DD2419 Robot Vision

Object Detection and 3D Pose Estimation

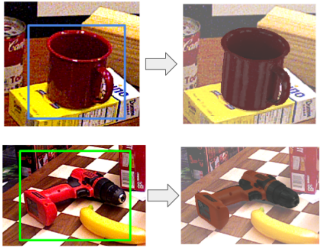
Leonard Bruns
Robotics, Perception and Learning, KTH Royal Institute of Technology

Outline

- Overview: Deep Learning for Vision
- Object Detection
- 3D Pose Estimation

Overview: Deep Learning for Vision

- Deep learning-based approaches dominate computer vision field since about 10 years
- Many vision tasks have nowadays been addressed with deep learning



https://thispersondoesnotexist.com/

Overview: Deep Learning for Vision

General Deep Learning

- Deep learning is function approximation on large datasets
- Given inputs $\boldsymbol{X}$ and corresponding outputs (targets) $\boldsymbol{Y}$, find $f(\boldsymbol{X}) \approx \boldsymbol{Y}$
- In practice: family of function $f_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} l(f_{\boldsymbol{\theta}}(\boldsymbol{X}), \boldsymbol{Y}) \tag{1}$$

- Goal is that unseen inputs $\boldsymbol{X}'$ still produce expected output $\boldsymbol{Y}'$
  - I.e., f should *generalize* to unseen inputs
- Generalization comes from
  - Large datasets,
  - Architecture and specific function types,
  - Hierarchical representations (deep vs shallow learning),
  - ...

Overview: Deep Learning for Vision

Tasks

- Various vision tasks can be approached with deep learning
- Deep learning nowadays state-of-the-art in all of these tasks
- Well-established tasks (with large benchmarks and datasets available)
    - Classification
    - Detection
    - Instance segmentation
    - Semantic segmentation
- Current research
    - 3D vision
    - Pose estimation
    - Image generation

Overview: Deep Learning for Vision

Classification

$$\xrightarrow{\quad f(x) \quad} \quad \tilde{y} = \begin{bmatrix} 0.03 \\ 0.9 \\ 0.01 \\ \vdots \\ 0.03 \end{bmatrix} \quad \longleftrightarrow \quad \boldsymbol{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$0.9 \rightarrow$ Dog

- Normally probability vector as output (one-hot encoding as target)
- Not well-defined in cluttered scenes

Overview: Deep Learning for Vision

Detection

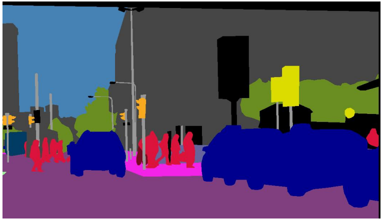
$$— f(x) —$$

- Various output representations exist
- More details later

Overview: Deep Learning for Vision

Semantic Segmentation

— $f(x)$ —

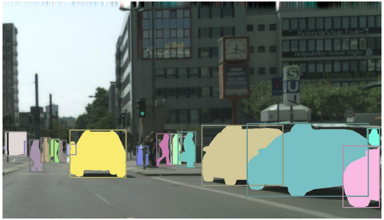


- One-hot encoding per-pixel
- Not possible to differentiate different instances

Overview: Deep Learning for Vision

Instance Segmentation

— $f(x)$ —



- Object detection + segmentation mask prediction
- Stuff (uncountable things) is not segmented / classified

