Karlsruher Institut für Technologie

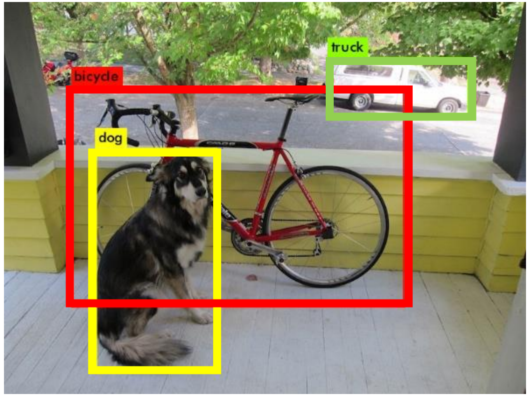
Hochperformante Humanoide Technologien

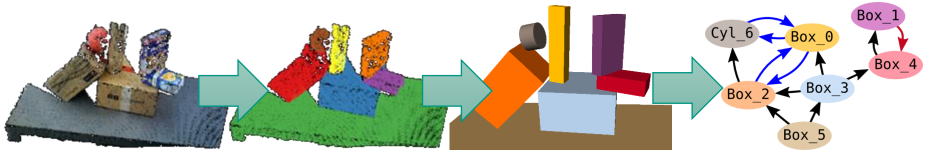
Robotics III: Sensors and Perception in Robotics

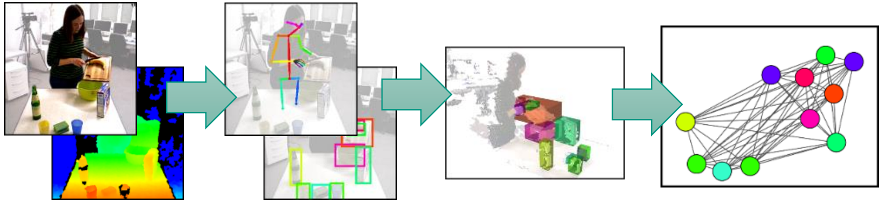
Chapter 06: Scene Understanding

Tamim Asfour

http://www.humanoids.kit.edu







Overview

- Introduction
- Scene Representations
- Machine Learning for Object Relations
- Leveraging Object Relations

Scene Understanding: Definition

Scene Understanding describes the cognitive process of transforming **raw visual input** into a **semantic scene representation**.

Raw Visual Data → Scene Understanding → Scene Representation

Examples:

- Color images
- Depth images
- Point clouds

Examples:

- Pixel-wise Labeling
- Object bounding boxes
- Scene graph

Scene Understanding: Geometric Information

Image (RGB / RGB-D)
- Regular structure (2D grid)
- RGB: color, RGB-D: color + depth

Point Cloud
- Natural 3D representation
- Unordered sets

Voxel Grid
- Regular structure (3D grid)
- Sparse data representation

Object Mesh
- High information density
- Artificial data format

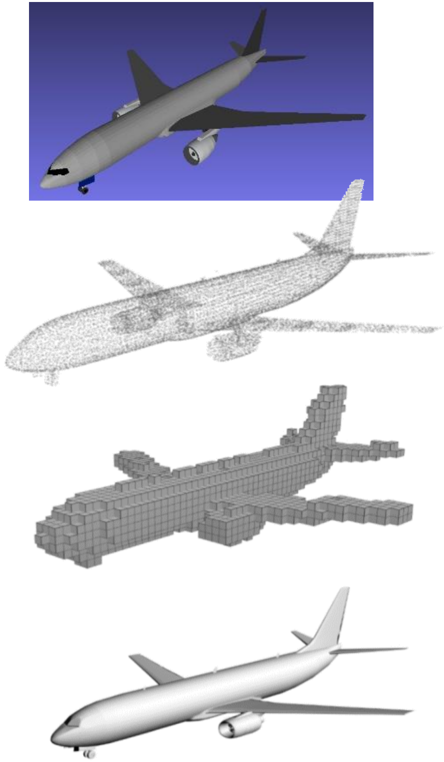
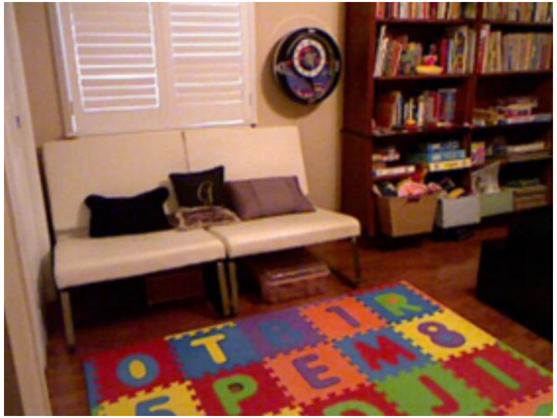


Image from Li, Y., Pirk, S., Su, H., Qi, C. R., & Guibas, L. J. (n.d.). FPNN: Field Probing Neural Networks for 3D Data; https://arxiv.org/abs/1605.06240

Scene Understanding: Pixel-wise Labeling

Pixel-wise labeling annotates **each pixel** of the input image with a **semantic label**, e.g. floor, wall, and sofa.



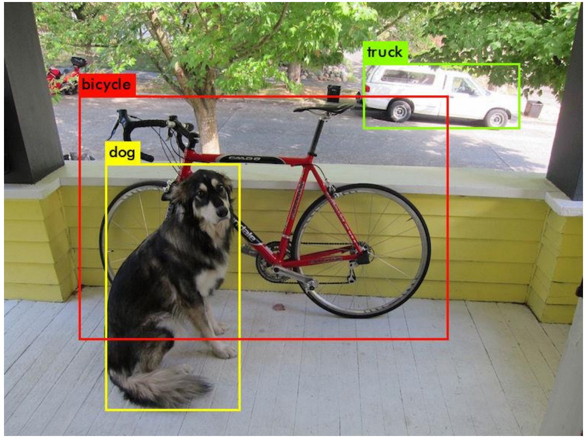
Scene Understanding

S. Gupta, P. Arbelaez, and J. Malik. “Perceptual organization and recognition of indoor scenes from RGB-D images,” CVPR (2013).

Scene Understanding: Object Bounding Boxes

- Classification and localization of multiple objects
- Localization: Bounding box around the detected object

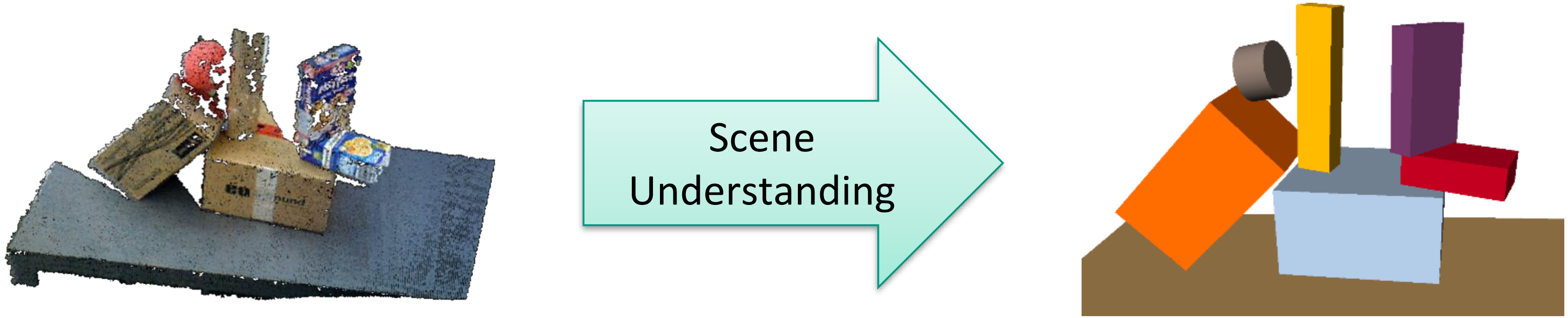
Scene Understanding



Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger," CVPR (2017).

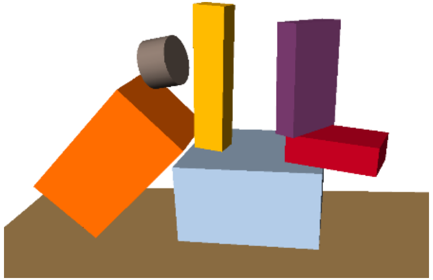
Scene Understanding: Primitive Extraction

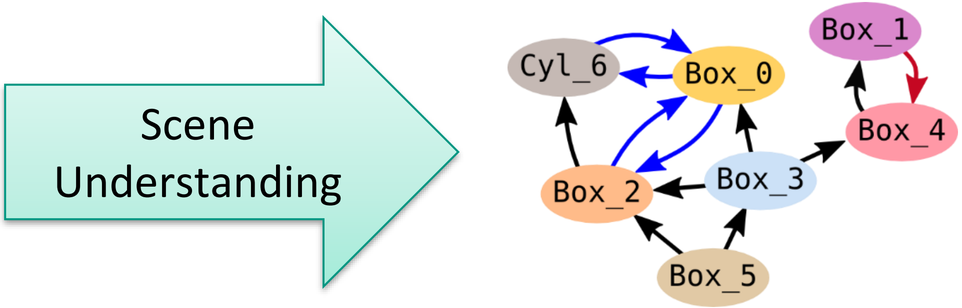
- Segment input into separate objects or object parts
- For each segment: Fit a **geometric primitive**
- Geometric primitives: planes, boxes, cylinders, spheres, …



Scene Understanding: Support Graph

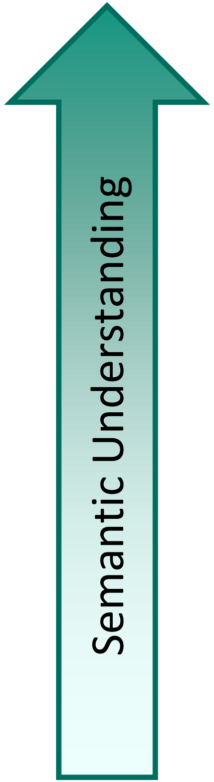
- **Physical relationship** between objects:
  Which objects are supported by which other objects?
- Representation: **Graph**
  - Nodes: objects
  - Edges: support relations





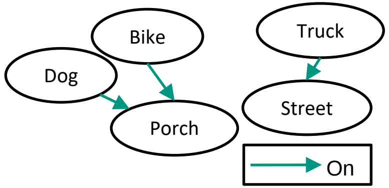
R. Kartmann, F. Paus, M. Grotz and T. Asfour, "Extraction of Physically Plausible Support Relations to Predict and Validate Manipulation Action Effects," RA-L (2018)

Levels of Semantic Understanding



High

Object Relations

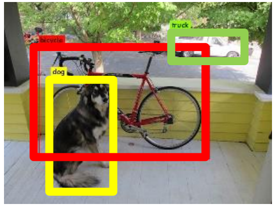


Spatial Relations
Temporal Relations
Support Relations

Object Instances

Object Instance Detection
Object Localization
6D Pose Estimation

Annotated Images



Classification
Bounding Boxes
Pixel-wise Labeling

Images

Color Images
Depth Images
Point Clouds

Low

Overview

- Introduction
- **Scene Representations**
  - Annotated Images
  - Object Instances
  - Object Relations
- Machine Learning for Object Relations
- Leveraging Object Relations

Overview

- Introduction
- **Scene Representations**
  - **Annotated Images**
  - Object Instances
  - Object Relations
- Machine Learning for Object Relations
- Leveraging Object Relations

Image Classification and Object Localization

- Image classification **assigns one label** from a predefined set of class labels to an input image

- Object localization additionally **finds the parts of an image** belonging to the determined instance of an object class

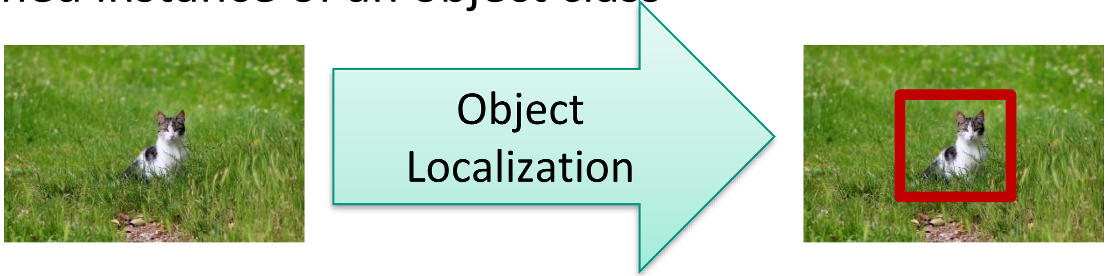


Image Classification: Single vs. Multiple Objects

- Single-object image classification assigns **one class label** per input image

Cat

Cat

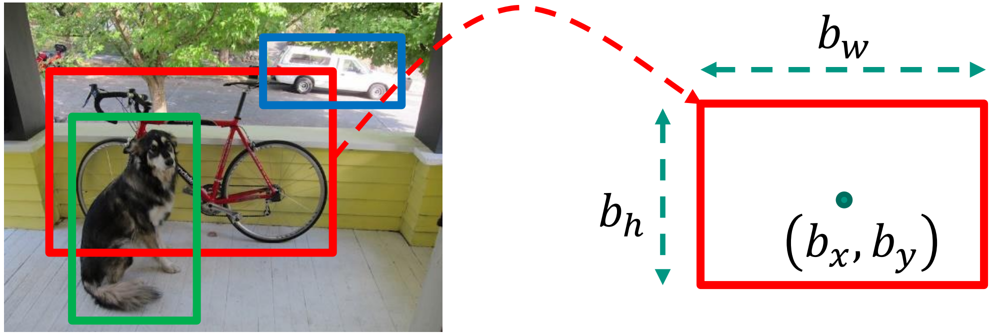
- Multi-object image classification assigns **more than one class label** per input image

Cat, Dog, Duck

Cat, Dog, Duck

Object Detection using Bounding Boxes

- Object detection is a multi-object image classification and localization task
  - Determines **bounding boxes** of every detected object in the image
  - Assigns the **label of the object class** to each bounding box
- A detected object can be described by its center (b_x, b_y), its width b_w and its height b_h

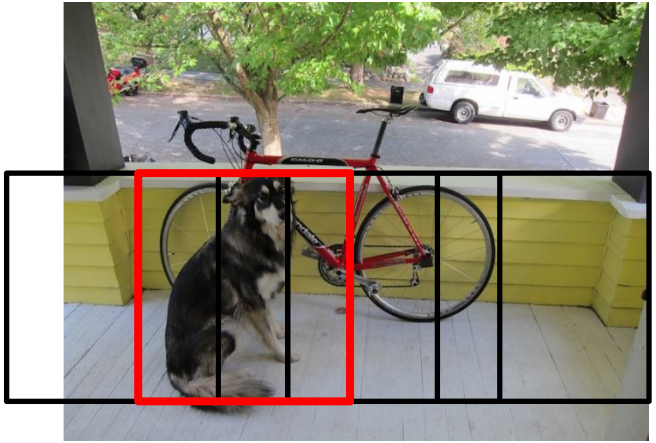


Dog, Bike, Truck

Object Detection for Multiple Objects (I)

- Reuse image classification for detection of multiple objects
  - Sliding window
  - Region Proposal Networks (RPN) to generate boxes

- **Sliding window** over the input image
  - Run image classification on each window
  - Example: Region-based CNN (R-CNN), Fast R-CNN



R. B. Girshick, "Fast R-CNN", ICCV, pp. 1440-1448 (2015)

Object Detection for Multiple Objects (II)

- **Region Proposal Networks (RPN)**
  - Use a network to propose possible object bounding boxes (image regions)
  - Only run the image classifier on the proposed bounding boxes
  - Example: Felzenszwalb et al., 2010

- Disadvantages of reusing image classification for object detection
  - Performance: Classification needs to be run **for each window**
  - Complexity: Classification and bounding box proposal are **different systems** which need to be trained/configured separately

P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part based models”. TPAMI (2010)

Object Detection using YOLO

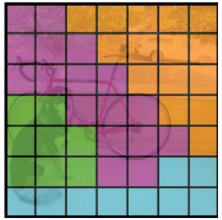
YOLO (You Only Look Once) solves multi-object detection as a **single regression problem** (compared to sliding window or RPN approaches)

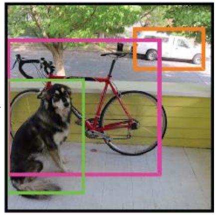
- Input: Color image
- Output: Bounding boxes, class label and class probability
- Uses a multi-layer convolutional neural network
- The network structure is simpler than most other state-of-the-art methods which allows **real-time execution** on modern GPUs
- Open Source: https://pjreddie.com/darknet/yolo/

Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In CVPR (2017).

