I

- Non-normalized correlation functions
  - Change depending on the illumination
- Normalized correlation functions
  - Invariant with respect to constant additive or multiplicative brightness differences
- In the following we will consider **squared grayscale images**

Non-normalized Correlation Functions

- **Non-normalized correlations functions** for two square grayscale images I_1 and I_2

$$c(I_1, I_2) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} f(I_1(u,v), I_2(u,v))$$

- Correlation-function c for images I_1, I_2 at position (u_0, v_0) with displacement (d_u, d_v) in a squared window of size $k \times \mathrm{k}$:

$$c(I_1, I_2, u_0, v_0, d_u, d_v) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} f(I_1(u_0 + u, v_0 + v), I_2(u_0 + d_u + u, v_0 + d_v + v))$$

- Function $f(\cdot)$ is determined by the correlation method

Non-normalized Correlation Functions II

- **Sum of Squared Differences (SSD):** $f(x,y) = (x-y)^2$

$$SSD(I_1, I_2) = \sum_u \sum_v (I_1(u,v) - I_2(u,v))^2$$

  - Squared Euclidean metric; not robust with respect to outliers, not invariant to different brightness

- **Sum of Absolute Differences (SAD):** $f(x,y) = |\, x - y \,|$

$$SAD(I_1, I_2) = \sum_u \sum_v |I_1(u,v) - I_2(u,v)|$$

  - Manhattan-metric; more robust with respect to outliers; not invariant to different brightness

Normalized Correlation Functions

- Extension to compensate additive constant brightness level shift d:
$$I_1(u,v) + d = I_2(u,v)$$
- Normalization:
  - Arithmetic mean of an image I
$$\bar{I} = \frac{1}{n^2}\sum_u \sum_v I(u,v)$$
  - Subtraction of mean-value ("zero-mean" normalization)
$$I_2^{'} = I_2\,(u,v) - \bar{I_2} = I_2(u,v) - \frac{1}{n^2}\sum_u \sum_v I_2(u,v)$$
$$= I_1(u,v) + d\, - \left(\frac{1}{n^2}\sum_u \sum_v I_1(u,v) + d\right) = I_1(u,v) - \bar{I_1} = I_1^{'}$$
  - Robust against constant brightness offset

Normalization

- Normalized correlations functions to compensate multiplicative brightness level shift

$$I_1(u,v) \cdot r = I_2(u,v)$$

- Normalisation
  - Frobenius norm :

$$\|I\|_H = \sqrt{\sum_u \sum_v I^2(u,v)}$$

  - Normalization by Frobenius norm

$$I_2' = \frac{I_2(u,v)}{\|I_2\|_H} = \frac{I_2(u,v)}{\sqrt{\sum_u \sum_v I_2^2(u,v)}} = \frac{I_1(u,v) \cdot r}{\sqrt{\sum_u \sum_v (I_1(u,v) \cdot r)^2}} = \frac{I_1(u,v)}{\sqrt{\sum_u \sum_v I_1(u,v)}} = \frac{I_1(u,v)}{\|I_1\|_H} = I_1'$$

Normalized Correlation Functions

- Forbenius norm of additive normalized image:

$$\|I'\|_H = \sqrt{\sum_u \sum_v (I(u,v) - \bar{I})^2}$$

- Example
  - Zero-Mean Normalized Sum of Squared Differences (ZNSSD)

$$ZNSSD(I_1, I_2) = \sum_u \sum_v \left[\frac{I_1(u,v) - \bar{I}_1}{\|I_1'\|_H} - \frac{I_2(u,v) - \bar{I}_2}{\|I_2'\|_H}\right]^2$$

  - Zero-Mean Normalized Sum of Absolute Differences (ZNSAD)

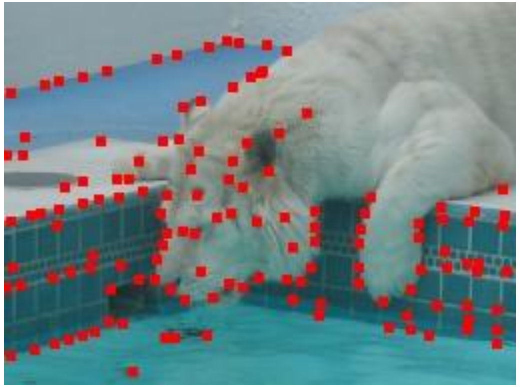
$$ZNSAD(I_1, I_2) = \sum_u \sum_v \left|\frac{I_1(u,v) - \bar{I}_1}{\|I_1'\|_H} - \frac{I_2(u,v) - \bar{I}_2}{\|I_2'\|_H}\right|$$

Outline

- Correlation Functions
- **Corner Detectors**
  - Moravec operator
  - Harris Corner Detector
  - Good Features to Track
  - Machine-learned Features
- Feature Descriptors
  - Simple Descriptors
  - SIFT
  - SURF
  - MSER
- Pose Estimation
  - Monocular
  - Stereo Images
  - Depth Images
  - Neural Networks

Moravec Operator

- Developed 1977 by Hans P. Moravec
- Goal:
  - Recognize regions of interest in consecutive camera-images
- Interest Points concept
  - An interest point is defined as a point where a sliding window filter has strong variations when moved in any direction
- Use of autocorrelation-function



http://www.roborealm.com/help/Moravec.php

Moravec Operator (II) – Steps

- Focus area is a small quadratic window (e.g. 3×3 or $5 \times\ 5$) and a point (u, v) in the centre
- Window is moved in four pre-defined directions (horizontal, vertical, diagonal) and compared with basis value
- Difference between original and moving window is calculated with SSD (Sum of Squared Differences):

$$D(u,v,s,t) = \sum_{(u_i,v_i)\in W(u,v)} \left(I(u_i+s,v_i+t) - I(u_i,v_i)\right)^2$$

$W(u,v)$ is the quadratic window with centre (u,v)

$(s,t) \in \{(1,0),(0,1),(1,1),(-1,1)\}$

Moravec Operator (III) – Possible cases

- **Case 1:** Value of D is for all translations low
  → test window is in a (nearly) **homogenous** area

- **Case 2:** Value of D along a certain direction R is low,
  for translation orthogonal to R the value is high
  → test window contains an **edge** along R

- **Case 3:** Value of D for a translation in any direction is high
  → test window contains a **corner** (Interest Point)

Moravec Operator IV - Algorithm

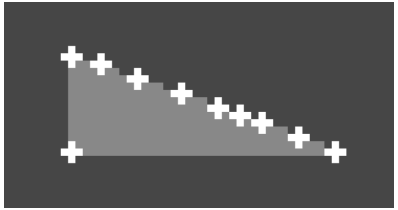
- The test window is shifted over the entire image
- The metric has to return low values in case 1 and 2 and high values in case 3 (corner)
- **Input:** grayscale image $I(u, v)$, threshold k
- **Output:** Set M of calculated interest points

$M := \varnothing$
for all pixels (u, v) *in* I *do*
    $m := inf.$
    for all (s, t) *in* S *do*
        $m := \min\{m, D(x, y, s, t)\}$
    if $m \geq k$ *then*
        $M := M \cup \{(u, v)\}$
return M

Moravec Operator V - Disadvantages

- Non-isotropic operator response:
  - Result of the Moravec operator depends on the shift-direction
  - Since only four directions are tested the result cannot be invariant to rotation
- Noisy operator response
  - The window is binary and quadratic
  - Pixels located in the corner have the same weight, which may cause error
- Strong response to a point on an edge:
  - Operator is sensitive to corner points, that have a slight deviation to the predefined shift-directions

Typical Moravec operator result:
Finds points on corners **and** noisy edges



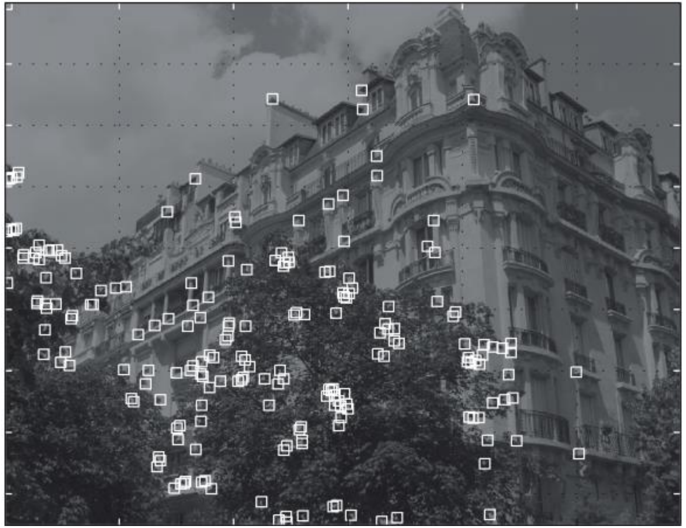
Harris Corner Detector

Karlsruher Institut für Technologie

- Developed in 1988 by
  Chris Harris and Mike Stephens

- **Goal:** Replace the four predefined directions in the Moravec operator with smaller step size

- **Approach:** Use first order Taylor series of the image function



Peter Corke: Robotics, Vision and Control, Fundamental Algorithms in MATLAB®, Springer 2011

Harris Corner Detector II

- Image function is approximated with Taylor expansion
- First order Taylor series:

$$I\ (u+s, v+t) \approx I(u,v) + \begin{pmatrix} I_x(u,v) & I_y(u,v) \end{pmatrix} \cdot \begin{pmatrix} s \\ t \end{pmatrix}$$

I_x and I_y are directional derivatives, which can be calculated with Prewitt- or Sobel operator.