Overview: Deep Learning for Vision

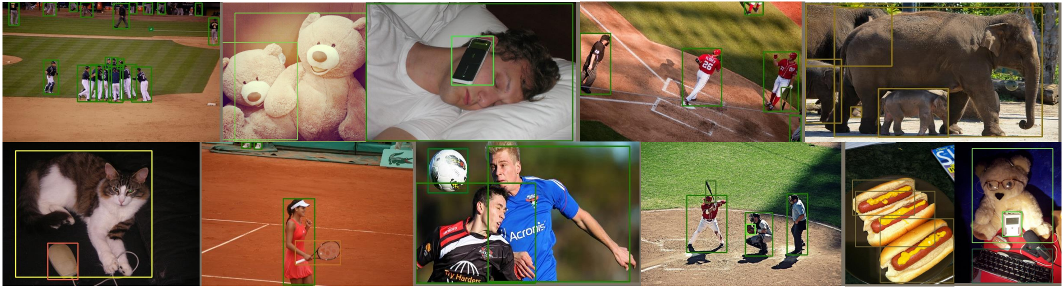
Panoptic Segmentation

$$— f(x) —$$

- Combining instance and semantic segmentation
- Instance segmentation for things (countable)
- Semantic segmentation for stuff (uncountable)

Object Detection

- Goal: predict bounding box around objects and classify the object
- We provide you with a baseline which should be easy to extend and understand



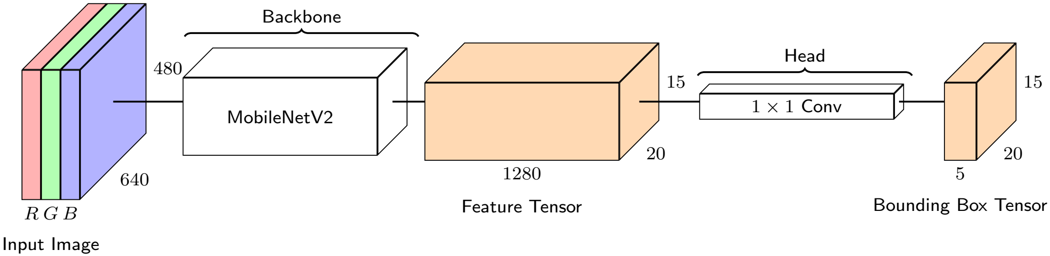
Object Detection

Bounding Box Parametrization

- Neural network-based detectors follow same prinicples
  1. Extract useful image features using larger *backbone*
  2. Predict bounding box position, size and class based on these features
- Various output parametrizations possible
  - Bounding boxes
    - Anchor-based: Default bounding boxes (nominal width + height)
    - Anchor-free: No prior bounding box sizes
  - Classification: one-hot encoding per bounding box (i.e., N-dimensional vector for N classes)

Object Detection

Baseline

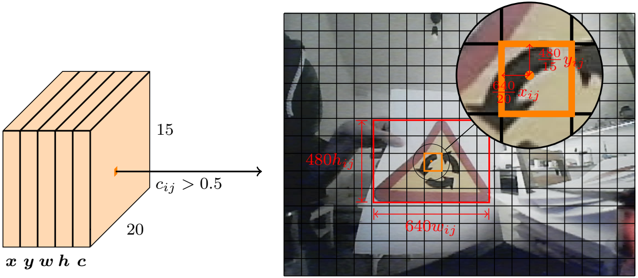


- Pretrained backbone to speed up training (and improve performance)
- Be careful not to train too long (overfitting such small datasets easily possible)

Object Detection

Baseline, Bounding Box Parametrization

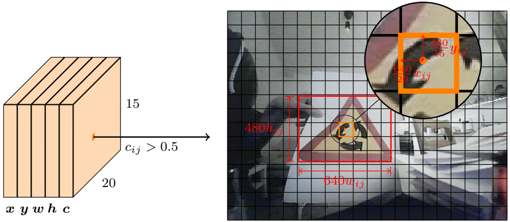
- Bounding box represented by 4 parameters: $x_{ij}, y_{ij}, w_{ij}, h_{ij}$



Object Detection

Baseline, Classification

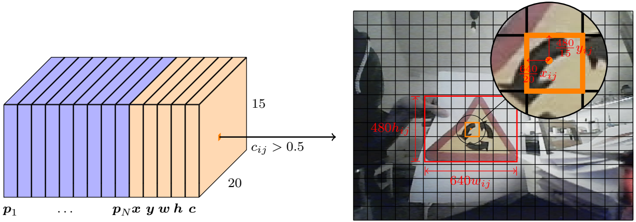
- Baseline does not output categories yet
- Idea 1: add category to output volume (see below)
- Idea 2: train separate classification network and only evaluate top-scoring bounding boxes (two-stage)