YOLO: Overview

- Split the image into a $S \times S$ grid
- Predict B bounding boxes per grid cell
- Classify each grid cell
- Use non-maximum suppression to filter bounding boxes (detect every object only once)

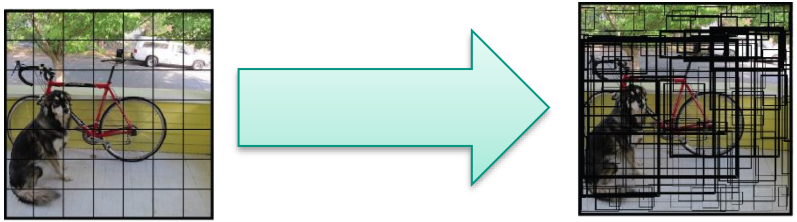




Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., “You only look once: Unified, real-time object detection”. ICVPR (2016)

YOLO: Bounding Box Prediction

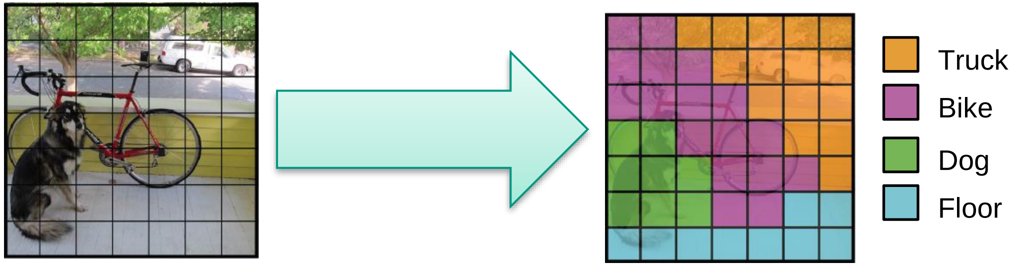
- Predict B bounding boxes per grid cell
- Each cell is responsible for detecting objects whose center falls into the corresponding cell



- Each bounding box can be described by 5 parameters
  - Geometric parameters (b_x, b_y, b_w, b_h)
  - Confidence of object detection
- Since the grid has the size $S \times S$, the network predicts $(S^2 \cdot B \cdot 5)$ parameters for the bounding boxes

YOLO: Classification

- Predict C conditional **class probabilities per grid cell**
- $P(\text{Class}_i|\text{Object})$: Probability of class i given an object exists in the grid cell



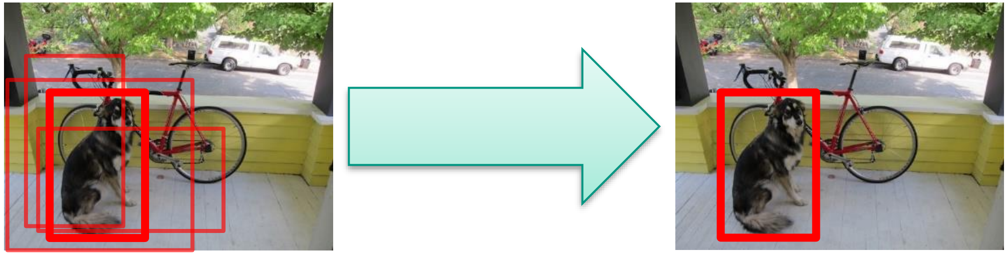
- The image on the right color-codes the most likely class label, but YOLO predicts probabilities for all classes
- Classification is only done once per cell not per bounding box
- The network predicts $(S^2 \cdot C)$ class probabilities

YOLO: Single Regression Model

- Single Regression
  - Input: $448 \times 448 \times 3$ RGB color image
  - Output: $S^2 \cdot (B \cdot 5 + C)$
    - Bounding box values: $S^2 \cdot B \cdot 5$
    - Class probabilities: $S^2 \cdot C$
- Model: Multi-layer CNN
  - Convolutional layers
  - Max-pooling layers
- Training
  - Pre-trained convolutional layers (ImageNet 1000-class)
  - Add layers to predict desired output

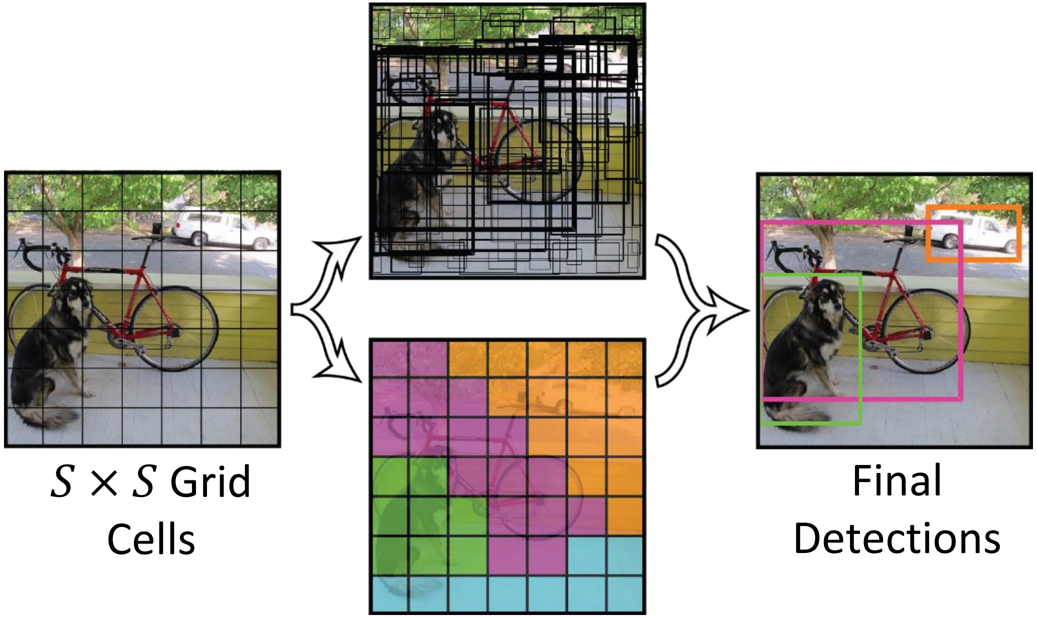
YOLO: Non-maximum Suppression

- The network predicts **multiple bounding boxes** per cell
- Most of the predicted boxes will have
  - Low confidence or
  - Overlap with other boxes with a higher confidence
- Non-maximum suppression **discards bounding boxes** which have
  - a confidence below a certain threshold or
  - the largest shared area with other boxes



YOLO: Pipeline

$S \times S \times B$
Bounding Boxes



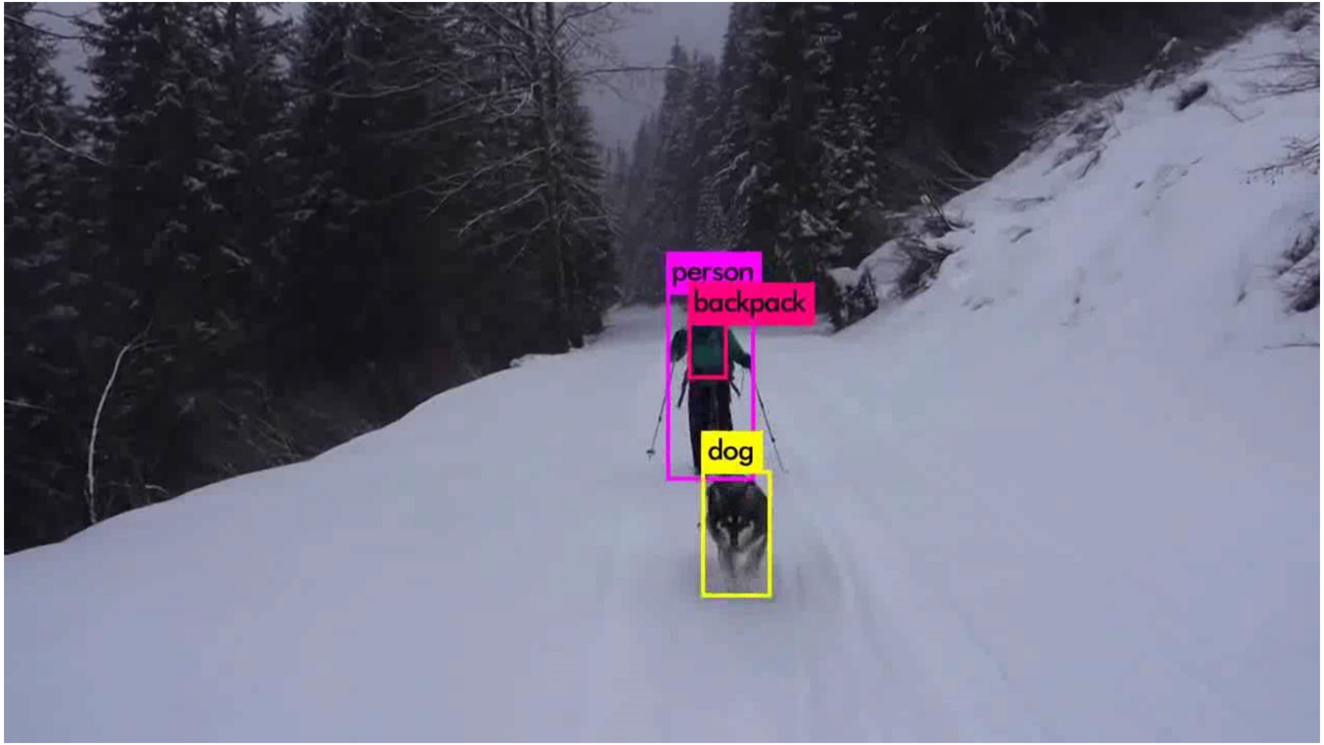
$S \times S$ Grid Cells

$S \times S$
Classifications

Final Detections

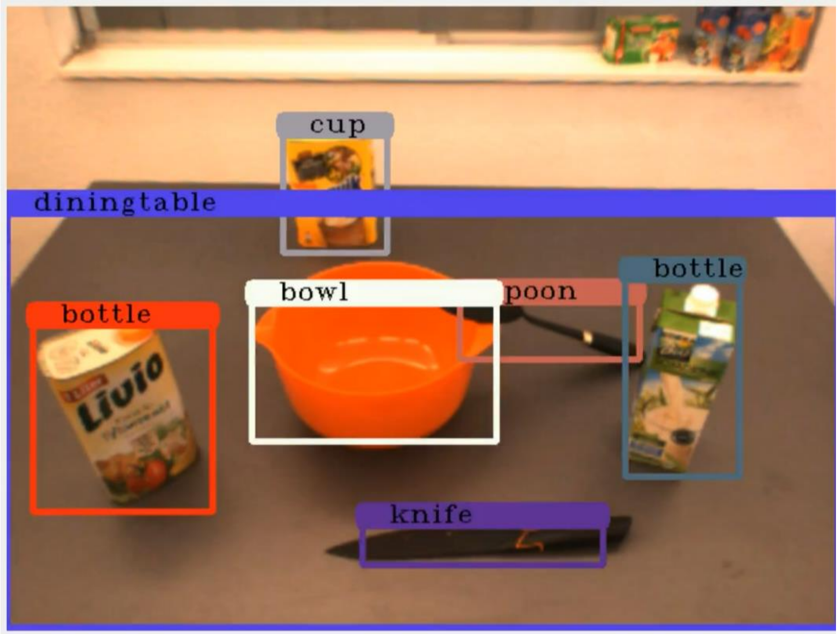
Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., “You only look once: Unified, real-time object detection”. ICVPR (2016)

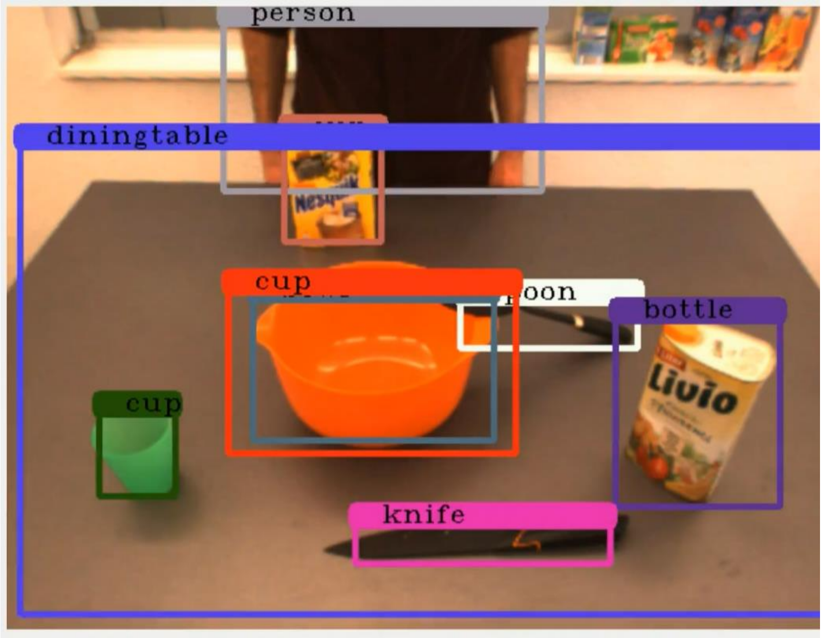
Object Detection using YOLO: Example



https://www.youtube.com/watch?v=MPU2HistivI

Example: YOLO on ARMAR-III





Segmentation

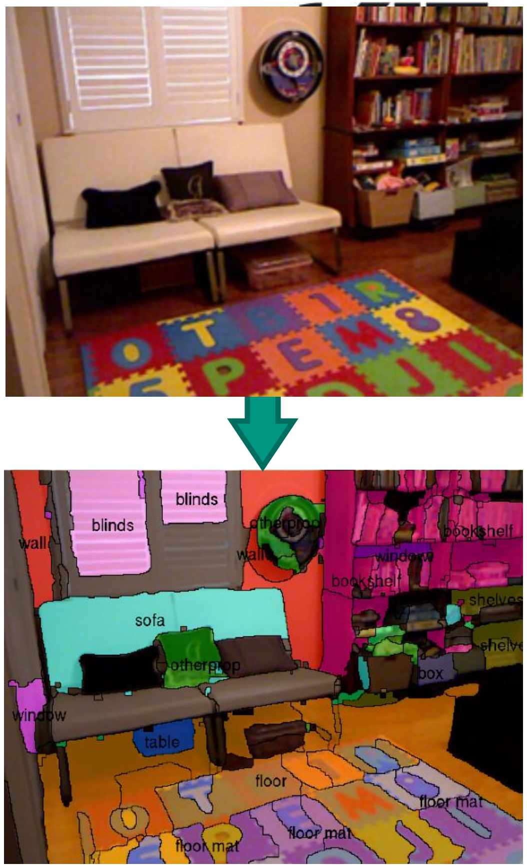
Problem:

- Given an input image (RGB/RGB-D) or point cloud.
- We want an **element-wise labelling** of …
  - segment ID (usually an integer)
  - Class, type, role, … (→ semantic segmentation)
  - instance ID (→ instance segmentation)

Question: What constitutes a "segment"?

- Regions of similar color, shape, appearance, …
- Objects, object parts, surfaces, …

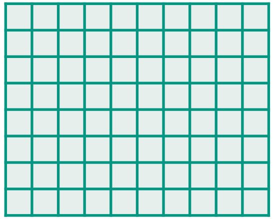
⇒ **Depends on the task!**



S. Gupta, P. Arbelaez, and J. Malik. "Perceptual organization and recognition of indoor scenes from RGB-D images," CVPR (2013).