Harris Corner Detector III

- Use of Taylor function for $D(u, v, s, t)$ (Moravec-Operator) results in:

$$D(u,v,s,t) = \sum (I(u_i + s, v_i + t) - I(u_i, v_i))^2$$

$$\approx \sum \left(I(u_i, v_i) + \begin{pmatrix} I_x(u_i, v_i) & I_y(u_i, v_i) \end{pmatrix} \cdot \begin{pmatrix} s \\ t \end{pmatrix} - I(u_i, v_i) \right)^2$$

$$= \sum \left(\begin{pmatrix} I_x(u_i, v_i) & I_y(u_i, v_i) \end{pmatrix} \cdot \begin{pmatrix} s \\ t \end{pmatrix} \right)^2$$

$$= \sum \left(\begin{pmatrix} s & t \end{pmatrix} \cdot \begin{pmatrix} I_x^2(u_i, v_i) & I_x(u_i, v_i) \cdot I_y(u_i, v_i) \\ I_x(u_i, v_i) \cdot I_y(u_i, v_i) & I_y^2(u_i, v_i) \end{pmatrix} \cdot \begin{pmatrix} s \\ t \end{pmatrix} \right)$$

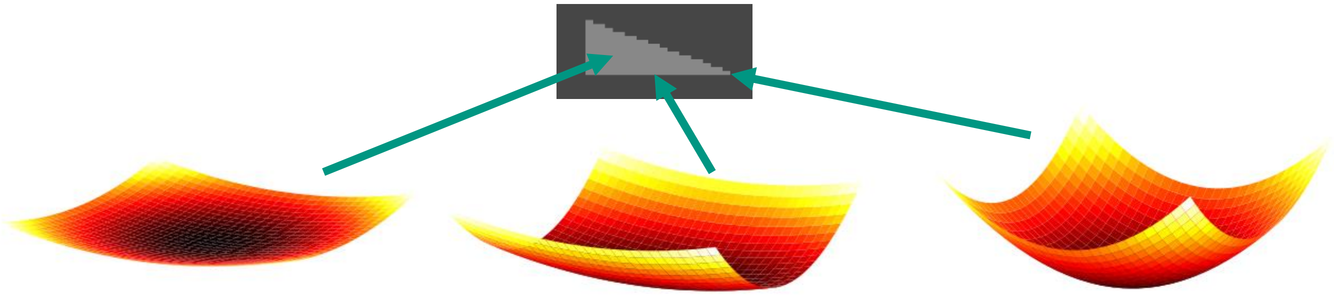
$$= \begin{pmatrix} s & t \end{pmatrix} \cdot M(u, v) \cdot \begin{pmatrix} s \\ t \end{pmatrix}$$

Image structure tensor

$$M(u,v) = \begin{pmatrix} \sum I_x^2(u_i, v_i) & \sum I_x(u_i, v_i) I_y(u_i, v_i) \\ \sum I_x(u_i, v_i) I_y(u_i, v_i) & \sum I_y^2(u_i, v_i) \end{pmatrix}$$

Harris Corner Detector IV

- The image structure tensor $M(u, v)$ is a 2×2 matrix computed from image derivatives
- It corresponds to an approximation of the local auto-correlation function



Flat region   Edge   Corner

- Eigenvalues λ_1 and λ_2 of M give information about distribution of gradients
  - **Flat region**: λ_1 and λ_2 small; Contour lines are a large ellipse
  - **Edge region**: $\lambda_1 \gg \lambda_2$ or vice versa; stretched ellipse
  - **Corner region**: λ_1 and λ_2 large; small ellipse

Recap: Eigenvalue & Eigenvector

- Eigenvectors $\boldsymbol{x}$ with Eigenvalues λ are defined as:

$$A \cdot x = \lambda \cdot x$$

- Values can be computed by solving

$$\det(A - \lambda I) = 0$$

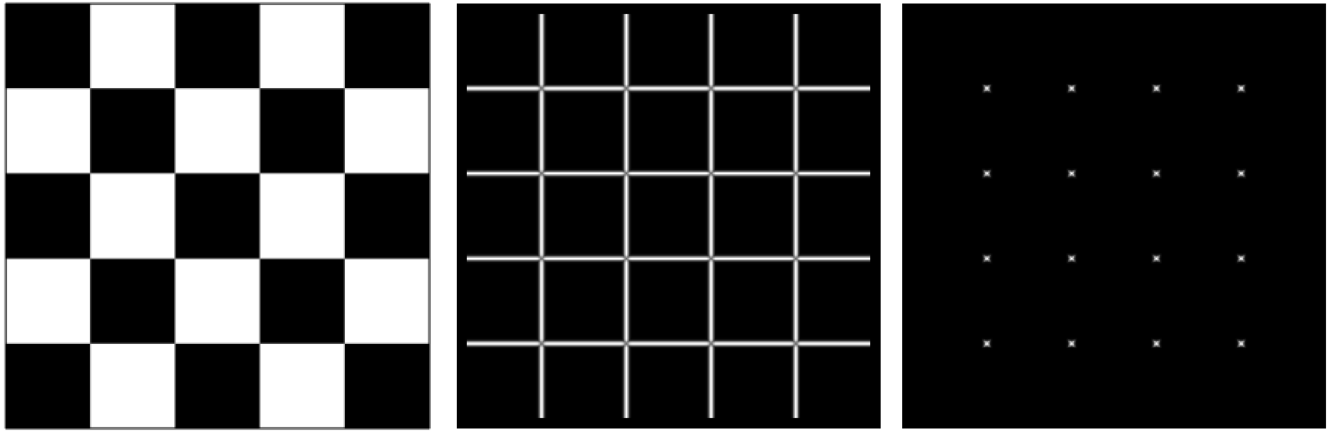
- For $M(u, v)$ the solution is given by

$$\lambda_{\pm} = \tfrac{1}{2}[(m_{11} + m_{22}) \pm \sqrt{4m_{12}m_{21} + (m_{11} - m_{22})^2}]$$

- The values of $\lambda_{\pm}$ and $x_{\pm}$ indicate the amplitude and direction of the largest/smallest change in $D(u, v, s, t)$

Eigenvalues on Corners

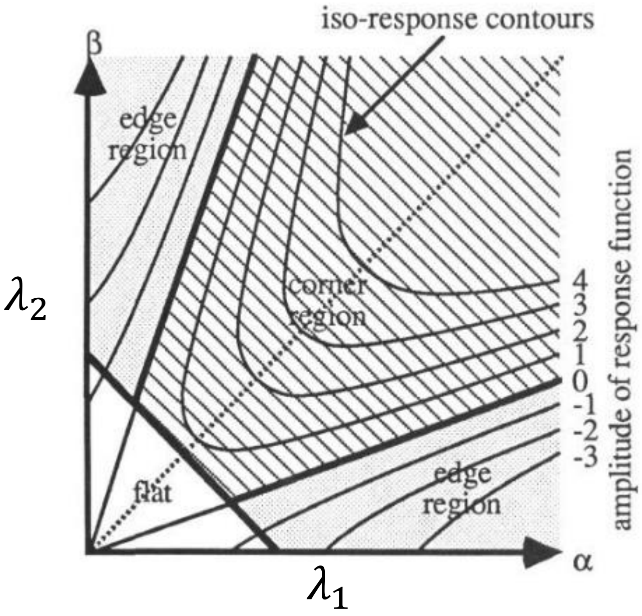
- For corners, the interesting value is λ_-
  - Large value indicates that the gradient is large in any direction
  - Therefore, it must be a corner



I $\qquad\qquad$ λ_+ $\qquad\qquad$ λ_-

Harris Corner Detector V

- Regions in $\lambda_1\lambda_2$-space give corner/edge/flat classification:



Harris, Chris, and Mike Stephens. "A combined corner and edge detector." *Alvey vision conference*. Vol. 15. No. 50. 1988.