Object Detection

Baseline, Classification

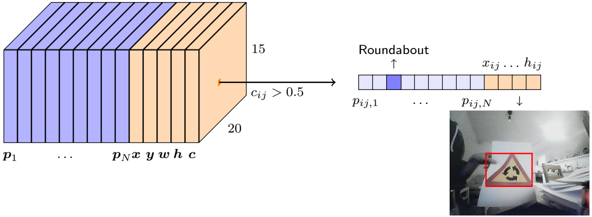
- Baseline does not output categories yet
- Idea 1: add category to output volume (see below)
- Idea 2: train separate classification network and only evaluate top-scoring bounding boxes (two-stage)



Object Detection

Baseline, Classification

- Baseline does not output categories yet
- Idea 1: add category to output volume (see below)
- Idea 2: train separate classification network and only evaluate top-scoring bounding boxes (two-stage)



Object Detection

Baseline, Outlook

- Synthetic or more real data
- Data augmentation
- Other backbones (e.g., ResNet)
- Other bounding box parametrizations, for example, default bounding boxes with anchors
- Improve speed by reducing input image size (e.g., 320x240)
- Network before or after fish-eye undistortion?
- Non-maximum suppression / merging of bounding boxes

Object Detection

Outlook, Non-Maximum Suppression

- Often multiple bounding boxes are predicted for the same object
- For all bounding boxes with $IoU > t$, only keep bounding box with highest confidence [1]
- Alternative: merge bounding boxes

[1] https://pytorch.org/docs/stable/torchvision/ops.html#torchvision.ops.nms

Object Detection

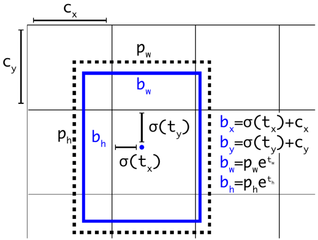
Outlook, Data Augmentation

- Performance correlates a lot with amount of training data
- Idea: generate new training data by using the already existing data
- Many possibilities
  - Zoom
  - Rotate (how does bounding box change?)
  - Flipping (be careful with mirrored traffic signs...)
  - Color
  - Various noise / occlusions

Object Detection

Example, YOLOv3

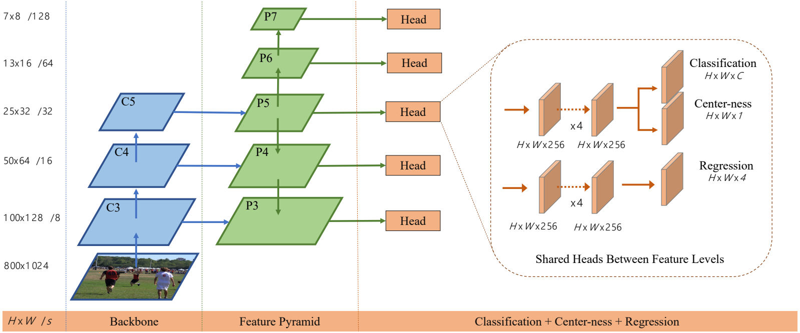
- DarkNet backend
- Multiple anchor boxes (found with k-means clustering) per feature location
- Bounding box parametrized based on default size

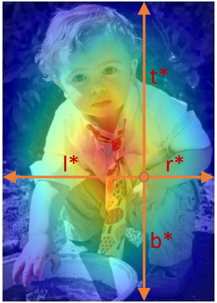


Object Detection

Example, FCOS

- Feature-pyramid network (fuse high-level with low-level features)
- Prediction at multiple levels, no anchor boxes
- Centerness prediction to improve results





Object Detection

Training, General

- Training neural networks normally follow the same principle
- Network $f_{\boldsymbol{\theta}}(x)$ with network params $\boldsymbol{\theta}$
- Ideally would do the following optimization