Segmentation: Image vs. Point Cloud

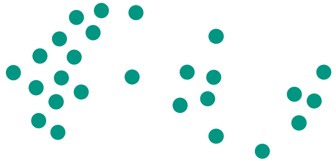
An image is a **2D grid of pixels** with RGB or RGB-D information.



- ⇒ Adjacency (i.e. neighboring pixels) is clear.
- Convolutions can be applied (→ filters).
- Resolution is homogeneous and (usually) constant.

A point cloud is a **collection of 3D points** with XYZ and RGB information.

- No specific order ⇒ Finding neighboring points is more difficult/time consuming.
- Resolution is inhomogeneous and variable (e.g. when registering multiple point clouds).
- Contains explicit 3D information ⇒ Allows to find …
  - clusters which are spatially separated
  - edges where (estimated) surface normals change



Segmentation: Example of Classical Methods

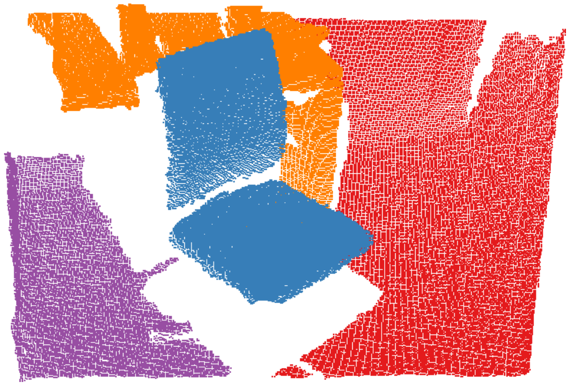
Input

Examples from PCL (Point Cloud Library)

- https://pointclouds.org/
- Usually require fine-tuning of parameters.

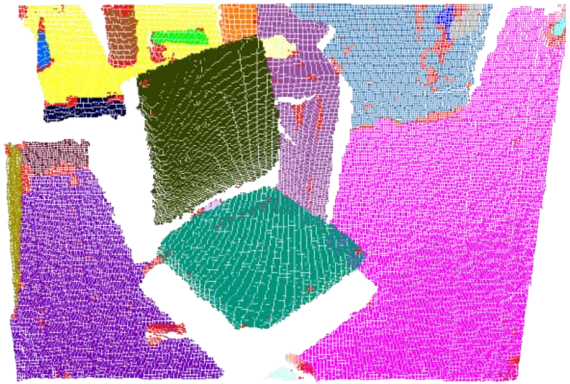
Euclidean Clustering

Nearest neighbor clustering using Euclidean distance



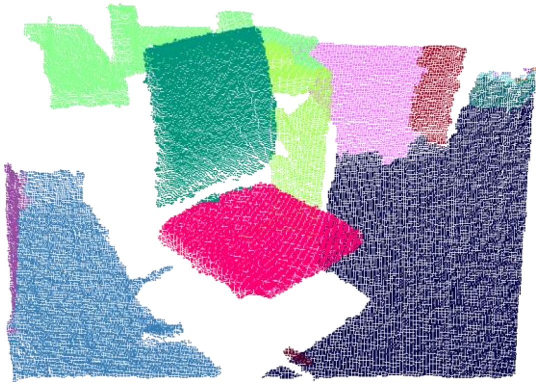
Region Growing

Grows segments from a seed until thresholds are met (e.g. normal).



LCCP (Local convexity connected patches)

Groups oversegmeted patches to convex shapes.



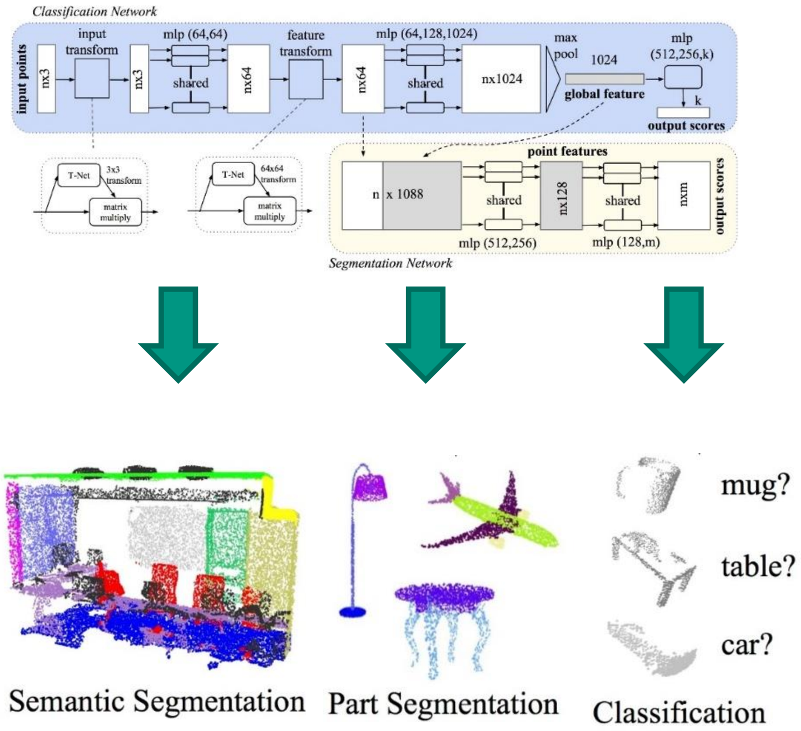
Point Cloud Segmentation with Neural Networks

- Point Clouds are inherently **unstructured**
- Neural Networks need ordered input

- PointNet applies symmetrical function, i.e. max-pooling, avg-pooling …

→ Result is independent of the **ordering of the point set**

→ Result is independent of the **number of input points**



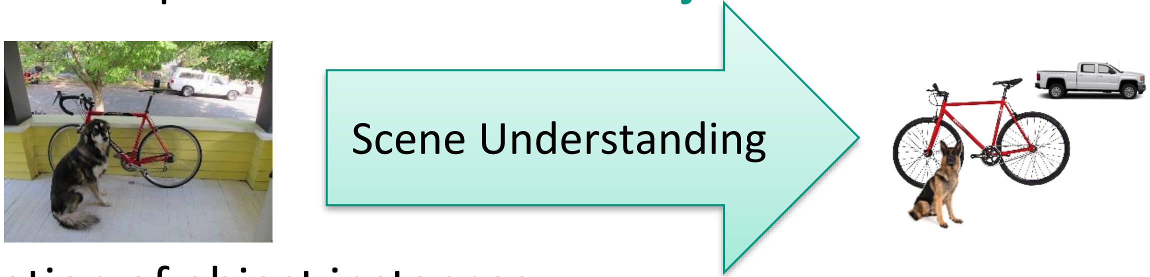
Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2016). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

Overview

- Introduction
- **Scene Representations**
  - Annotated Images
  - **Object Instances**
  - Object Relations
- Machine Learning for Object Relations
- Leveraging Object Relations

Scene Representation: Object Instances

- A scene can be represented as a **set of object instances**



- Representation of object instances
  - Class label
  - Instance identifier
  - Localization information
- Localization information
  - Object instance segmentation
  - 6D object pose

Object Instance Segmentation

- Object instance segmentation determines the **part of an image** which belong to the corresponding object instances
- Parts of an image can be determined on **different detail levels**
  - Approximate: Bounding Box
  - Exact: Pixel-wise Labeling

Bounding Box

Pixel-wise Labeling

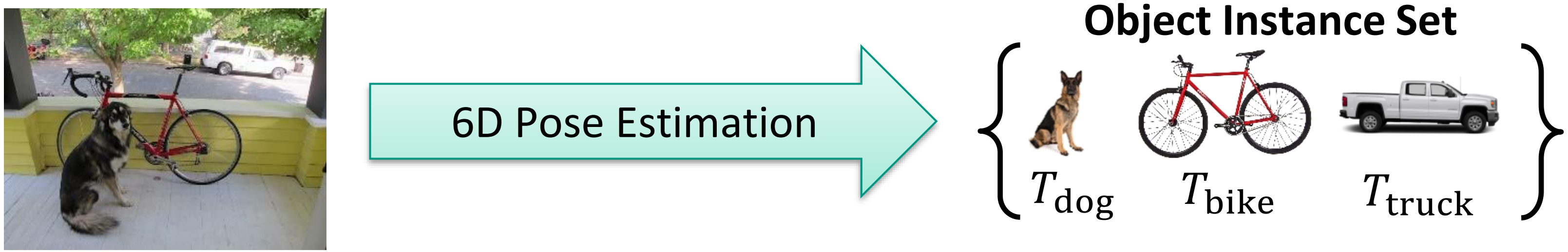
Instance Segmentation with Mask R-CNN

- Mask R-CNN performs the following tasks
  - **Multi-object** image classification and detection in form of bounding boxes
  - **Pixel-wise labeling** of each bounding box
- Method
  - Extraction of bounding boxes
    - **Region proposal network** for object candidates
    - Run image classification for **each proposed region**
  - Pixel-wise labeling
    - Fully-convolutional network for **semantic segmentation**
    - Run semantic segmentation for **each bounding box**

He, K., Gkioxari, G., Dollár, P., and Girshick, R, "Mask R-CNN". ICCV (2017)

6D Pose Estimation

- 6D pose estimation determines the **position and orientation** of a detected object in the **camera's coordinate system**
- Relevant for detection and localization of **known objects**
- Typical representations of a 6D pose:
  - Homogenous transformation matrix $T = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}$
    - with rotation matrix $R \in \mathbb{R}^{3\times3}$
    - and translation vector $t \in \mathbb{R}^3$
  - Orientation as unit quaternion $q \in H$ and a translation vector $t \in \mathbb{R}^3$

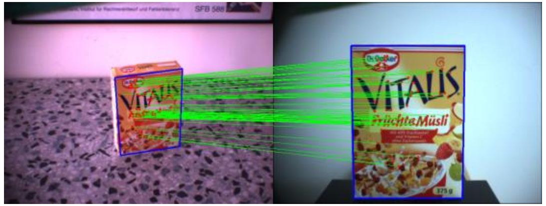


Pose Estimation using Harris/SIFT

- Goal: Robust, real-time pose estimation of known objects
- Idea: Combine
  - **Harris corner** detector and
  - **SIFT** (Scale Invariant Feature Transform) descriptors

- Steps
  - Hough Transform
  - RANSAC (Random Sample Consensus)
  - Least squares homography estimation

Details about corner detectors and features in a previous chapter



Azad, P., Asfour, T., and Dillmann, R., "Combining Harris interest points and the SIFT descriptor for fast scale-invariant object recognition." IROS (2009)

Pose Estimation using Harris/SIFT: Example

3D model drawn as an overlay to show the pose estimation result

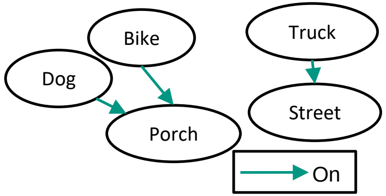
Overview

- Introduction
- **Scene Representations**
  - Annotated Images
  - Object Instances
  - **Object Relations**
- Machine Learning for Object Relations
- Leveraging Object Relations

Scene Representation: Object Relations

- A scene can be represented as **object instances** and their **relations**

Scene Understanding

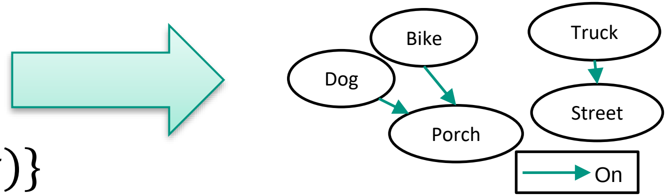


- Given a set of object instances $\mathcal{O}$, a binary relation between object instance pairs is element of the set $R \subseteq \mathcal{O} \times \mathcal{O}$
  - If and only if the relation holds between objects $o_i \in \mathcal{O}$ and $o_j \in \mathcal{O}$, then $\left(o_i, o_j\right) \in R$
- Example above:
  - Object instance set $\mathcal{O} = \{Dog, Bike, Truck, Porch, Street\}$
  - Binary relation $R_{on} = \{(Dog, Porch), (Bike, Porch), (Truck, Street)\}$

Object Relations: Graph Representation

- Binary object relations can be **encoded as a directed graph**
- A **directed graph** $G = (V, E)$ consists of
  - Set of vertices V
  - Set of ordered pairs called edges E
- **Construction** of a directed graph $G_R = (V_R, E_R)$ based on object relations $R \subseteq \mathcal{O} \times \mathcal{O}$
  - The set of object instances is the node set: $V_R = \mathcal{O}$
  - The binary relation is the edge set: $E_R = R$
- Example:

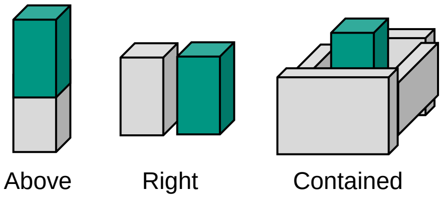
$$\mathcal{O} = \{Dog, Bike, Truck, Porch, Street\}$$
$$R_{on} = \{(Dog, Porch), (Bike, Porch), (Truck, Street)\}$$



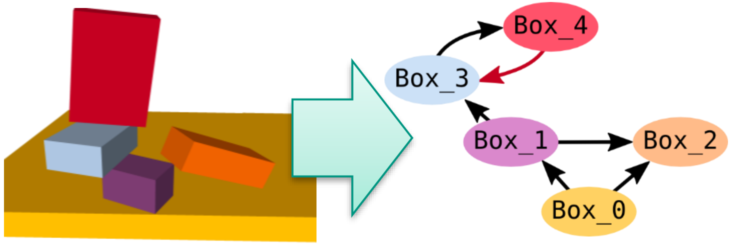
Object Relation Types

- Object relations differ in their
  - Type
  - Temporal context
- Example types of relations: **spatial, support**

Spatial Relations



Support Relations

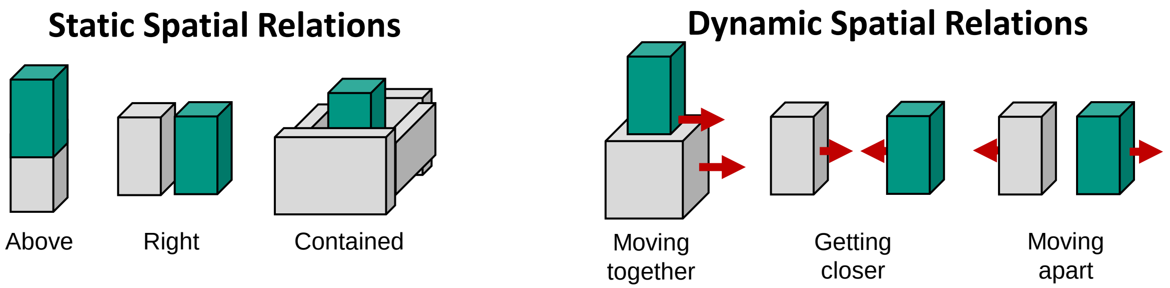


- Temporal context
  - Static: Consider a **single frame**
  - Dynamic: Consider changes **over time**