Harris Corner Detector VI

- Eigenvalue decomposition has expensive computation
- Alternative measure of corner response proposed by Harris/Stephens:

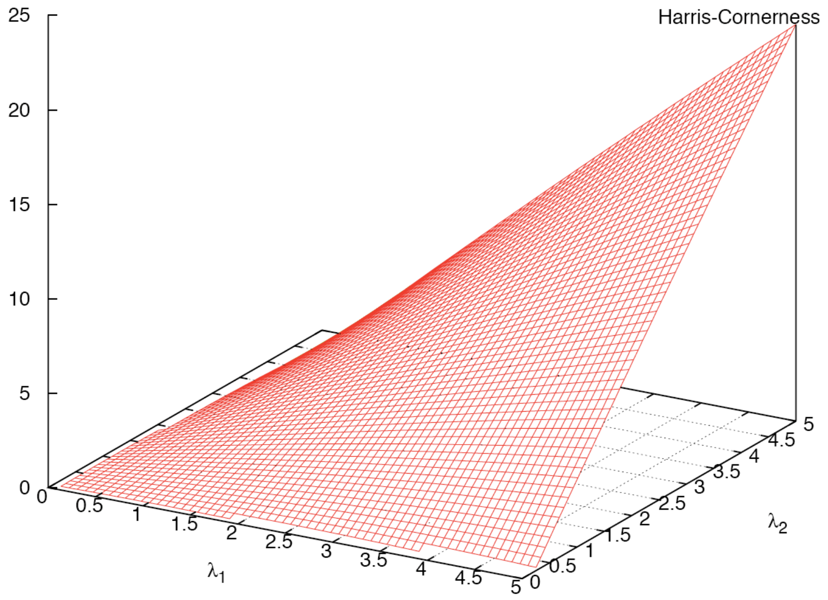
$$\begin{aligned} C(u,v) &= \lambda_1\lambda_2 - \kappa(\lambda_1+\lambda_2)^2 \\ &= \det M(u,v) - \kappa\big(\operatorname{trace} M(u,v)\big)^2 \\ &= m_{11}m_{22} - m_{12}m_{21} - \kappa(m_{11}+m_{22})^2 \end{aligned}$$

- κ is determined empirically and usually in the range between 0.04 and 0.15

- No eigenvalue decomposition of M; instead, evaluate the determinant and trace of the M

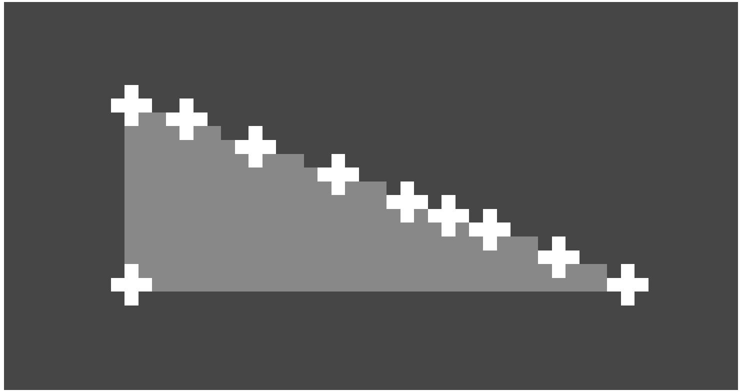
Harris Corner Detector VII

- Corners are assigned when local maxima are found
- Harris Corner Response for $\kappa = 0.04$:

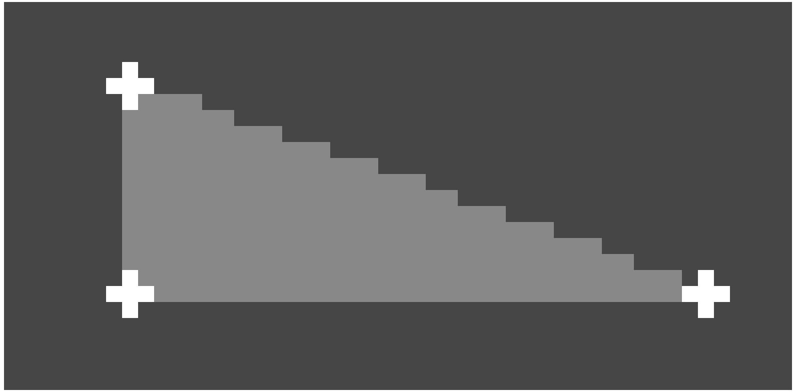


Harris Corner Detector VIII

- Example: Harris Corner Detector solves the problems of Moravec Operator



Moravec Corner



Harris Corner

Harris Corner Detector IX

- Example on real image with OpenCV in Python:

```
img = cv2.imread(filename)
gray = np.float32(cv2.cvtColor(img,cv2.COLOR_BGR2GRAY))
dst = cv2.cornerHarris(gray,2,3,0.04)
```

Result of cornerness function

Values above threshold colored red:

```
img[dst>0.02*dst.max()]=[0,0,255]
```

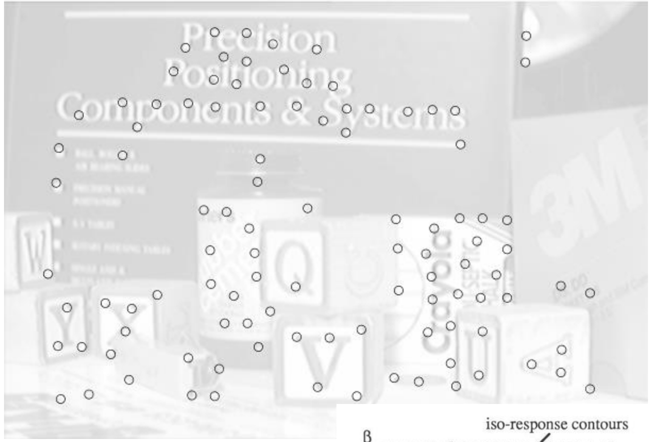
Good Features To Track

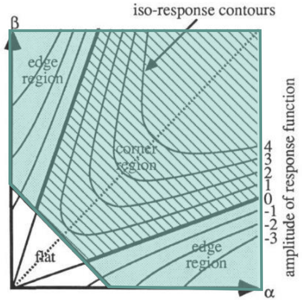
- Developed in 1994 by Jianbo Shi and Carlo Tomasi
- Improved version of Harris Corner Detector
- Eigenvalues are calculated explicitly
- Condition for feature:

$$\min(\lambda_1, \lambda_2) > \lambda$$

Both eigenvalues have to be above a threshold instead of threshold for cornerness function of Harris (similar to Moravec)

Corners are more stable for tracking





Shi, Jianbo. "Good features to track." *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994.

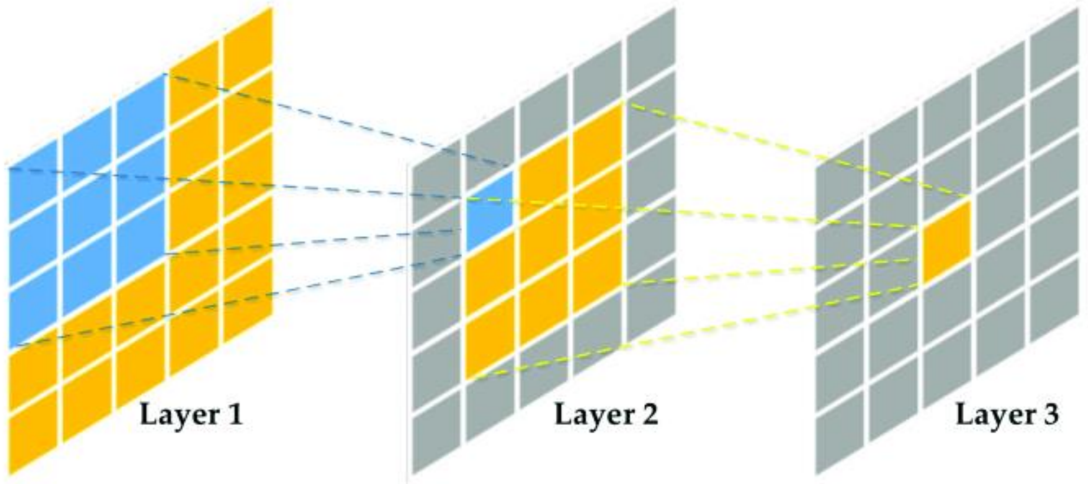
Machine-learned Features (1)

- Until recently features were chosen by experts
- Use of convolution and machine-learned filters for feature extraction:
  Convolutional Neural Networks (CNNs)