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} l(f_{\boldsymbol{\theta}}(\boldsymbol{x'}), \boldsymbol{y'}) \tag{2}$$

  where $\boldsymbol{x'}, \boldsymbol{y'} \sim \mathcal{R}$ are unseen samples from the real world
- Reality: use a labelled dataset $\mathcal{D} \sim \mathcal{R}$ of images $\boldsymbol{x}$ with corresponding labels $\boldsymbol{y}$

Object Detection

Training, General

Training Procedure

1. Initialize network parameters $\boldsymbol{\theta}$
2. Sample (or generate) batch of inputs and targets $\boldsymbol{X}, \boldsymbol{Y} \sim \mathcal{D}$
3. Forward pass $f_{\boldsymbol{\theta}}(\boldsymbol{X}) = \tilde{\boldsymbol{Y}}$
4. Compute loss function $l = l(\tilde{\boldsymbol{Y}}, \boldsymbol{Y})$
5. Update network parameters based on $\frac{\partial l}{\partial \boldsymbol{\theta}}$ to reduce l
6. Repeat from step 2

Object Detection

Training, Target Assignment

- Need to convert labels (i.e., bounding boxes) to suitable *target* (i.e., ideal output)
- Bounding boxes
  - Cell of ground truth bounding box center gets confidence 1 (see baseline)
  - Anchor box overlaps sufficiently with ground truth bounding box $\rightarrow 1$ (e.g., YOLOv3)
  - Regression target for box size (depends on used convention)
- Classification
  - One-hot encoding, set nth entry of target vector to 1, otherwise 0

Object Detection

Training, Losses

- Goal: $l(\tilde{\boldsymbol{Y}}, \boldsymbol{Y})$ should measure how close output is to target
- Must be differentiable
- Simple choice: (weighted) mean squared error [1]

$$l(\tilde{\boldsymbol{Y}}, \boldsymbol{Y}) = \frac{1}{N} \sum_{i=1}^{N} (\tilde{Y}_i - Y_i)^2 \tag{3}$$

- Classification often uses cross entropy loss [2]

---

[1] `https://pytorch.org/docs/stable/nn.functional.html#mse-loss`
[2] `https://pytorch.org/docs/stable/nn.functional.html#cross-entropy`

Object Detection

Overfitting and Generalization

- Training for too long will overfit the training data
- Loss *will* keep going down
  - Does not mean that performance is improving
- Solution: evaluate on separate validation set
  - Must be really separate (i.e., different lighting, environment, etc.)
  - Naive split of current dataset probably not sufficient

Training

Validation


Validation

- Alternative: qualitative assessment of unlabelled validation images

Object Detection

Evaluation Metrics

- Goal: assess performance independent of loss function (on validation data)
- I.e., loss function changes, evaluation metric stays the same
  - Allows to compare loss functions
  - Allows to assess generalization performance
- IoU: Intersection over union to compare two bounding boxes

$$IoU = \frac{Intersection}{Union} \tag{4}$$

Object Detection

Evaluation Metrics

- Goal: assess performance independent of loss function (on validation data)
- Precision (based on IoU threshold)

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

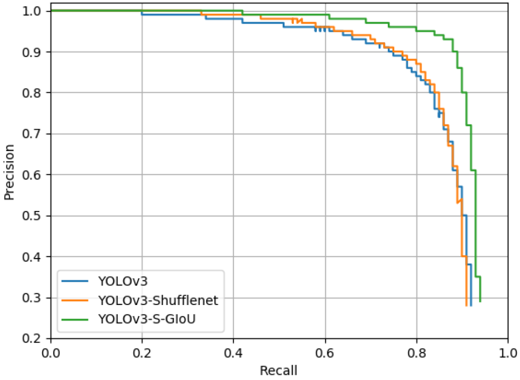
- Recall (based on IoU threshold)

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

Object Detection

Evaluation Metrics

- Goal: assess performance independent of loss function (on validation data)
- Precision-recall curve (varying confidence threshold)