Spatial Relations

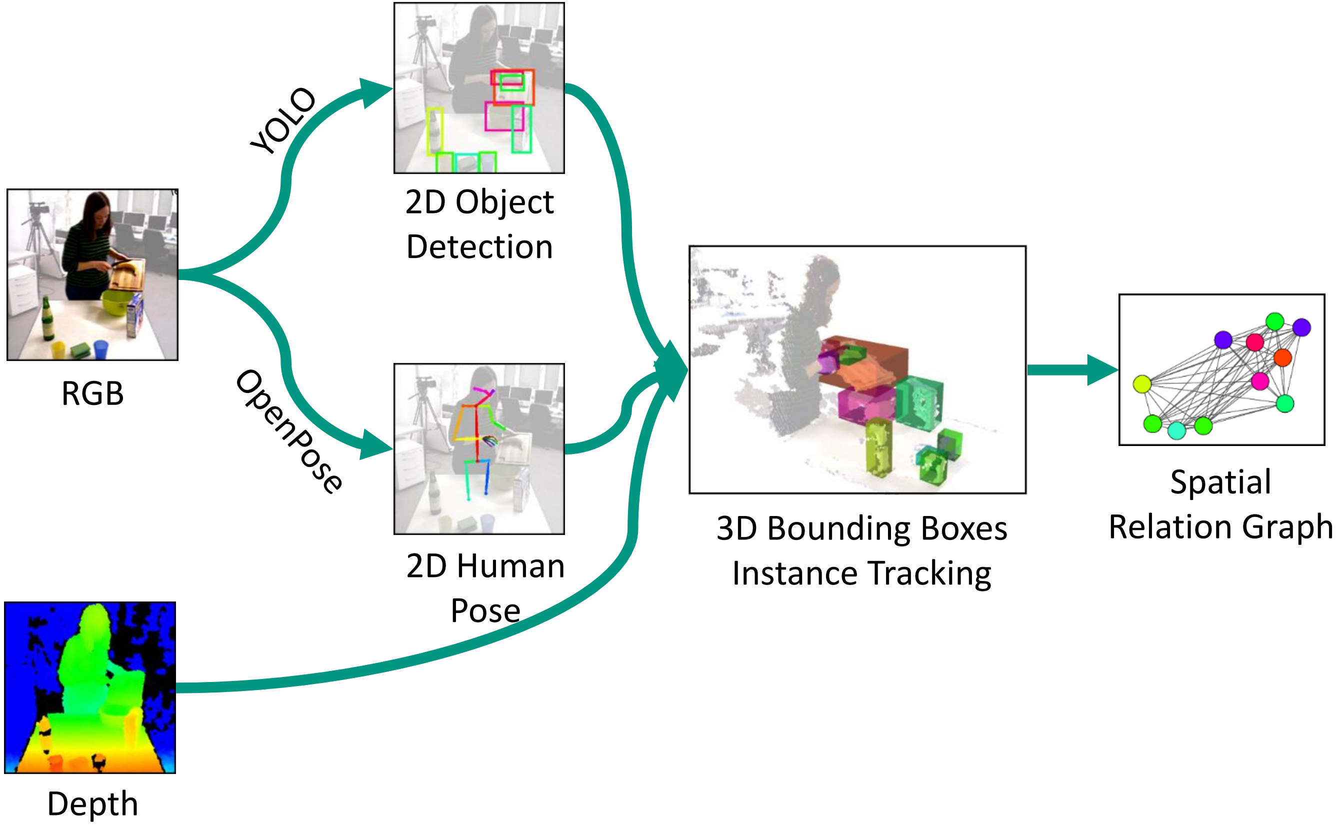
- Spatial relations describe the **relative position** of two objects

- Different temporal context of spatial relations
  - Static
  - Dynamic



Ziaeetabar, F., Aksoy, E. E., Wörgötter, F., and Tamosiunaite, M., “Semantic analysis of manipulation actions using spatial relations.” ICRA, 2017

Extraction of Spatial Relations from RGB-D

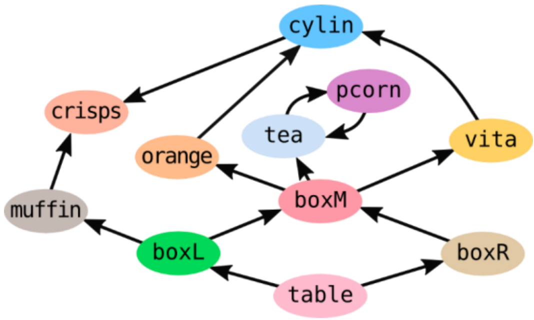


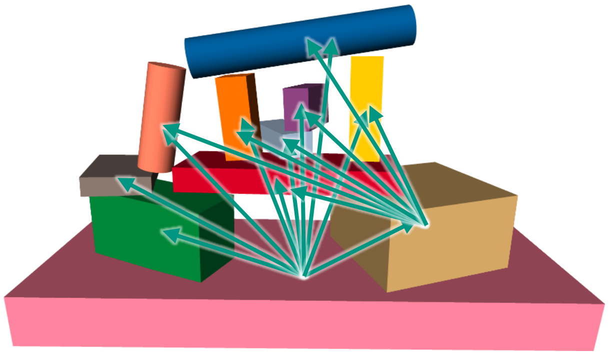
Support Relations

- For two objects $A, B \in \mathcal{O}$ we denote $\mathrm{SUPP}(A, B)$ iff. removing A causes B to lose its motionless state, i.e. A supports B.

> Mojtahedzadeh, R., Bouguerra, A., Schaffernicht, E., and Lilienthal, A. J., “Support relation analysis and decision making for safe robotic manipulation tasks”. Robotics and Autonomous Systems (2015)

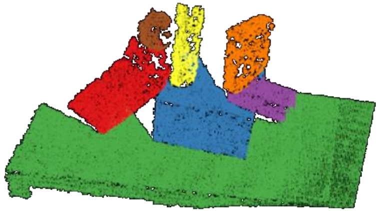
- Representation: Support Graph
  - ➔ Transitively reduced



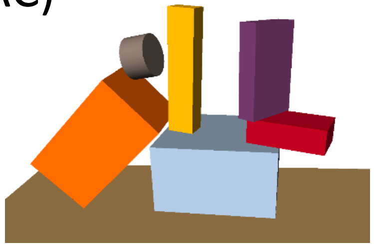


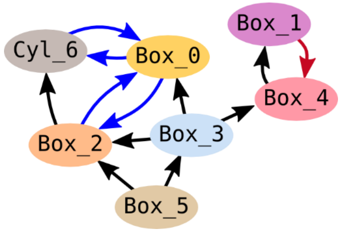
Extraction of Support Relations

- Input point cloud from RGB-D camera
- Segment into object hypotheses (LCCP, Region Growing)



- Extract object geometry (RANSAC)
- Build support graph





R. Kartmann, F. Paus, M. Grotz and T. Asfour, "Extraction of Physically Plausible Support Relations to Predict and Validate Manipulation Action Effects," Robotics and Automation Letters (RA-L), 2018

Manipulation of Object Relations

Problem so far: Scene → Relations

- Which relations are present in the scene? (discriminative)
- Useful for, e.g., action recognition

*Put the apple tea **in front of** the corny.*

*Let the apple tea be **on the other side** of the corny.*

Problem now:

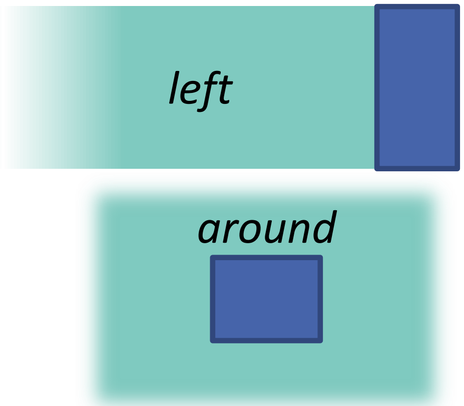
Where to place objects to realize a spatial relation ⇒ **Find suitable placing position**

$$(\mathrm{Scene}_t, \mathrm{Relation}) \rightarrow \mathrm{Scene}_{t+1}$$

- What is the best object placement to realize a spatial relation (generative)
- Useful for, e.g., action execution

Generative Model for Spatial Relations

Place an object according to a spatial relation
⇒ **Find suitable placing position**

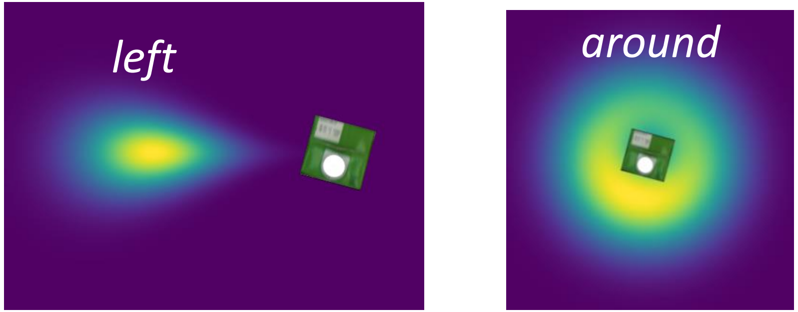


Idea: Use discriminative models?

- Problem: Which target position in the valid areas to choose?

Better: Use **generative** models.

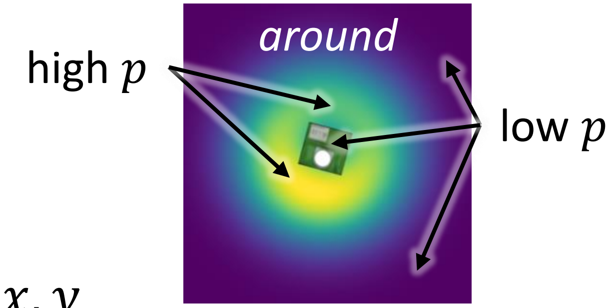
- Model a spatial relation as a **probability distribution over placing positions**.
- Sample from distribution to find suitable placing positions.



Spatial Relations: Polar Coordinates

Goal: Suitable representations of spatial relations in a two-dimensional plane.

- How to model, e.g., "*around*" in a simple manner?
  - (Multivariate) Gaussian: peak in the center
  - GMM[1]: complex, requires many components



Alternative idea: Use **distance and direction** instead of x, y

⇒ Probability distribution in **polar coordinates**

[1] Gaussian Mixture Model

Kartmann, R., Zhou, Y., Liu, D., Paus, F., and Asfour, T., "Representing Spatial Object Relations as Parametric Polar Distribution for Scene Manipulation Based on Verbal Commands." IROS 2020

Spatial Relations: Polar Distribution

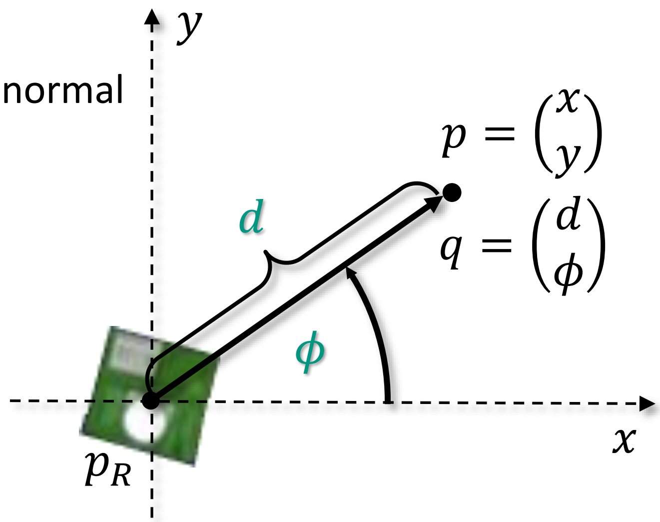
Use **distance and direction** instead of $x, y \Rightarrow$ Distribution in **polar coordinates**

Distribution defined in **polar coordinate system (PCS)** at reference object.

- Cartesian $p = (x \quad y)^T \rightarrow$ Polar $q = (d \quad \phi)^T \in \mathbb{R}^2$
- Distance: $d \sim \mathcal{N}\left(\mu_d, \sigma_d^2\right)$ (Gaussian)
- Angle: $\phi \sim \mathcal{M}\left(\mu_\phi, \sigma_\phi^2\right)$ (von Mises; circular normal distribution)
- Consider d and ϕ independent:

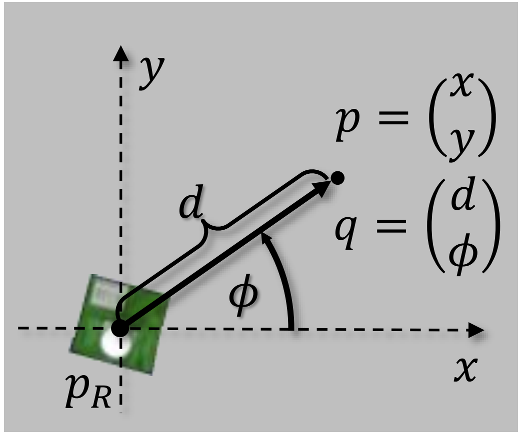
$$p(d, \phi) = p\left(d \middle| \mu_d, \sigma_d^2\right) \cdot p\left(\phi \middle| \mu_\phi, \sigma_\phi^2\right)$$

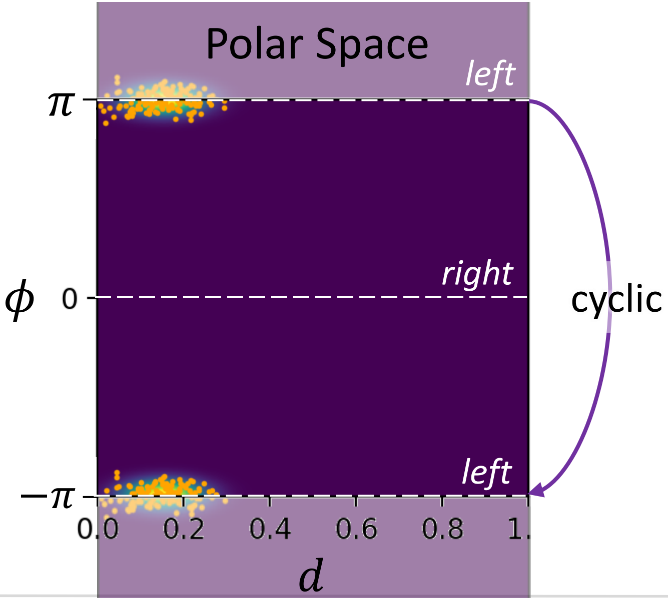
- Means $\mu_d > 0, \mu_\phi \in [-\pi, \pi]$
- Variances $\sigma_d^2, \sigma_\phi^2 \in \mathbb{R}_+$

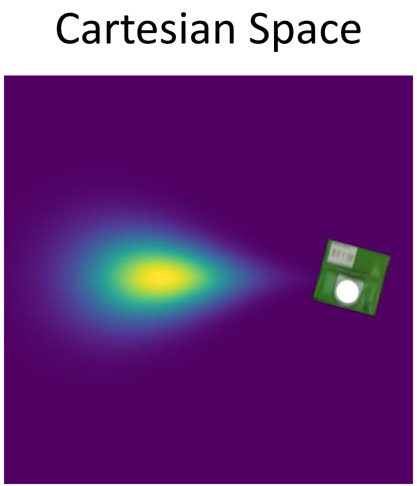


Spatial Relations: Estimating Polar Distribution

- Polar distribution has a simple form
  ⇒ Can be estimated from data using Maximum Likelihood Estimation (MLE).
- Example: "*Left*": $\mu_\phi \approx \pm\pi$





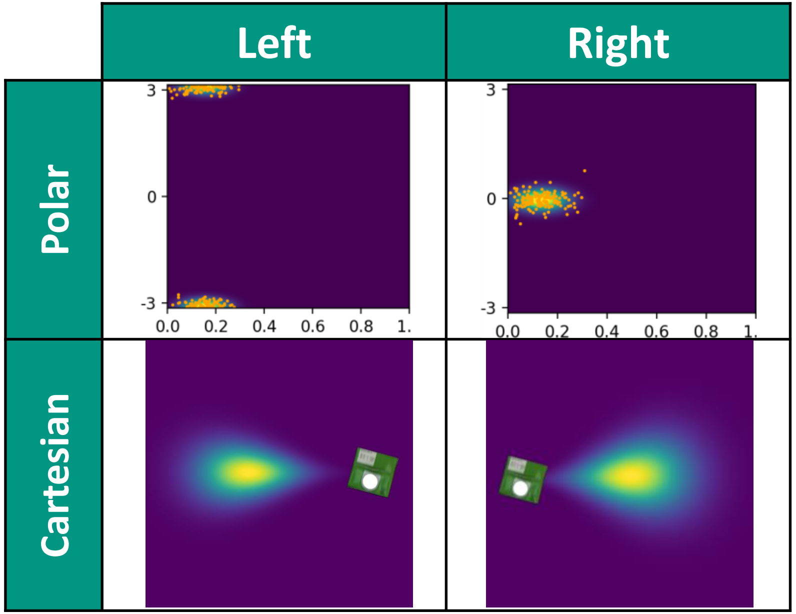


Spatial Relations: Static Relations

- Static relations: Depend only on reference object's current position.
- Define $d = 0$ as reference object's size ⇒ adapt the distribution to the object size.