- Training of CNNs
  - Given image (input) and correct label/classification (output)
  - Backpropagation with loss function (compare actual output with correct output)

Machine-learned Features (2)

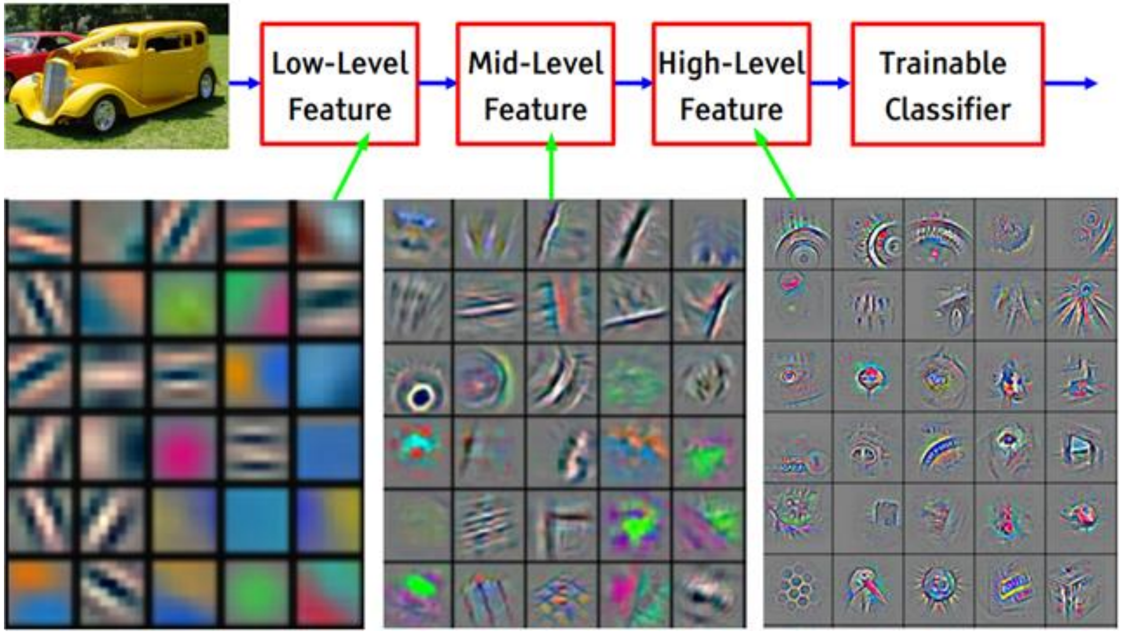
- Stack of multiple layers



Source: [MSF-Net: Multi-Scale Feature Learning Network for Classification of Surface Defects of Multifarious Sizes]

Machine-learned Features

- Stack of multiple layers: Hierarchically concatenated features



© RSIP https://www.rsipvision.com/exploring-deep-learning, 2020

Outline

- Correlation Functions
- Corner Detectors
  - Moravec operator
  - Harris Corner Detector
  - Good Features to Track
  - Machine-learned Features
- **Feature Descriptors**
  - Simple Descriptors
  - SIFT
  - SURF
  - MSER
- Pose Estimation
  - Monocular
  - Stereo Images
  - Depth Images
  - Neural Networks

Feature Descriptors

- To identify correspondences between extracted features in images (e.g., for object detection, pose estimation, etc.) a unique description for a feature is required

- **Feature Detector**:
  - Algorithm that detected locations of Points of Interest in an image
- **Feature Descriptor**
  - Algorithm that provides a feature vector (descriptor) of Points of Interest in an image
  - Descriptors represent “numerical fingerprints” of the features

Simple Descriptors

- Most simple approach:
  - Description of a local feature as the **quadratic window** around the feature centre (key point = image section)
  - Matching of 2 features with **correlation function**

- Pros:
  - Easy to implement
  - Low computational cost

- Cons:
  - Not invariant to changes in scale or rotation
  - Memory inefficient (resource limited systems)

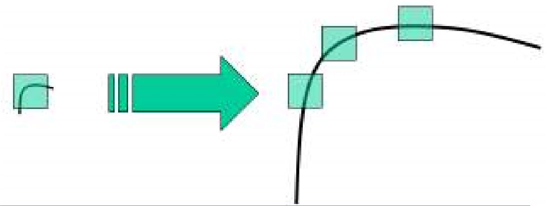
Simple Descriptors II

- Robust and compact description of key points by Lepetit et al. 2004
  - Description of image section by **view set**
  - Generation of **synthetic views** of key points by **random affine transformations**

  - **Illumination changes** handled by **normalizing of intensities** of all patches
    - Same minimal and maximal value in all patches → improved contrast
  - High memory demand (multiple descriptors for each feature)
  - Large computational effort (many descriptors to compare)

Scale Invariant Feature Transform (SIFT) – I

- Detected features might change if the image is rotated or scaled
  - Example: Harris Corner Detector is invariant to rotation but not to scaling



https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

- SIFT detects keypoints that are invariant to orientation and scale
- **Approach:**
  1. Scale-space extrema detection
  2. Keypoint localization
  3. Orientation assignment
  4. Keypoint descriptor

Lowe, David G., Distinctive Image Features from Scale-Invariant Keypoints, IJCV 2004

Scale Invariant Feature Transform (SIFT)

- Developed in 1999 by David G. Lowe, refined in 2004; very popular

- Approach (overview):
  - Find interest points using the SIFT detector:
    - Filter image with **difference of Gaussian** (DoG) kernels
    - Stack the filtered images and identify extrema (**Gaussian pyramid**)
    - Find best candidates
  - Calculate SIFT descriptor
    - Divide region into cells, calculate **gradient orientations**
    - Generate **histograms**

Scale Invariant Feature Transform (SIFT)

- Regions around a feature point are characterized partially **invariant to rotation and scaling** in a certain range
- **Invariant** to **intensity** and **contrast** changes and small **geometric deformations**

- **Algorithm**
  1. Scale-space extrema detection
  2. Keypoint localization
  3. Orientation assignment
  4. Keypoint descriptor

Scale-Space Extrema Detection

Karlsruher Institut für Technologie

- **Create Gaussian pyramid:** Scale space of an image $I(u,v)$ as **convolution** with a variable-scale Gaussian (→ blurred images)

$$L(u,v,\sigma) = G(u,v,\sigma) * I(u,v)$$

Gaussian Kernel:

$$G(u,v,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(u^2+v^2)/2\sigma^2}$$

- Keypoints are scale-space **extrema** in Difference-of-Gaussian (DoG) space convolved with the image (two scales separated by factor k):

$$\begin{aligned} D(u,v,\sigma) &= (G(u,v,k\sigma) - G(u,v,\sigma)) * I(u,v) \\ &= L(u,v,k\sigma) - L(x,y,\sigma) \end{aligned}$$

- DoG is a more efficient approximation of scale normalized LoG-Operator (Laplacian of Gaussian)

H2T

Scale-Space Extrema Detection

- **Construction of $D(x, y, \sigma)$:** The initial image is incrementally convolved with Gaussians to produce images separated by a constant factor k in scale space, shown stacked in the left column

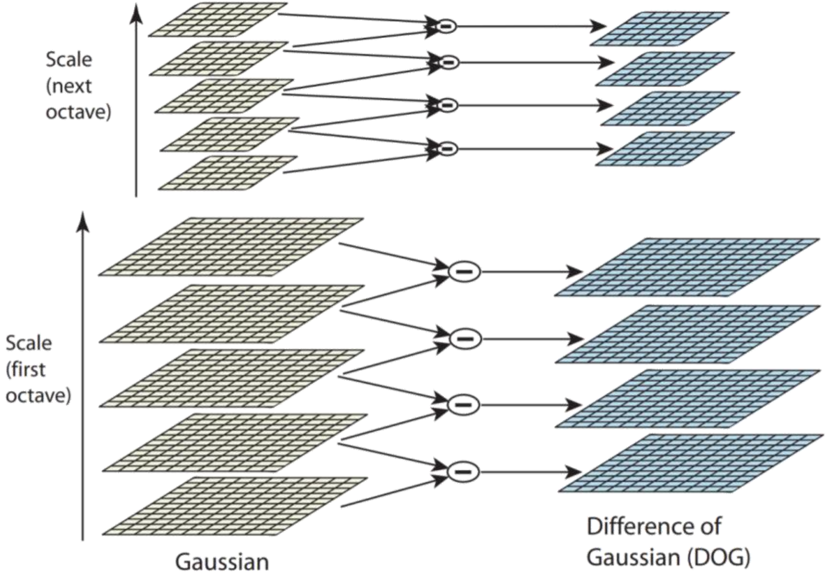
$\sigma = 2^{4/s}$

$\sigma = 2^{3/s}$

$\sigma = 2^{2/s}$

$\sigma = 2^{1/s}$

$\sigma = 1$

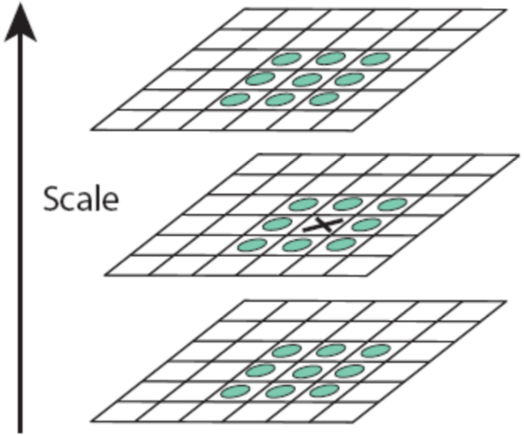


s is number of images per octave (here $s = 5$)

Each octave's image size is half of the previous one.