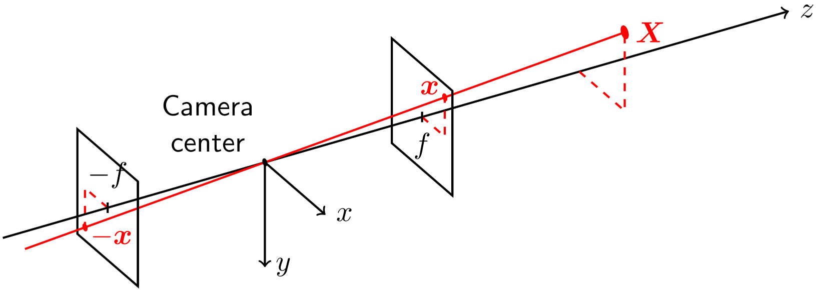


3D Pose Estimation

- Goal: estimate 6D pose of known object from single image
- General approach
  - Model camera geometry (pinhole camera + nonlinear distortion)
  - Use known object information to derive constraints
    - Features
    - Size
    - Shape
    - Pose constraints
    - ...
  - Solve equations to estimate pose

3D Pose Estimation

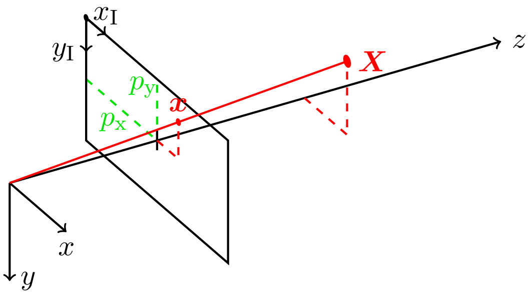
Pinhole Camera



$$\boldsymbol{x} = \underbrace{\begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\mathbf{P}} \boldsymbol{X} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} \sim \begin{pmatrix} fX/Z \\ fY/Z \\ 1 \end{pmatrix} \tag{7}$$

3D Pose Estimation

Pinhole Camera



$$\boldsymbol{x} = \underbrace{\begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\mathbf{P}} \boldsymbol{X} = \begin{pmatrix} fX + p_{\mathrm{x}}Z \\ fY + p_{\mathrm{y}}Z \\ Z \end{pmatrix} \sim \begin{pmatrix} fX/Z + p_{\mathrm{x}} \\ fY/Z + p_{\mathrm{y}} \\ 1 \end{pmatrix} \tag{8}$$

3D Pose Estimation

Camera Calibration

- Most general matrix

$$\mathbf{P} = \begin{pmatrix} f_{\mathrm{x}} & \tau & p_{\mathrm{x}} & 0 \\ 0 & f_{\mathrm{y}} & p_{\mathrm{y}} & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{9}$$

- Normally $\tau = 0, f_{\mathrm{x}} \approx f_{\mathrm{y}}, p_{\mathrm{x}} \approx \frac{width}{2}, p_{\mathrm{y}} \approx \frac{height}{2}$
- Idea: collect images of easy known, easy to detect 3D points and find $\mathbf{P}$ that minimizes error

https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html

3D Pose Estimation

Camera Calibration

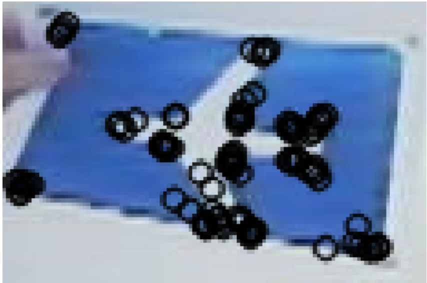
- Nonlinear distortions need to be compensated before using pinhole model
- For example, fish-eye distortion

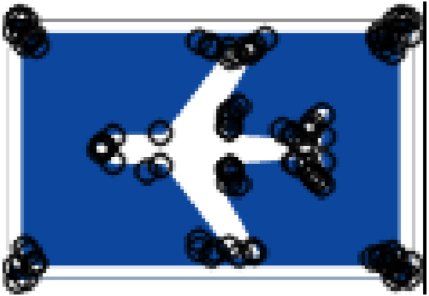
`https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html`