Left vs *Right*:

- Mainly differ in mean direction μ_ϕ

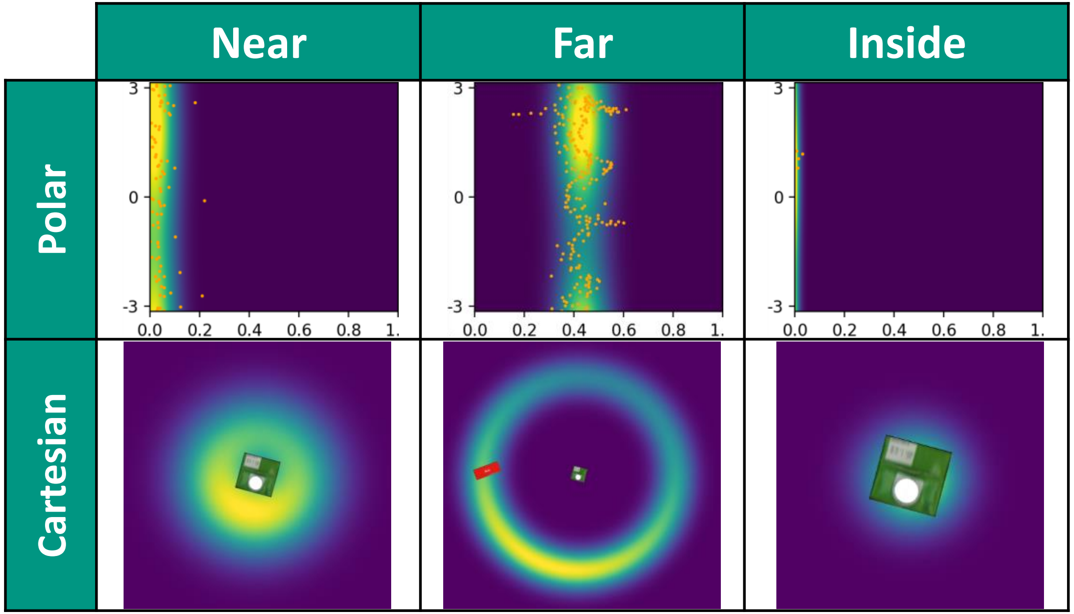


Spatial Relations: Static Relations

- Static relations: Depend only on reference object’s current position.
- Define $d = 0$ as reference object’s size ⇒ adapt the distribution to the object size.

Near vs. *Far* vs. *Inside:*

- Mainly differ in mean distance μ_d
- High direction variance σ_ϕ^2



Spatial Relations: Dynamic Relations

- Dynamic relations: Dependent on target object. ⇒ Align polar coordinate system so:
  - $d = 1 \triangleq$ initial distance to target object.
  - $\phi = 0 \triangleq$ initial direction to target object.

Closer:

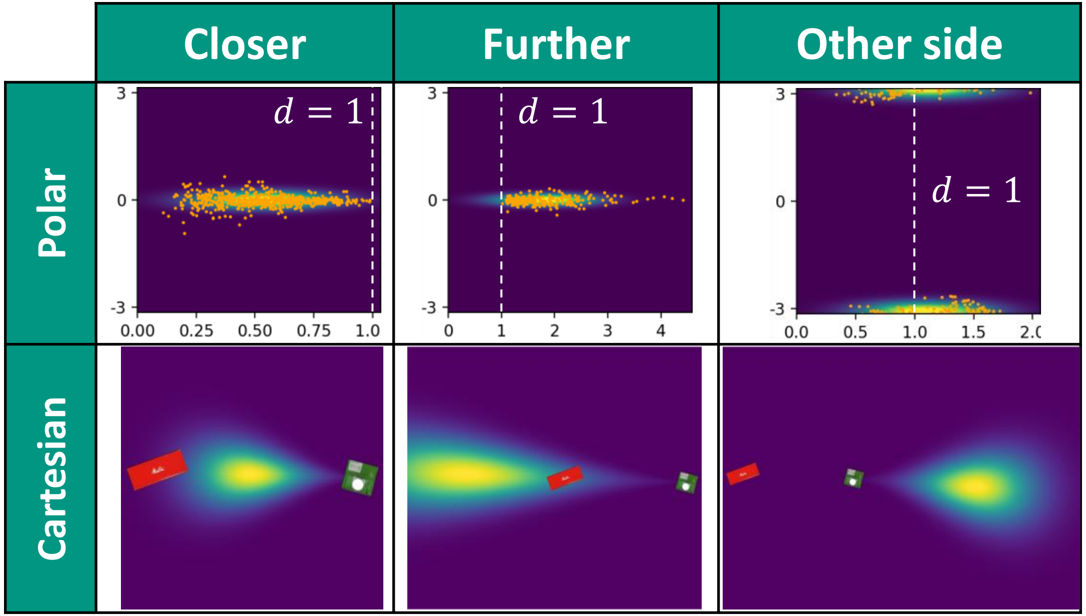
- $\mu_d \approx 0.5, \mu_\phi = 0$
- ⇒ between objects

Farther:

- $\mu_d > 1, \mu_\phi = 0$
- ⇒ farther than current distance

Other side:

- $\mu_d \approx 1, \mu_\phi = \pm\pi$
- ⇒ 180° away from current direction

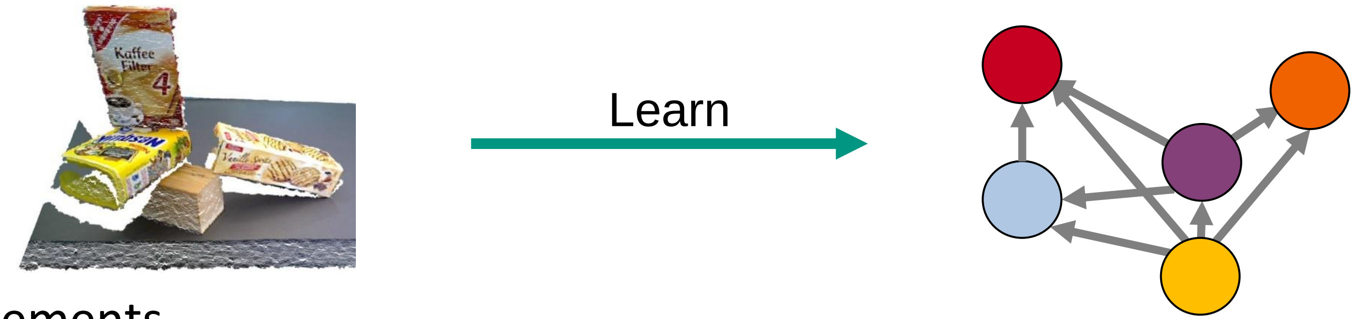


Overview

- Introduction
- Scene Representations
- **Machine Learning for Object Relations**
  - **Motivation**
  - Graph Networks
- Leveraging Object Relations

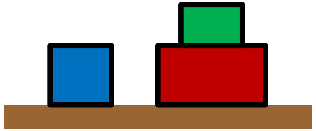
Machine Learning for Object Relations

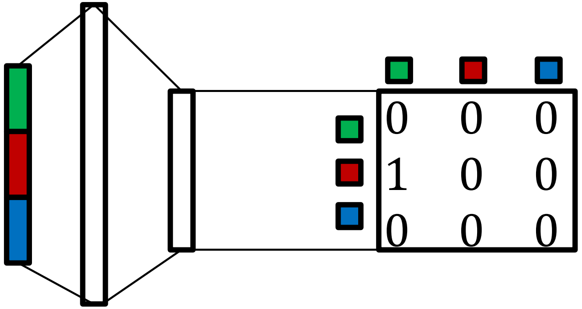
- Goal: Use **Machine Learning** (ML) to predict object relations



- Requirements
  - Number of objects is **variable**
  - Result should be **order invariant**
- Problems with standard ML approaches
  - Input size must be fixed
  - Order of the input is relevant

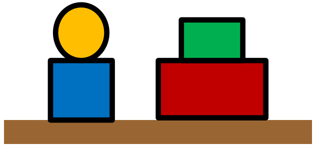
ML for Object Relations: Classical Approach

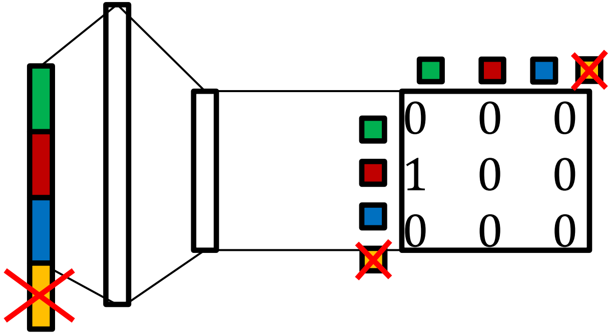




- Stack objects properties (pose, color, size, …) into a single input vector
- Use a Multi-Layer Perceptron to produce the desired output
- Encode the output as an adjacency matrix containing support relations

ML for Object Relations: Problems

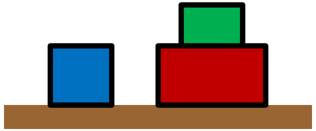


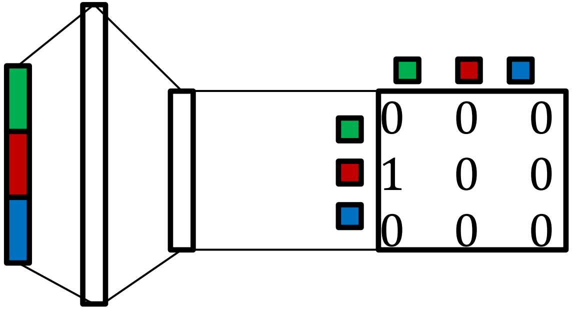


How to handle **variable number** of objects?

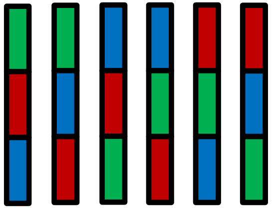
- Stacked input vector has different dimension
- Output matrix has different dimension

ML for Object Relations: Problems

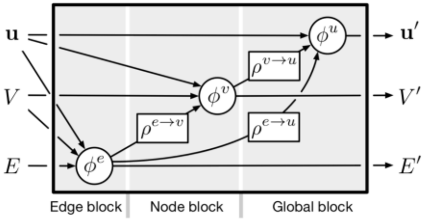




- How to handle **order** of objects?
  - Order is arbitrary
  - Train on all combinations of n objects: $n!$
  - Computationally expensive



- **Solution: Graph Networks**



Overview

- Introduction
- Scene Representations
- **Machine Learning for Object Relations**
  - Motivation
  - **Graph Networks**
- Leveraging Object Relations

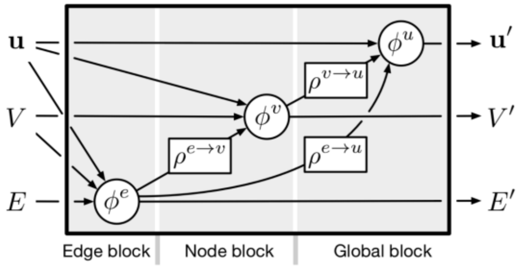
Graph Networks

- Graph networks operate on **graph structures** (input and output are graphs)

- Graph $G = (V, E, \boldsymbol{u})$
  - Vertices V
  - Edges E
  - Global attributes $\boldsymbol{u}$

- Central building block is a **GN block**
  - Input graph $G = (V, E, \boldsymbol{u})$
  - Output graph $G' = (V', E', \boldsymbol{u}')$
  - Vertices, edges and attributes can change

- Code: https://github.com/deepmind/graph_nets



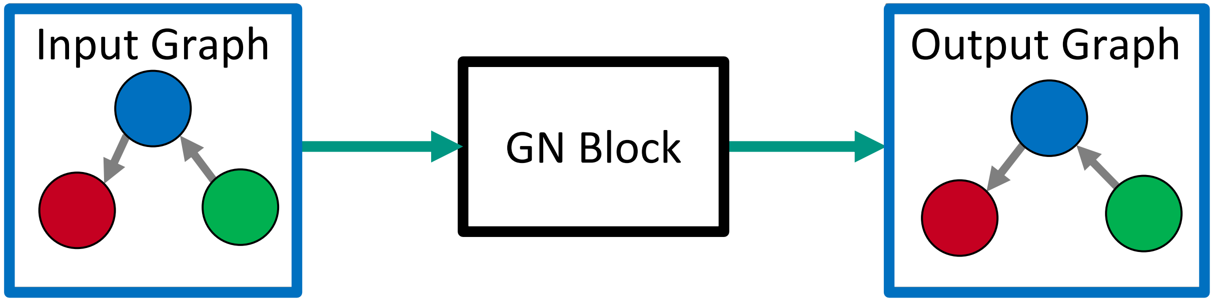
Full GN Block

Battaglia, P. W. et al. “Relational inductive biases, deep learning, and graph networks.” arXiv preprint arXiv:1806.01261 (2018).

Graph Networks: GN Block

- Graph Networks (GNs) as proposed by Battaglia et al., 2018

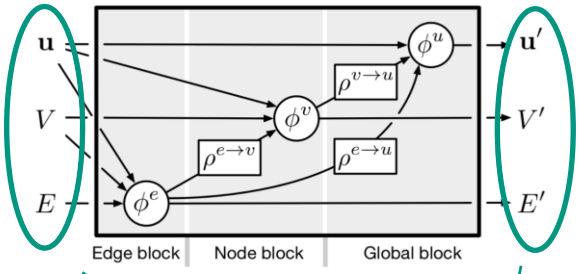
- Basic building block: GN Block

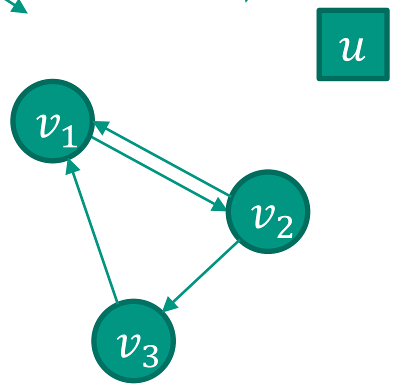


Battaglia, P. W. et al. “Relational inductive biases, deep learning, and graph networks.” arXiv preprint arXiv:1806.01261 (2018).