SIFT - Keypoint Localization

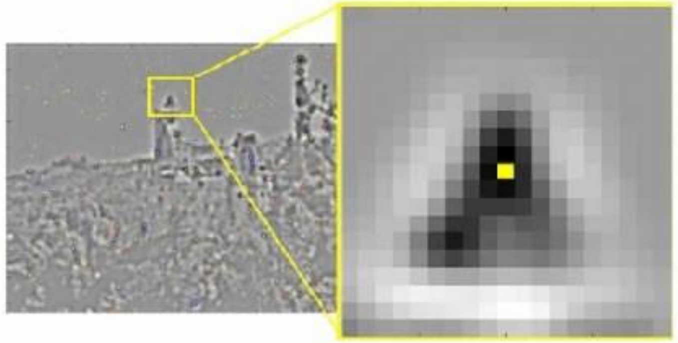
- Detect local extrema in scale space:
  - Each sample point is compared to its **26 neighbors** (8 neighbors in the current image and 9 neighbors each in the scales above and below)
  - A point is selected as local extrema only if it is larger than all of these neighbors or smaller than all of them
  - For each extrema (max or min) found, output is the **location** and the **scale**
  - Extrema can be localized with **sub-voxel accuracy** using the using a Taylor-Series expansion of $D(x, y, \sigma)$



Orientation Assignment

- Detected keypoints are connected to the scale at which they were found → scale invariance
- Rotation invariance is obtained by assigning an orientation to each keypoint

- **Idea:**
  - Calculate gradients in DoG of the keypoint
  - Assign dominant gradient orientation to keypoint


A keypoint

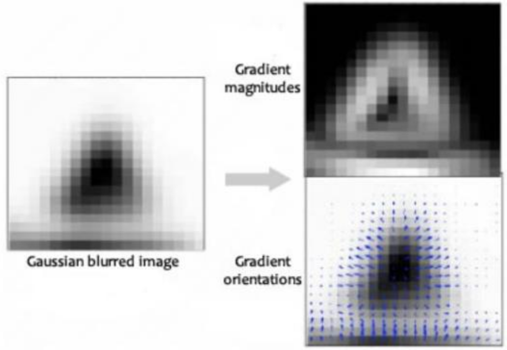
Orientation Assignment

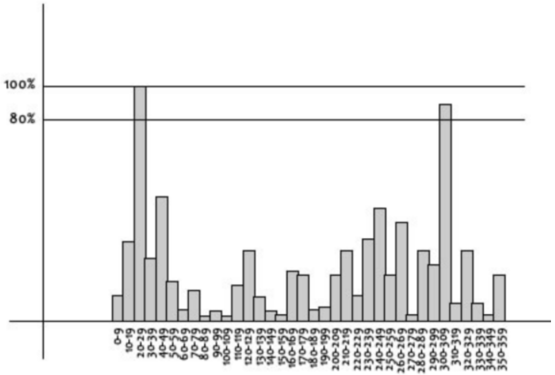
Calculation of dominant gradient orientations:

- **Step 1:** Calculate gradients in horizontal and vertical directions in quadratic 16x16 pixel window (Gauss-weighted)
- **Step 2:** Calculate gradient orientation θ and amplitude m:

$$m = \sqrt{g_x^2 + g_y^2} \qquad \theta = \arctan \frac{g_y}{g_x}$$

- **Step 3:** Calculate a **histogram** of gradients
  - Quantized into 10° steps (36 bins)
  - Amount added to the histogram is proportional to m
- **Step 4:** Search for **global maximum**
  - All values within 80% of the maximum value are valid orientations





Keypoint Descriptor

- So far, each keypoint has a location, scale, orientation.
- Now: compute a descriptor for the local image region of each keypoint that is highly distinctive and invariant as possible to variations such as changes in viewpoint and illumination.
- Example with 8x8 region divided into 4 cells

Gaussian window

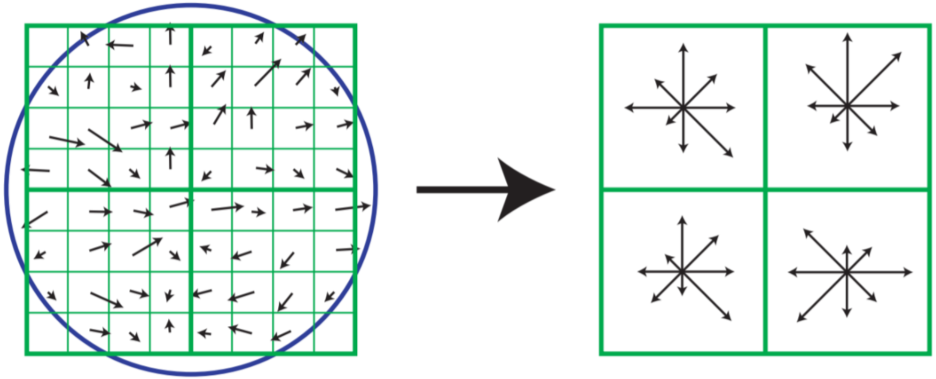


Image gradients

Keypoint descriptor

Harris Corner Detector vs. SIFT key point Detector

Harris

SIFT

Major advantages of SIFT

- **Locality:** features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance
- **Extensibility:** can easily be extended to a wide range of different feature types, with each adding robustness

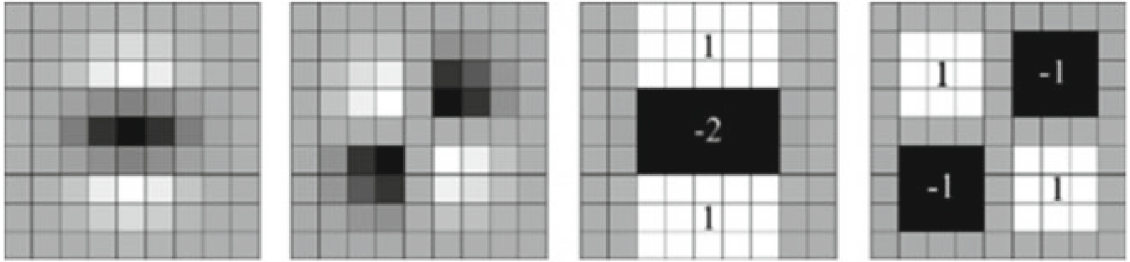
Use of SIFT

Karlsruher Institut für Technologie

- Object recognition
- Motion tracking
- Stereo calibration
- Image indexing and retrieval
- Robot navigation
- …

Speeded Up Robust Features (SURF)

- Designed as an efficient alternative to SIFT Features
- Detection stage relies on simple **2D box filters** instead of ideal Gaussian derivatives
  - Convolutions with box filters can be easily calculated with integral images (sum of pixel values in a given image)
  - Calculations in parallel for different scales



Left to right: Gaussian second order derivatives (with $\sigma = 1.2$) in y-, xy-direction and their approximations in the same directions, respectively.

Source: Ali Ismail Awad and Mahmoud Hassaballah. 2016. Image Feature Detectors and Descriptors: Foundations and Applications (1st ed.). Springer Publishing Company, Incorporated.

Maximally Stable Extremal Regions (MSER)

- Region detection algorithm developed in 2002 by Jiri Matas et al.