3D Pose Estimation

Feature Extraction

- Several methods to extract point features
- For example, SIFT, SURF, FAST, BRIEF, ORB, ... [1]
- Output of feature extractor: position + descriptor





[1] https://docs.opencv.org/master/db/d27/tutorial_py_table_of_contents_feature2d.html

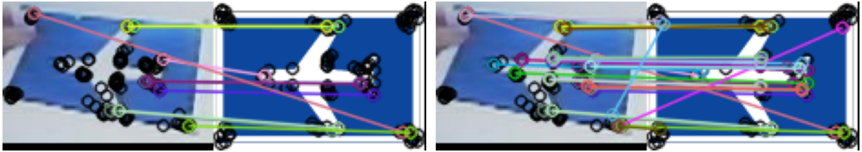
3D Pose Estimation

Feature Matching

- Match features based on descriptor
  - Brute-force matching (try all combinations)
  - Nearest-neighbor matching (faster, but approximate)
- Ratio-test: keep best 2 matches, only keep best match if one is clearly better

$$\operatorname{dist}(d, d'_{\text{2nd best}}) \overset{!}{<} r \cdot \operatorname{dist}(d, d'_{\text{best}}) \tag{10}$$

- Cross-check: KP1 has to get KP2 and KP2 has to get KP1



https://docs.opencv.org/master/dc/dc3/tutorial_py_matcher.html

3D Pose Estimation

Perspective-N-Point

- Given 3D $\leftrightarrow$ 2D point correspondences, find ${}^{\mathrm{c}}\mathbf{T}_{\mathrm{w}}$, s.t.,

$$\boldsymbol{x}_i = \mathbf{P}\,{}^{\mathrm{c}}\mathbf{T}_{\mathrm{w}}\,\boldsymbol{X}_i \tag{11}$$

- 6 DOFs $\Rightarrow$ 3 correspondences required (still 8 different solutions, 4 behind camera)
- More correspondences $\Rightarrow$ unique solution
  - Iterative optimization with initial guess: slow, but most accurate
  - Non-iterative: fast, but suboptimal, good to get initial guess

https://docs.opencv.org/master/d9/d0c/group__calib3d.html#ga549c2075fac14829ff4a58bc931c033d

3D Pose Estimation

RANSAC

- Matched features will contain outliers
- Use RAndom SAmple Consensus (RANSAC) to robustify estimation
- General idea
    - Sample minimum set of matches to estimate pose
    - Count number of inliers
    - Repeat N and use set with maximum number of inliers

3D Pose Estimation

Pipeline, Overview

Feature-based Pose Estimation

1. Compute features inside detected bounding box
2. Match features with canonical traffic sign image
3. Estimate translation and rotation using perspective-n-point with RANSAC
4. Additional sanity checks to remove errorneous detections

3D Pose Estimation

Alternatives

- Learning to predict the pose together with bounding box
  - Labeling quite labor intensive
  - Synthetic data might work, but: domain gap
- Estimate pose from multiple views
  - Position of bounding box gives bearing
  - Size of bounding box gives distance
  - Apply filter to optimize pose from multiple frames
- Particle filter + photometric comparison (easier to incorporate prior information)

References I

[1] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2004.

[2] Alexander Kirillov et al. "Panoptic Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9404–9413.

[3] Yann Labbé et al. "CosyPose: Consistent Multi-view Multi-object 6D Pose Estimation". In: *European Conference on Computer Vision*. Springer. 2020, pp. 574–591.

[4] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

References II

[5] Haojie Ma et al. "Detection of collapsed buildings in post-earthquake remote sensing images based on the improved YOLOv3". In: *Remote Sensing* 12.1 (2020), p. 44.

[6] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[7] Zhi Tian et al. "FCOS: Fully Convolutional One-stage Object Detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9627–9636.