Graph Networks: Graph Representation

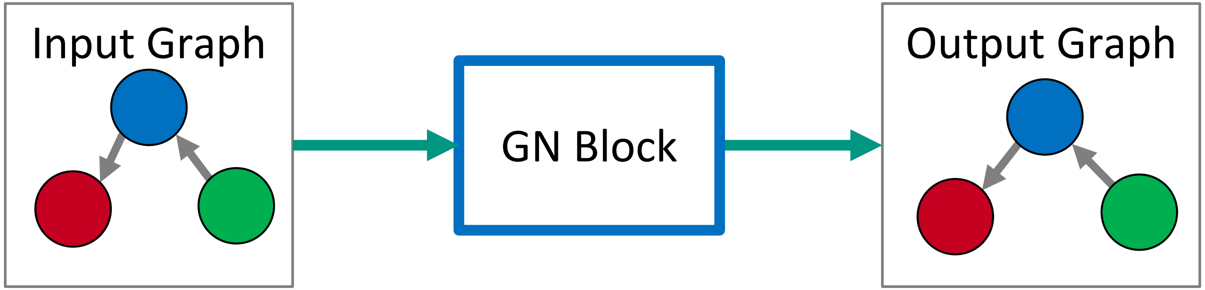
- Directed, attributed multi-graph
- Nodes $V = \{\, \boldsymbol{v}_i \mid i \in [1, N_v] \,\}$
  - Node attributes $\boldsymbol{v}_i \in \mathbb{R}^{d_v}$
- Edges $E = \{\, (\boldsymbol{e}_k, r_k, s_k) \mid k \in [1, N_e], r_k, s_k \in [1, N_v] \,\}$
  - Edge attributes $\boldsymbol{e}_k \in \mathbb{R}^{d_e}$
  - Receiver node index r_k
  - Sender node index s_k
- Global attribute $u \in \mathbb{R}^{d_u}$





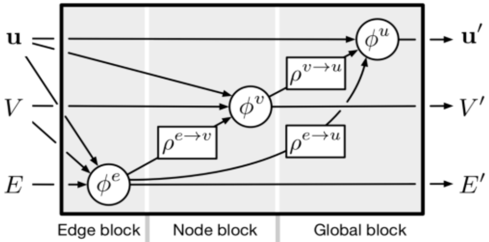
Graph Networks: GN Block

- Graph Networks (GNs) as proposed by Battaglia et al., 2018
- Basic building block: GN Block

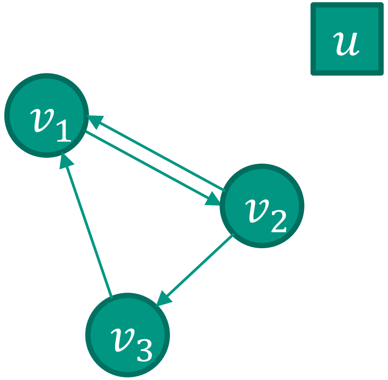


Graph Networks: GN Block

- Consists of three update and three aggregation functions
  - Update Φ^e, Φ^v, Φ^u
  - Aggregate $\rho^{e\to v}, \rho^{e\to u}, \rho^{v\to u}$
- Process:



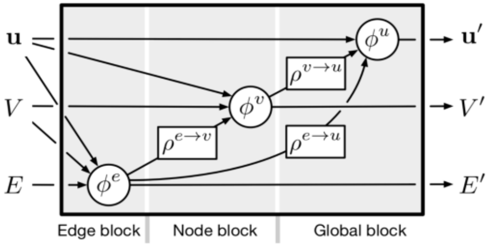
Full GN Block

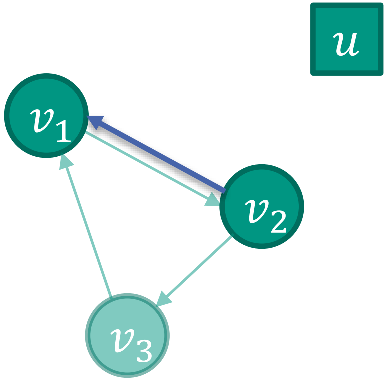


Graph Networks: GN Block

- Consists of three update and three aggregation functions
  - Update Φ^e, Φ^v, Φ^u
  - Aggregate $\rho^{e\to v}, \rho^{e\to u}, \rho^{v\to u}$
- Process:
  1. Update edges depending on sender, receiver and global state

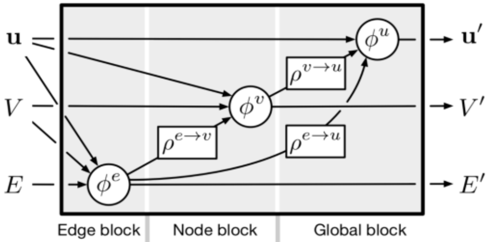
$$\boldsymbol{e}'_k = \Phi^e\left(\boldsymbol{e}_k, v_{r_k}, v_{s_k}, \boldsymbol{u}\right)$$





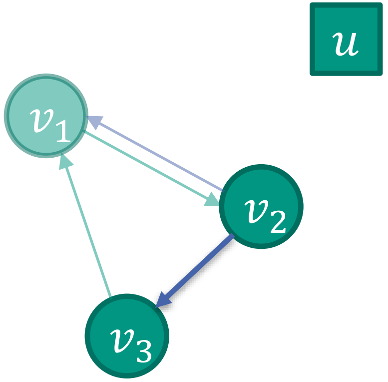
Graph Networks: GN Block

- Consists of three update and three aggregation functions
  - Update Φ^e, Φ^v, Φ^u
  - Aggregate $\rho^{e\to v}, \rho^{e\to u}, \rho^{v\to u}$



- Process:
  1. Update edges depending on sender, receiver and global state

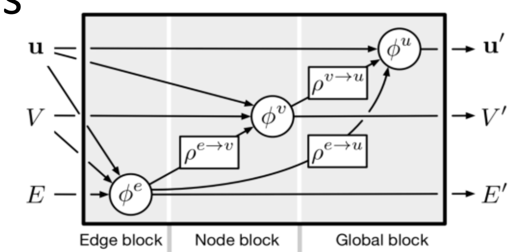
$$\boldsymbol{e}'_k = \Phi^e\left(\boldsymbol{e}_k, v_{r_k}, v_{s_k}, \boldsymbol{u}\right)$$

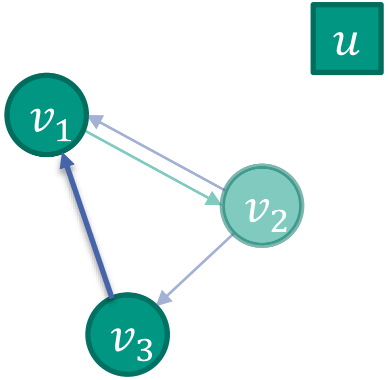


Graph Networks: GN Block

- Consists of three update and three aggregation functions
  - Update Φ^e, Φ^v, Φ^u
  - Aggregate $\rho^{e\to v}, \rho^{e\to u}, \rho^{v\to u}$
- Process:
  1. Update edges depending on sender, receiver and global state

$$\boldsymbol{e}'_k = \Phi^e(\boldsymbol{e}_k, v_{r_k}, v_{s_k}, \boldsymbol{u})$$

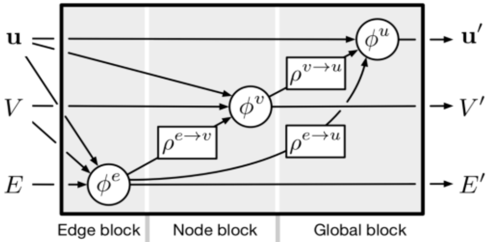


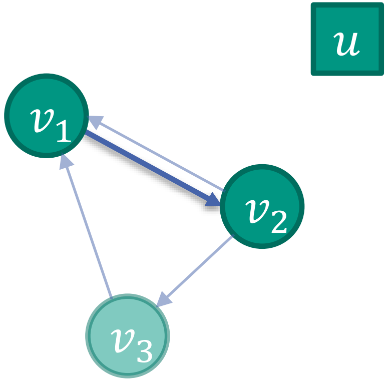


Graph Networks: GN Block

- Consists of three update and three aggregation functions
  - Update Φ^e, Φ^v, Φ^u
  - Aggregate $\rho^{e\rightarrow v}, \rho^{e\rightarrow u}, \rho^{v\rightarrow u}$
- Process:
  1. Update edges depending on sender, receiver and global state

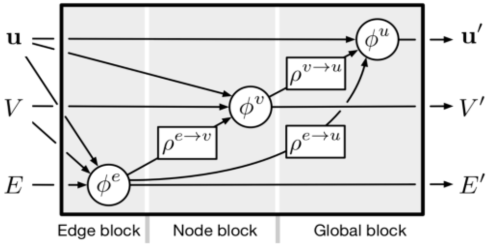
$$\boldsymbol{e}'_k = \Phi^e(\boldsymbol{e}_k, v_{r_k}, v_{s_k}, \boldsymbol{u})$$





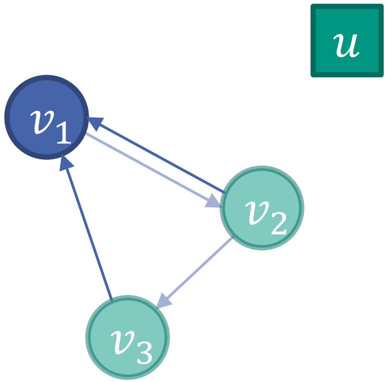
Graph Networks: GN Block

- Consists of three update and three aggregation functions
  - Update Φ^e, Φ^v, Φ^u
  - Aggregate $\rho^{e\to v}, \rho^{e\to u}, \rho^{v\to u}$



- Process:
  1. Update edges depending on sender, receiver and global state

$$\boldsymbol{e}'_k = \Phi^e\left(\boldsymbol{e}_k, v_{r_k}, v_{s_k}, \boldsymbol{u}\right)$$



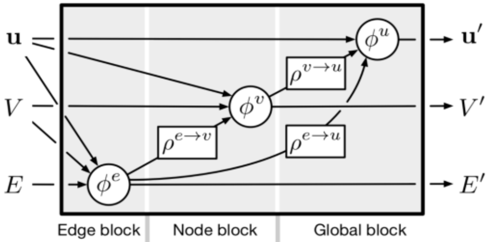
  2. Update receiving nodes

$$\boldsymbol{v}_i{}' = \Phi^v\left(\boldsymbol{v}_i, \boldsymbol{u}, \rho^{e\to v}(E'_i)\right)$$

E'_i: Incoming edges to $\boldsymbol{v}_i$, i.e. $r_k = i$

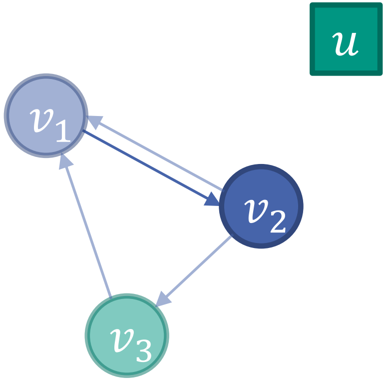
Graph Networks: GN Block

- Consists of three update and three aggregation functions
  - Update Φ^e, Φ^v, Φ^u
  - Aggregate $\rho^{e\to v}, \rho^{e\to u}, \rho^{v\to u}$
- Process:



1. Update edges depending on sender, receiver and global state

$$\boldsymbol{e}_k' = \Phi^e\left(\boldsymbol{e}_k, v_{r_k}, v_{s_k}, \boldsymbol{u}\right)$$



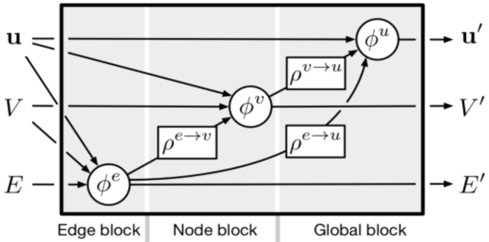
2. Update receiving nodes

$$\boldsymbol{v}_i' = \Phi^v\left(\boldsymbol{v}_i, \boldsymbol{u}, \rho^{e\to v}(E_i')\right)$$

E_i': Incoming edges to $\boldsymbol{v}_i$, i.e. $r_k = i$

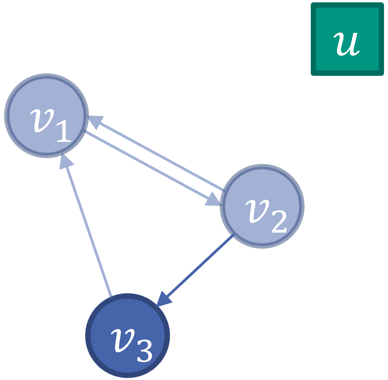
Graph Networks: GN Block

- Consists of three update and three aggregation functions
  - Update Φ^e, Φ^v, Φ^u
  - Aggregate $\rho^{e\to v}, \rho^{e\to u}, \rho^{v\to u}$



- Process:
  1. Update edges depending on sender, receiver and global state

$$\boldsymbol{e}'_k = \Phi^e\left(\boldsymbol{e}_k, v_{r_k}, v_{s_k}, \boldsymbol{u}\right)$$



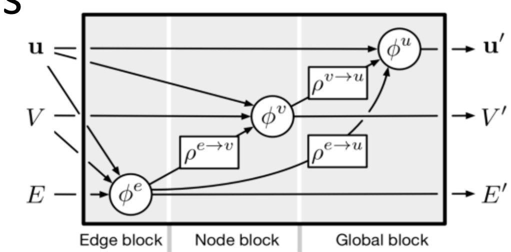
  2. Update receiving nodes

$$\boldsymbol{v}_i{}' = \Phi^v\left(\boldsymbol{v}_i, \boldsymbol{u}, \rho^{e\to v}(E'_i)\right)$$

E'_i: Incoming edges to $\boldsymbol{v}_i$, i.e. $r_k = i$

Graph Networks: GN Block

- Consists of three update and three aggregation functions
  - Update Φ^e, Φ^v, Φ^u
  - Aggregate $\rho^{e\rightarrow v}, \rho^{e\rightarrow u}, \rho^{v\rightarrow u}$



- Process:
  1. Update edges depending on sender, receiver and global state

$$\boldsymbol{e}'_k = \Phi^e(\boldsymbol{e}_k, v_{r_k}, v_{s_k}, \boldsymbol{u})$$

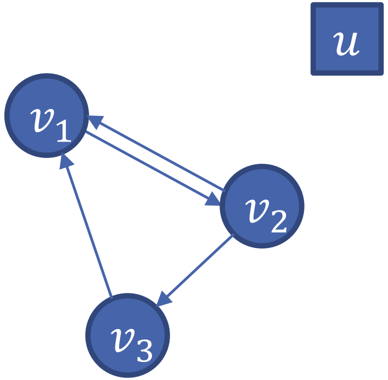
  2. Update receiving nodes

$$\boldsymbol{v}_i{}' = \Phi^v(\boldsymbol{v}_i, \boldsymbol{u}, \rho^{e\rightarrow v}(E'_i))$$

E'_i: Incoming edges to $\boldsymbol{v}_i$, i.e. $r_k = i$

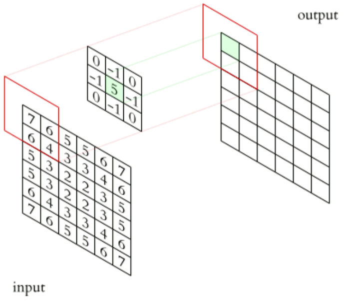
  3. Update the global state

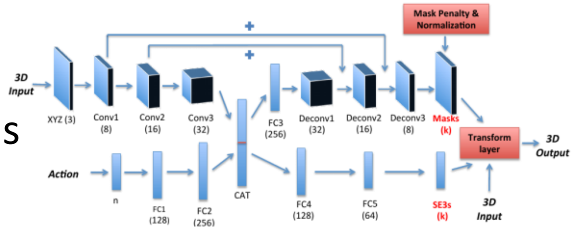
$$\boldsymbol{u}' = \Phi^u(\boldsymbol{u}, \rho^{e\rightarrow u}(E'), \rho^{v\rightarrow u}(V'))$$



Inductive Bias

- Structuring a learning problems introduces **Inductive Bias**
- Examples:
  - CNNs use convolutional kernels
    - **Translational invariance**: Features can be extracted independent of their pixel position (same kernel)
    - **Locality**: Features depend only on neighboring pixels
  - SE3-Nets (Byravan and Fox, 2017) use SE(3) transformations
    - Objects move like **rigid bodies**





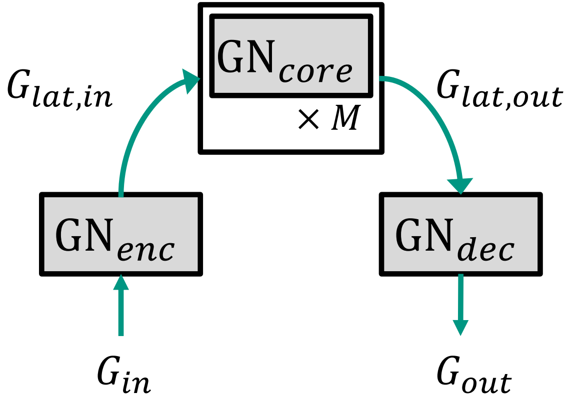
Graph Networks: Relational Inductive Bias

- Update functions Φ^e, Φ^v, Φ^u
  - Are **reused** for all nodes and edges (similar to convolutional kernels)
  - Implementation: MLP, CNN
- Aggregate functions $\rho^{e\rightarrow v}, \rho^{e\rightarrow u}, \rho^{v\rightarrow u}$
  - **Invariant to permutations** of the input
  - Variable number of arguments
  - Implementation: sum, average, min, max
- Edges determine which objects interact
  ➔ Computational dependency reflects relational structures
- Reuse of update function
  ➔ Allows **combinatorial generalization**