- Detected regions should be invariant under:
  - Illumination changes
  - Affine Transformations (Rotation, Translation, Scaling, Reflection, Shear)

- Maximally Stable Extremal Regions are defined solely by the **intensities** of an image
  - Find regions that remain consistent over a wide range of intensity thresholds
  - Take the most stable version of a consistent region

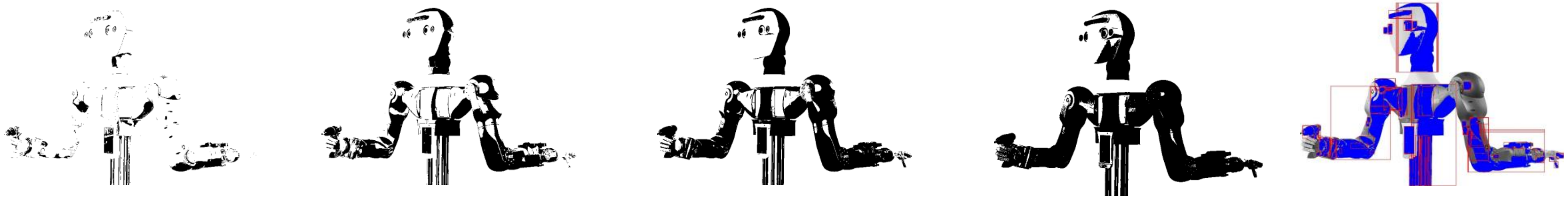
Matas, J., Chum, O., Urban, M., & Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. Image and vision computing, 22(10), 761-767.

Maximally Stable Extremal Regions (MSER) - II

- Create all possible thresholded versions of a gray-scale image
  - Each pixel above threshold is set to “white” and each pixel below is set to “black”

$$\mathrm{I_t}(\mathrm{x}, y) = \begin{cases} 0 & if\ I(x, y) < t \\ 255 & if\ I(x, y) > t \end{cases}$$

  - Find connected areas for each thresholding level
  - Create a list of all connected components and their size for a given threshold value
  - The region at threshold $\boldsymbol{t_{stable}}$ with the minimum rate of change of its area is taken as the Maximally Stable Extremal Region



t = 50 | 100 | 150 | 200 | Detected Regions

Maximally Stable Extremal Regions (MSER) - III

https://www.youtube.com/watch?v=6d6V5aWUynI

Maximally Stable Extremal Regions (MSER) - III

- Example: Detected MSER regions

Efficient Point Features

- **Combination** of corner detector and descriptor
  - Expensive **scale space analysis** is **avoided**
  - **Scale-independency** is reached by computing features at several predefined spatial scales explicitly
  - Allows **real-time** image processing (30 fps and more)
  - Examples:
    - FAST Detector + SIFT/Ferns-Descriptor: (Wagner et al., 2008)
    - Harris Corner detector + SIFT-descriptor: (Azad et al. , 2009)

Azad, P., Asfour, T. and Dillmann, R., *Combining Harris Interest Points and the SIFT Descriptor for Fast Scale-Invariant Object Recognition*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4275-4280, October, 2009

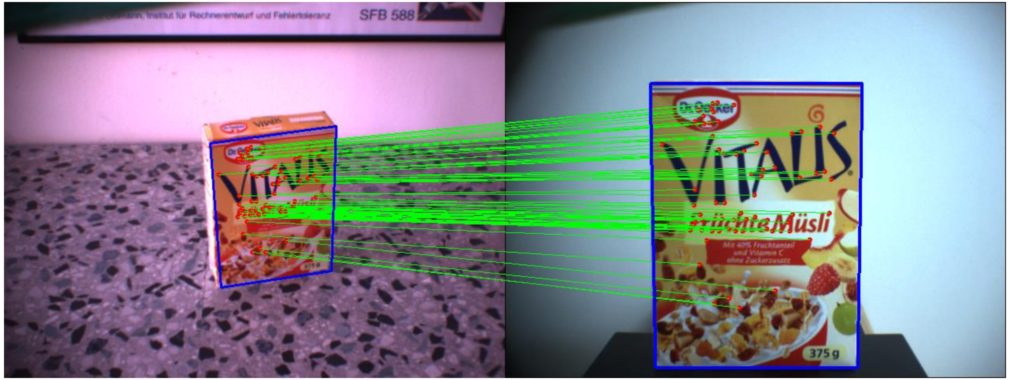
Object Detection with Feature Descriptors

- Feature descriptors can be used to efficiently detect objects and estimate their location

- Approach:
  - Extract feature descriptors of image
  - Identify correspondences between image features and object features:
    - Brute Force
    - Nearest Neighbors
    - RANSAC
  - Filter the matches and calculate the transformation

Object Detection with Feature Descriptors II

Unfiltered Correspondences (tolerant threshold for matching)



Filtered correspondences with RANSAC and determination of homography
+ Result of 2D-localization (blue box left side)

Outline

- Correlation Functions
- Corner Detectors
  - Moravec operator
  - Harris Corner Detector
  - Good Features to Track
  - Machine-learned Features
- Feature Descriptors
  - Simple Descriptors
  - SIFT
  - SURF
  - MSER
- **Pose Estimation**
  - Monocular
  - Stereo Images
  - Depth Images
  - Neural Networks

6D Pose Estimation

- Grasping requires knowledge of the object pose
  - Where to grasp?
  - Which grasp to choose?
- Grasps can be precomputed on object meshes
- But: Execution of the grasp requires 6D pose of the object in the scene

6D Pose:
$$\begin{pmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$
Position and orientation!

6D Pose Estimation

- Problem definition:
  - Given is a 3D model of the object
  - **Task:** Find the transformation (rotation and translation) which determines the 6D pose of the Object coordinate system (model) in World coordinate system
- In the following: World coordinate system = Coordinate system of (left) camera
- Different approaches depending on the camera system used:
  - Monocular: 2D-3D point correspondences
  - Stereo: 3D-points from stereo triangulation
  - Depth: point clouds

Monocular Pose Estimation - I

- Basics:
  - 2D-3D point correspondences
    - 3D points of the model (world coordinate system)
    - 2D points from current view (image coordinates)
  - Compute homography of 2D-3D point correspondences and use it for tracking of 2D-Points, e.g. with Kanade-Lucas-Tomasi Tracker (KLT-Tracker)

Monocular Pose Estimation - II

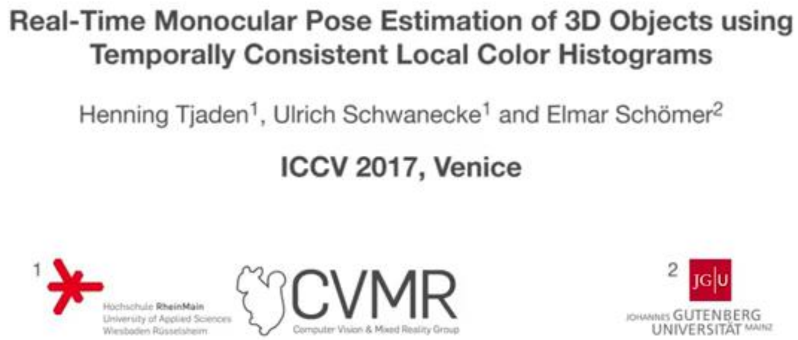
Karlsruher Institut für Technologie

- Algorithms for 6D pose estimation from 2D-3D point correspondences (called **Perspective n-Point**, PnP problem)

  - POSIT (Pose from Orthography and Scaling with Iterations)
    - Published in 1992 by Daniel F. DeMenthon and Larry S. Davis
    - In original version: 3D points are not allowed to be co-planar
    - Extended for co-planar 3D points: (Oberkampf et al., 1996)
  - Further Algorithms
    - (Lu et al., 2000)
    - (Schweighofer and Pinz, 2006)
    - (Moreno-Noguer et al., 2007)
    - (Schweighofer and Pinz, 2008)

Monocular Pose Tracking using Color Histograms

- Once the initial object pose is known, tracking of the object becomes feasible
  - Temporally Consistent Local Color histograms (TCLC-Histograms) are computed for the initial pose
  - Change of 6D-Pose is calculated for each new frame

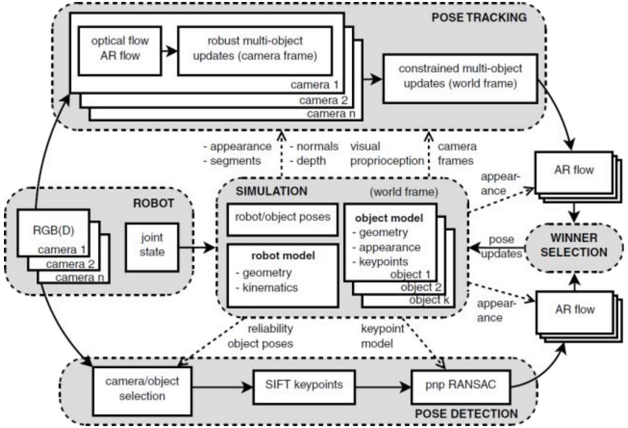


> H. Tjaden, U. Schwanecke and E. Schömer, "Real-Time Monocular Pose Estimation of 3D Objects Using Temporally Consistent Local Color Histograms," *2017 IEEE International Conference on Computer Vision (ICCV)*