Graph Networks: Encode-Process-Decode

- GN blocks can be **combined** into more complex models
- A common pattern is **Encode-Process-Decode**
  - Encode the input graph G_{in} into the latent representation $G_{lat,in}$
  - Run a GN block multiple times ($\times M$) on $G_{lat,in}$ producing $G_{lat,out}$
  - Decode the latent representation $G_{lat,out}$ into the output graph G_{out}

- Encoding into a **latent representation** allows for ML efficient data processing
- Multiple processing steps allow the network to **propagate information along the edges** of the graph



Overview

- Introduction
- Scene Representations
- Machine Learning for Object Relations
- **Leveraging Object Relations**
  - Bimanual Action Recognition
  - Placing Objects Based on Verbal Commands
  - Support Relations for Safe Bimanual Manipulation

Overview

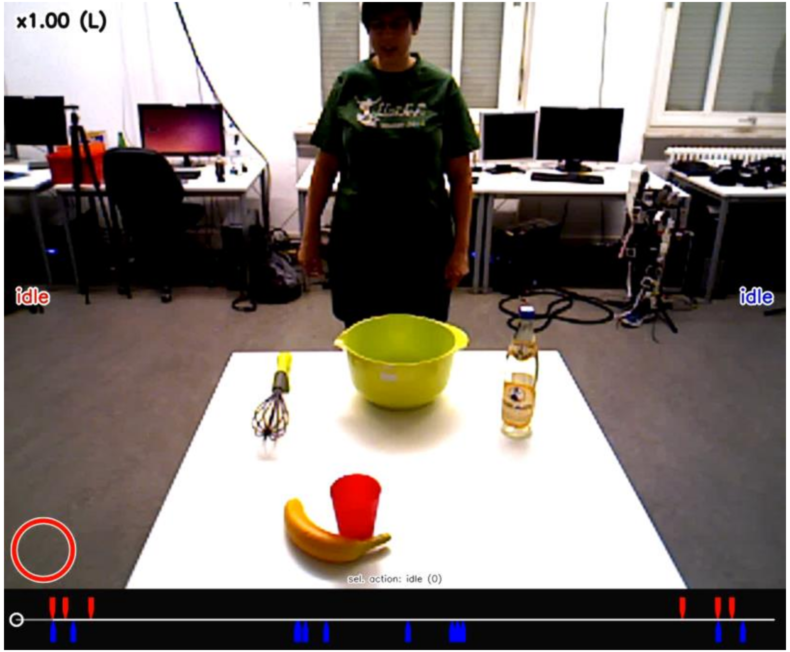
- Introduction
- Scene Representations
- Machine Learning for Object Relations
- **Leveraging Object Relations (@H²T)**
  - **Bimanual Action Recognition**
  - Placing Objects Based on Verbal Commands
  - Support Relations for Safe Bimanual Manipulation

Bimanual Action Recognition: Goal

- In a bimanual manipulation task, both hands perform different actions like holding, pouring, stirring, etc.
- **Goal:** Recognize **actions of both hands**
- **Idea:** Use **spatial relations** between hands and objects

Challenges:

- Variable number of unordered objects
- Relevant and irrelevant objects



Ground Truth

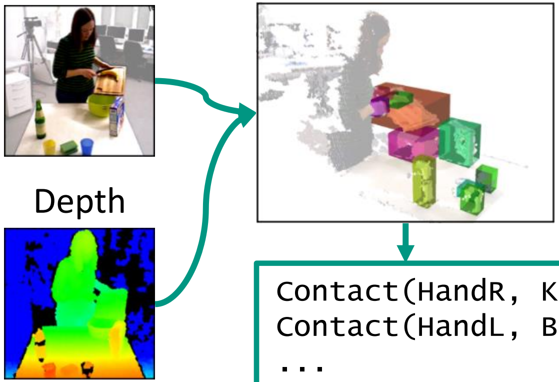
Bimanual Action Recognition: Preprocessing

- Extract spatial relations from RGB-D video of task execution.
- Per frame:
  - Estimate bounding boxes of hands and objects
  - Extract spatial relations between them
- Result:
  - List of predicates
  - Each predicate denotes one spatial relation between a pair of objects

RGB

Depth

3D bounding boxes instance tracking



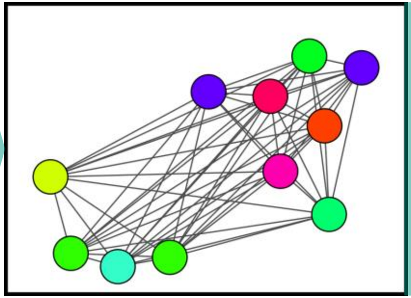
Spatial relations as list of predicates

```
Contact(HandR, Knife)
Contact(HandL, Board)
...
Above(Knife, Board)
Above(Banana, Board)
Above(Board, Bowl)
...
FixedMovingTogether(HandR, Knife)
...
```

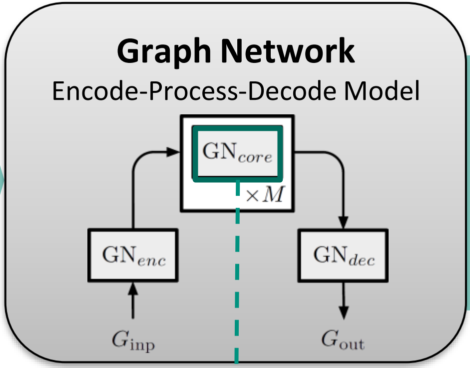
Bimanual Action Recognition: Overview

```
Contact(HandR, Knife)
Contact(HandL, Board)
...
Above(Knife, Board)
Above(Banana, Board)
Above(Board, Bowl)
...
FixedMovingTogether(HandR,
Knife)
...
```

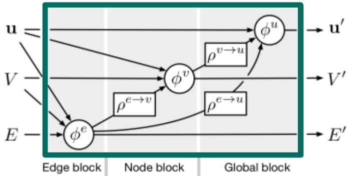
Spatial Relations as List of Predicates

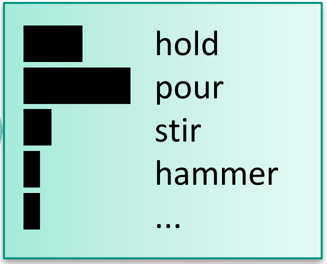


Spatial Relations as Scene Graph



GN Block



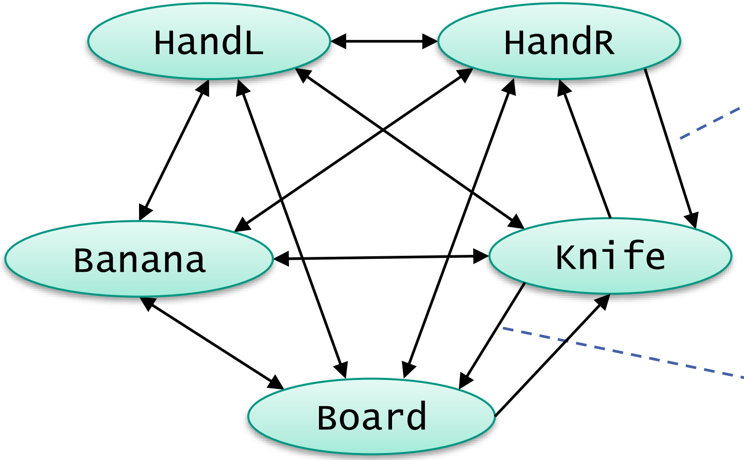


Distribution of ***Action Probabilities***

Dreher, C. R. G., Wächter, M. and Asfour, T., *Learning Object-Action Relations from Bimanual Human Demonstration Using Graph Networks*, Robotics and Automation Letters (RA-L), 2020

Bimanual Action Recognition: Build Scene Graph

- Encode spatial relations in scene graph:
  - Nodes: Hands and objects
  - Edges: Relations between hands and objects

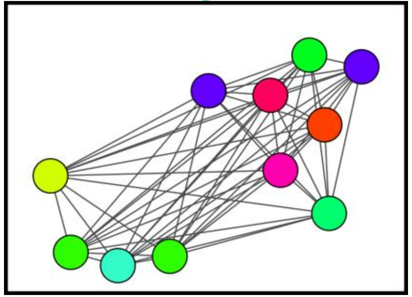


Contact	✓
Above	✗
FixedMovingTogether	✓
…	

Contact	✓
Above	✓
…	

List of Predicates

```
Contact(HandR, Knife)
Contact(HandL, Board)
...
Above(Knife, Board)
Above(Banana, Board)
Above(Board, Bowl)
...
FixedMovingTogether(HandR,
Knife)
...
```

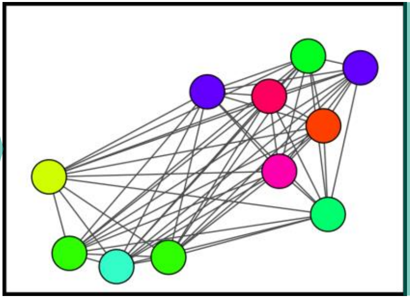


Scene Graph

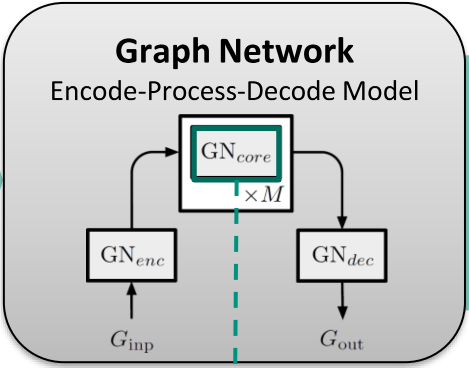
Bimanual Action Recognition: Overview

```
Contact(HandR, Knife)
Contact(HandL, Board)
...
Above(Knife, Board)
Above(Banana, Board)
Above(Board, Bowl)
...
FixedMovingTogether(HandR,
Knife)
...
```

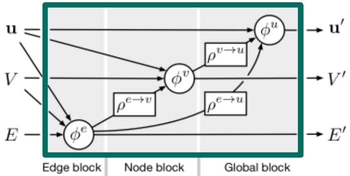
Spatial Relations as List of Predicates

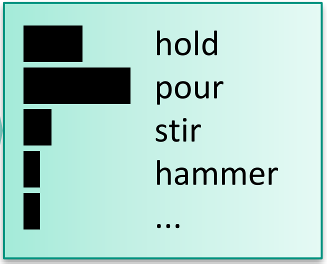


Spatial Relations as Scene Graph



GN Block





Distribution of ***Action Probabilities***

Dreher, C. R. G., Wächter, M. and Asfour, T., *Learning Object-Action Relations from Bimanual Human Demonstration Using Graph Networks*, RA-L (2020)

Bimanual Action Recognition: Graph Network (I)

- Encode **scene graph** for Graph Network: **Input graph**
  - Node attributes: 1-hot encoding of object class

$$v_i = (0\ 1\ 0\ \ldots\ 0) \in \{0,1\}^{12}$$ ← number of object classes

HandR →

0	1	0	0	0	0	0	0	0	0	0	0

HandR

  - Edge attributes: 0/1-vector of relations

$$e_k = (1\ 0\ 1\ 0\ \ \ldots\ 0) \in \{0,1\}^{15}$$ ← number of spatial relations

Contact	✓
Above	×
FixedMoving Together	✓
…	

→

1	0	1	0	0	0	0	1	0	0	0	1	0	0	0

Contact

FixedMovingTogether

Above

  - Global attribute: unused

Bimanual Action Recognition: Graph Network (II)

- Encode **action classification** for Graph Network: **Output graph**
  - Node and edge attributes: unused
  - Global attribute output: Action probabilities (softmax layer)

$$u' = (p_1 \;\; p_2 \;\; p_3 \ldots \; p_{14}) \in [0,1]^{14}, \quad \left(\textstyle\sum_i p_i = 1\right)$$

  - Global attribute target (label): 1-hot encoding of action (**right hand**)

$$u' = (0\;1\;0\;\;\ldots 0) \in \{0,1\}^{14}$$

number of action classes

0	0	0	0	0	0	0	1	0	0	0	0	0	0
	Cutting						Pouring					Sawing	

- **Left hand**: next slide

Bimanual Action Recognition: Graph Network (III)

- Global attribute target (label): 1-hot encoding of action (**right hand**)
- Recognition of **left hand's action:**
  - **Mirror** input graph (`HandL` ↔ `HandR`, `left` ↔ `right`) and **classify again**.
  - Same as mirroring RGB image and running processing (feature extraction) again.
- **Bimanual** action recognition:
  - Run the graph network **twice**.
  - 1x on original scene graph + 1x on mirrored scene graph

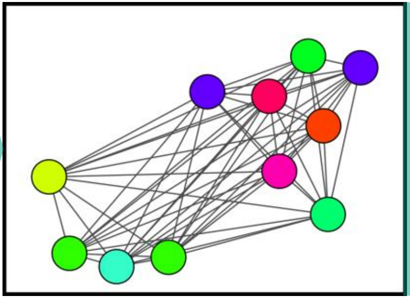
⇒ **Inductive bias**: Left and right hand behave similarly.

  - Network can be smaller (e.g., output size: 14 instead of 28)
  - Reuse data for both hands (2 scene graphs per frame)

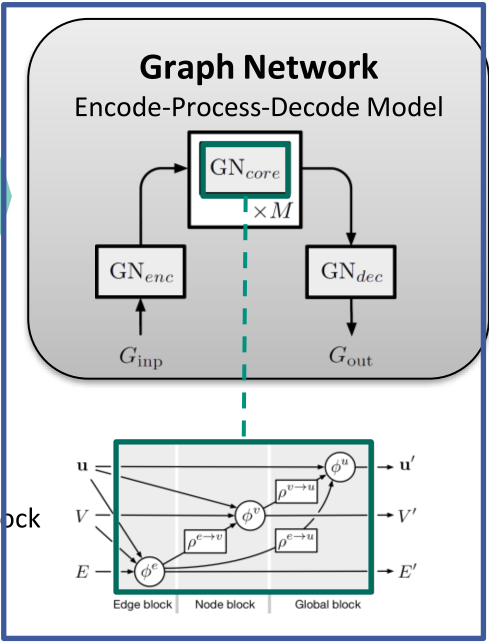
Bimanual Action Recognition: Overview

```
Contact(HandR, Knife)
Contact(HandL, Board)
...
Above(Knife, Board)
Above(Banana, Board)
Above(Board, Bowl)
...
FixedMovingTogether(HandR,
Knife)
...
```

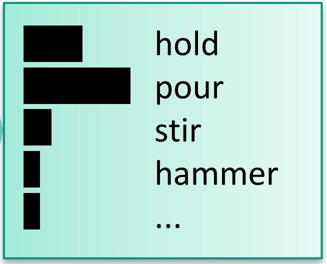
Spatial Relations as List of Predicates



Spatial Relations as Scene Graph



GN Block



Distribution of ***Action Probabilities***

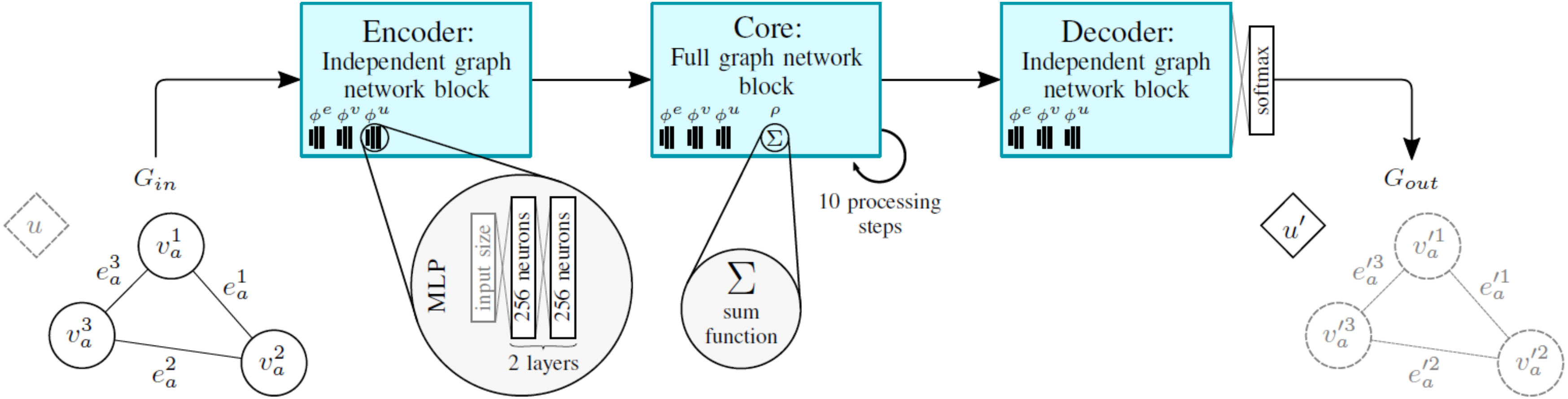
Dreher, C. R. G., Wächter, M. and Asfour, T., *Learning Object-Action Relations from Bimanual Human Demonstration Using Graph Networks*, RA-L (2020)

Bimanual Action Recognition: GN Architecture

Karlsruher Institut für Technologie

Standard Encode-Process-Decode architecture.