Monocular Pose Estimation: SimTrack

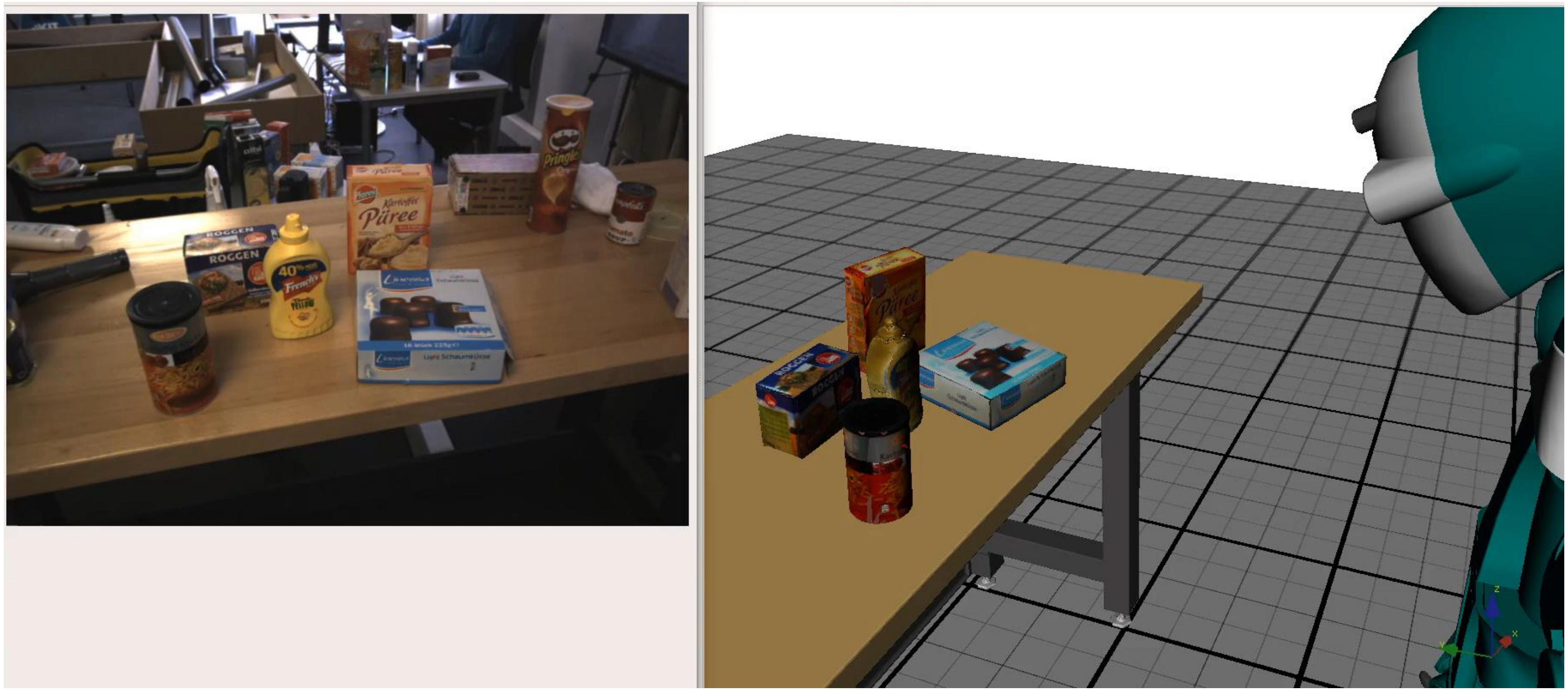
Detection and tracking of multiple objects

- **“Simulated” scene model**
  - Object models, current pose hypotheses
  - Used to render images of current model
- **Detection** based on SIFT-features & PnP
- **Tracking** based on *Augmented Reality flow*
  - Optical flow between AR image and camera image
  - AR image = objects rendered onto camera image
- **Selection** between detection and tracking pose candidates
  - Reliability measure based on proportion of valid AR flow



K. Pauwels and D. Kragic, "SimTrack: A simulation-based framework for scalable real-time object pose detection and tracking," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 1300-1307.

Monocular Pose Estimation: SimTrack @ H2T



Stereo-based Pose Estimation

- Multiple approaches exist, one possible solution:
  - Computation of 3D coordinates for feature points using **correlation** and **stereo triangulation** followed by:
    - **Fitting** of a geometric 3D primitive
    - **Registration** of a 3D object model

  - Advantages:
    - Robust, since stereo triangulation is used
    - Better accuracy (especially depth), depending on setup
  - Disadvantages:
    - Stereo calibration is needed
    - Inaccuracy with strong lens distortion

Example: Stereo-based Pose Estimation @ H2T

Pose Estimation on Depth Images

- Pose Estimation in 6D space is a challenging problem
- RGB-D Sensors naturally produce 3D data in the form of Point Clouds
  - No need to solve the **hard** 2D-3D problem

- BUT: Point Clouds are **unordered** (unlike 2D images)
  - Convolutions that were used to calculate features in 2D images cannot be easily applied
  - Neighborhoods need to be computed and are not implicitly defined
  - Learning methods (e.g. Neural Networks) have a hard time with unordered sets

Pose Estimation on Depth Images – ICP

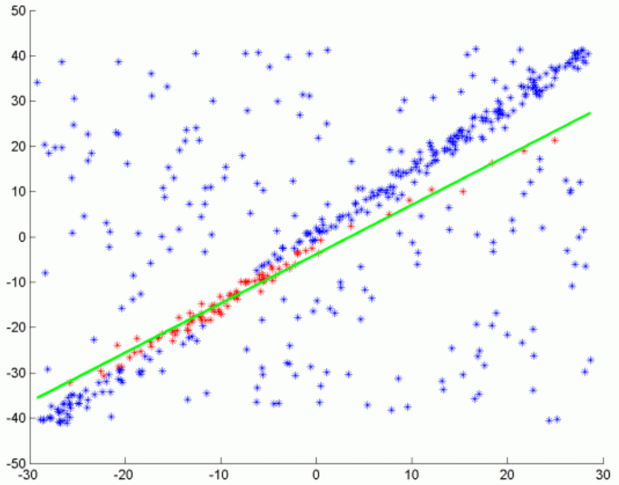
- **Iterative Closest Point (ICP)**
  - Iterative transformation of a point cloud to best match the reference (the object)
  - Can be used to align 3D models of objects
  - Works with incomplete data (e.g., from occlusions)

  - **Algorithm**: (see Robotics I)
    - For each point in the point cloud find the closest point in the reference set
    - Estimate the transformation that minimizes the distances of all correspondences
    - Transform the point cloud and iterate until a certain accuracy or the maximum number of iterations is reached

  - Local minimum: in case of complex object geometries
  - The higher the required accuracy, the higher is the runtime of the algorithm
  - Prone to errors for data with outliers

Pose Estimation on Depth Images – RANSAC

- **RANdom SAmple Consensus (RANSAC)**
  - Iterative method to estimate parameters of a model from data
  - Works with incomplete data and is robust against outliers
  - **Algorithm:**
    - Randomly sample subset from input data
    - Fit the model to best resemble the subset
    - Find the points in the input data that are closer than threshold to the model; those are called the **consensus set**
    - Repeat until the consensus set is large enough or a maximum number of iterations is reached



Pose Estimation using Neural Networks

- Challenges: training, camera dependent, inherent ambiguity with symmetries
  → Pose estimation is (still) dominated by classical methods

Methods based on point pair features, Template matching methods, Learning-based methods, Methods based on 3D local features

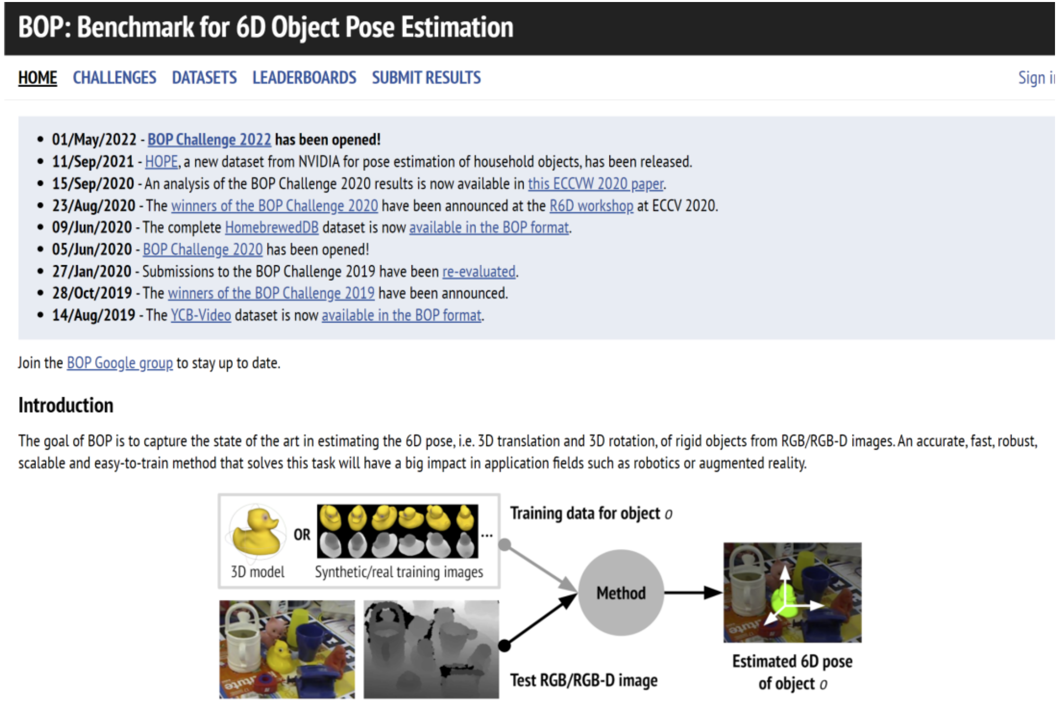
#	Method	LM	LM-O	IC-MI	IC-BIN	T-LESS	RU-APC	TUD-L	Average	Time (s)
1.	Vidal-18	87.83	59.31	95.33	96.50	66.51	36.52	80.17	74.60	4.7
2.	Drost-10-edge	79.13	54.95	94.00	92.00	67.50	27.17	87.33	71.73	21.5
3.	Drost-10	82.00	55.36	94.33	87.00	56.81	22.25	78.67	68.06	2.3
4.	Hodan-15	87.10	51.42	95.33	90.50	63.18	37.61	45.50	67.23	13.5
5.	Brachmann-16	75.33	52.04	73.33	56.50	17.84	24.35	88.67	55.44	4.4
6.	Hodan-15-nopso	69.83	34.39	84.67	76.00	62.70	32.39	27.83	55.40	12.3
7.	Buch-17-ppfh	56.60	36.96	95.00	75.00	25.10	20.80	68.67	54.02	14.2
8.	Kehl-16	58.20	33.91	65.00	44.00	24.60	25.58	7.50	36.97	1.8
9.	Buch-17-si	33.33	20.35	67.33	59.00	13.34	23.12	41.17	36.81	15.9
10.	Brachmann-14	67.60	41.52	78.67	24.00	0.25	30.22	0.00	34.61	1.4
11.	Buch-17-ecsad	13.27	9.62	40.67	59.00	7.16	6.59	24.00	22.90	5.9
12.	Buch-17-shot	5.97	1.45	43.00	38.50	3.83	0.07	16.67	15.64	6.7
13.	Tejani-14	12.10	4.50	36.33	10.00	0.13	1.52	0.00	9.23	1.4
14.	Buch-16-ppfh	8.13	2.28	20.00	2.50	7.81	8.99	0.67	7.20	47.1
15.	Buch-16-ecsad	3.70	0.97	3.67	4.00	1.24	2.90	0.17	2.38	39.1

T. Hodaň, F. Michel, E. Brachmann, W. Kehl, A. G. Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, C. Rother, BOP: Benchmark for 6D Object Pose Estimation, European Conference on Computer Vision (ECCV) 2018, Munich.

BOP: Benchmark for 6D Object Pose Estimation

- https://bop.felk.cvut.cz/home/

- “The goal of BOP is to capture the state of the art in estimating the 6D pose, i.e. 3D translation and 3D rotation, of rigid objects from RGB/RGB-D images. An accurate, fast, robust, scalable and easy-to-train method that solves this task will have a big impact in application fields such as robotics or augmented reality.”



Learning on Unordered Point Sets

- The output of neural network (Dense, Convolution, Recurrent …) is **not invariant** to the order of the input data

1	2	3	4	5	6	7	$\rightarrow$	2	3	1	9
1	5	7	4	2	6	3	$\rightarrow$	4	1	7	5

- Problem: How to order a point set in $\mathbb{R}^n$?

Vinyals, O., Bengio, S., Kudlur, M., & Brain, G. Order Matters: Sequence to sequence for sets; https://arxiv.org/abs/1511.06391

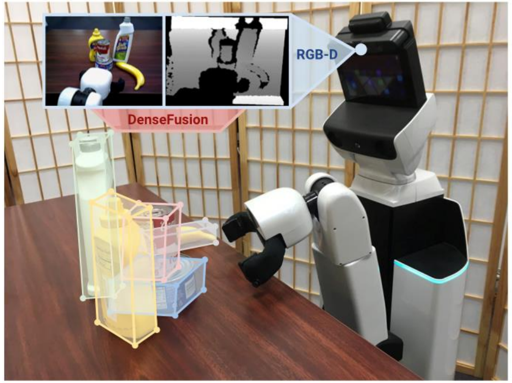
Pose Estimation with Neural Networks

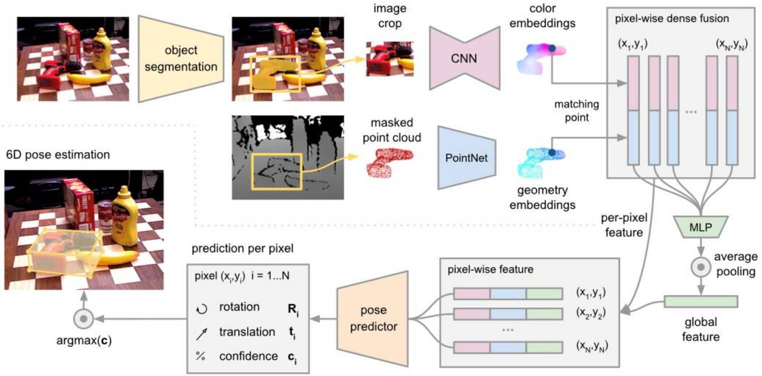
- Neural Networks can solve many intractable problems of classical methods by approximating them
  - Object detection and classification can provide prior information
  - Pixel-level segmentation and bounding box prediction sets constraints on position and orientation of the object
  - **2D-3D correspondences can be learned**

- Recent advances in **geometric deep learning** allow for deep learning on non-image input data
  - Volumetric models (VoxNet)
  - Point-based methods (PointNet)
  - Graph-based models (GraphCNN)

Example: DenseFusion

- Mask RCNN and PointNet for 6D pose estimation
- Input: Segmented RGB-D image (mask of pixels that belong to the object)
  - Masked depth image is fed to PointNet
  - Masked RGB image is fed to a Mask RCNN
  - Pixel-wise dense features (depth & RGB) are fused in image coordinates to calculate global features
  - 6D Pose is calculated using global and local features





Literatur

1. Shi, Jianbo. "Good features to track." *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994.
2. “Distinctive Image Features from Scale-Invariant Keypoints” - Lowe, David G., IJCV 2004
3. Ali Ismail Awad and Mahmoud Hassaballah. 2016. Image Feature Detectors and Descriptors: Foundations and Applications (1st ed.). Springer Publishing Company, Incorporated
4. Matas, J., Chum, O., Urban, M., & Pajdla, T. (2004). Robust wide-baseline stereo from maximally stable extremal regions. Image and vision computing, 22(10), 761-767.
5. Schweighofer, G., & Pinz, A. (2006). Robust pose estimation from a planar target. IEEE transactions on pattern analysis and machine intelligence, 28(12), 2024-2030
6. Moreno-Noguer, F., Lepetit, V., & Fua, P. (2007, October). Accurate Non-Iterative O (n) Solution to the PnP Problem. In 2007 IEEE 11th International Conference on Computer Vision (pp. 1-8). IEEE.
7. Schweighofer, G., & Pinz, A. (2008, September). Globally Optimal O(n) Solution to the PnP Problem for General Camera Models. In BMVC (pp. 1-10).
8. K. Pauwels and D. Kragic, "SimTrack: A simulation-based framework for scalable real-time object pose detection and tracking," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 1300-1307.
9. H. Tjaden, U. Schwanecke and E. Schömer, "Real-Time Monocular Pose Estimation of 3D Objects Using Temporally Consistent Local Color Histograms," *2017 IEEE International Conference on Computer Vision (ICCV)*
10. Xiang, Y., Schmidt, T., Narayanan, V., & Fox, D. (2017). Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes
11. Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L., & Savarese, S. (2019). Densefusion: 6d object pose estimation by iterative dense fusion. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3343-3352).