- All **update functions** Φ^e, Φ^v, Φ^u:
  Same network architecture (MLP with two layers of 256 neurons) ⟵ **These 9 MLPs are trained.**
- All aggregation functions $\rho^{e\rightarrow v}, \rho^{e\rightarrow u}, \rho^{v\rightarrow u}$: Sum
- 10 processing steps



H2T

Bimanual Action Recognition: Evaluation Data

KIT Bimanual Actions Dataset

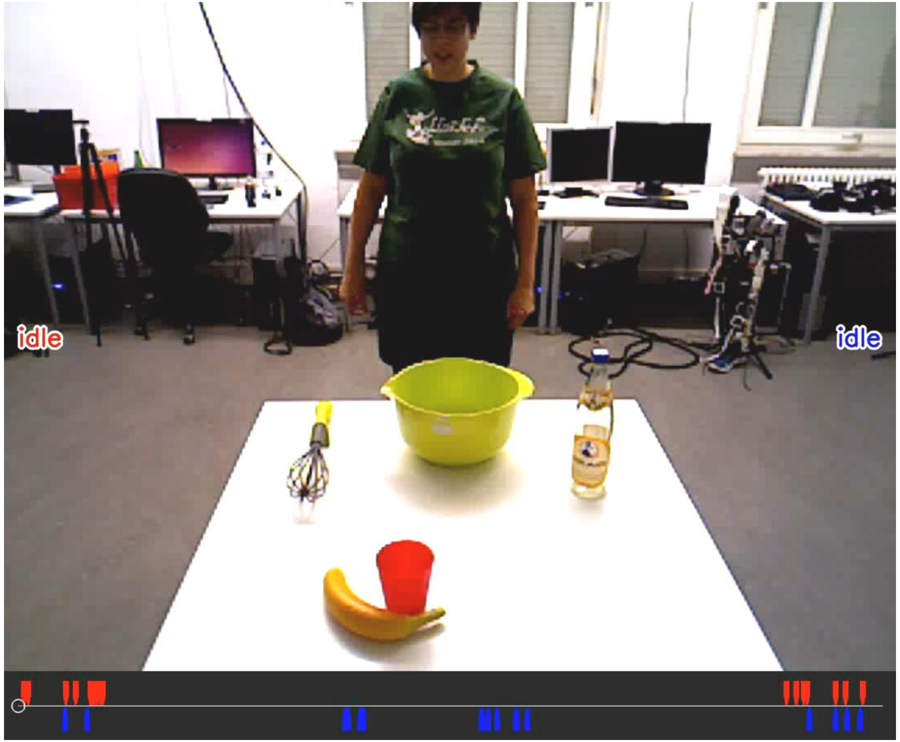
- RGB-D videos showing subjects perform bimanual actions in a kitchen or workshop context
- 6 subjects × 9 tasks × 10 repetitions = 540 recordings
- Manual annotations of performed action by each hand for each video frame.
- **First RGB-D dataset** for bimanual action recognition considering performed actions **of both hands individually**.

Available online at: bimanual-actions.humanoids.kit.edu

Bimanual Action Recognition: Qualitative Results

- Classify action performed by each hand in each frame.
- Visualize top candidate per hand
- Consecutive predictions of the same action class result in an action segment.

Right hand
Left hand

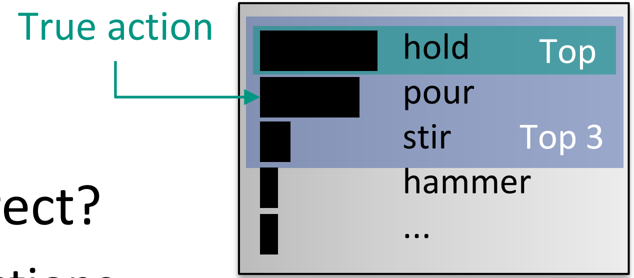


Bimanual Action Recognition: Quantitative Evaluation

- **Problem:** Very noisy object bounding boxes (resulting from noisy depth data)
  - wrong object geometry ⇒ wrong spatial relations (especially `contact`)
  - Was the network generally able to recognize the correct action?
- **Option 1:** Is the top predicted action class correct?
  - Strict to single top candidate.
  - Discards second-best prediction even if probability is high.

⇒ **Option 2:** Is one of the 3 top candidates correct?
  - Also considers second- and third-best predictions.
  - Remember: We estimate probabilities for all 14 action classes.

Action classification
(for 1 hand in 1 frame)

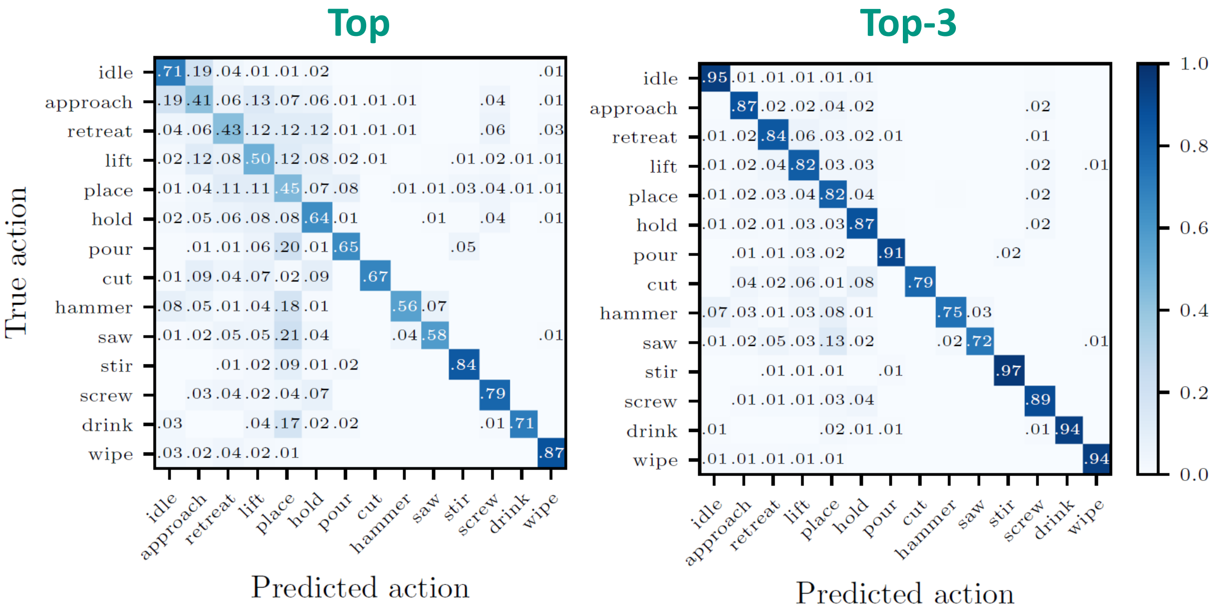


Bimanual Action Recognition: Quantitative Results (I)

- Leave-one-subject-out cross-validation on manually labelled dataset
- F_1 score of action classification (mean over 6 folds resulting from 6 subjects)

	F_1
Top	0.63
Top 3	0.86

- Confusion matrices: Predicted vs. true action classes



Bimanual Action Recognition: Discussion

Major confusions:

- *Place* instead of *saw*, *pour*, *drink*, … .

⇒ No relation to table and orientation considered.

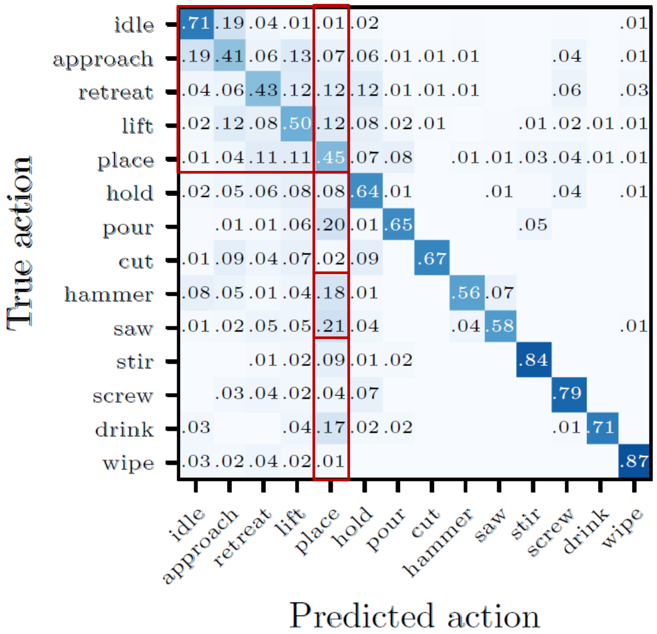
- *Idle*, *approach*, *retreat*, *lift*, *place*

⇒ Require correct dynamic relations, which are prone to noise

- *Hammer* vs. *saw*

⇒ Thin objects ⇒ 3D bounding box extraction from depth image not reliable in such cases.

Top



Bimanual Action Recognition: Discussion

Major confusions:

- *Place* instead of *saw*, *pour*, *drink*, … .

⇒ No relation to table and orientation considered.

- *Idle*, *approach*, *retreat*, *lift*, *place*

⇒ Require correct dynamic relations, which are prone to noise

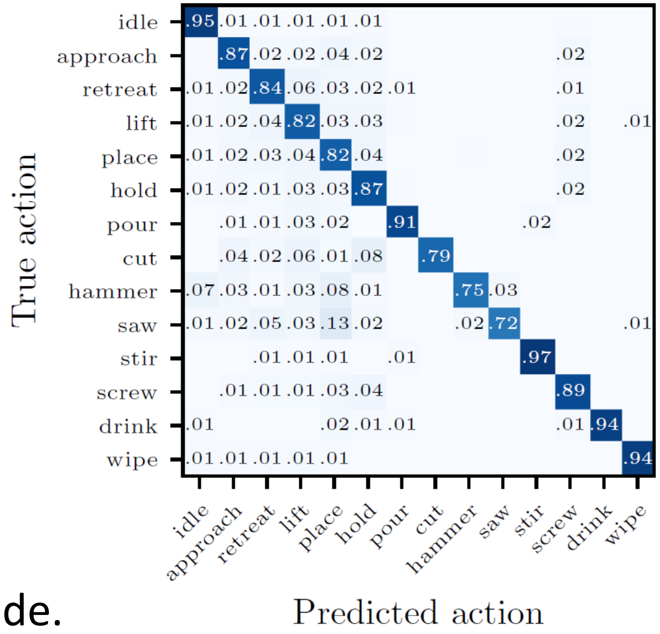
- *Hammer* vs *saw*

⇒ Thin objects ⇒ 3D bounding box extraction from depth image not reliable in such cases.

Top-3 evaluation

- Similar effects observable, although smaller magnitude.

Top-3



Overview

- Introduction
- Scene Representations
- Machine Learning for Object Relations
- **Leveraging Object Relations**
  - Bimanual Action Recognition
  - **Placing Objects Based on Verbal Commands**
  - Avoiding Side-Effects in Bimanual Manipulation

Placing Objects Based on Verbal Commands: Goal

Given:

- Verbal command specifying the spatial relation between two objects.

Goal:

- Place an object according to that spatial relation.

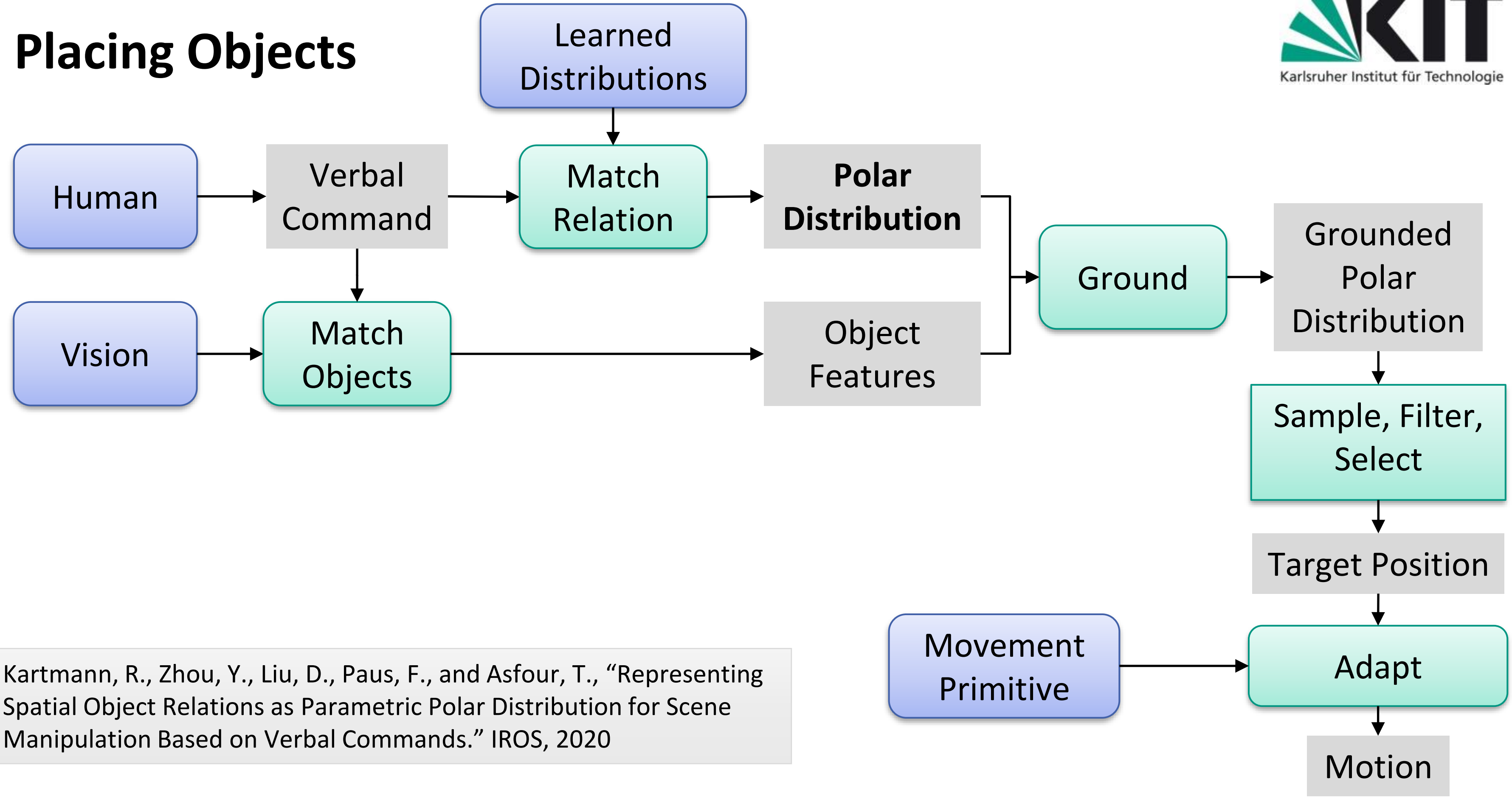
Idea:

- ⇒ Find suitable placing position using learned **polar distributions**.
- ⇒ Adapt movement primitive to move object to placing position.

Put the apple tea ***in front of*** *the corny.*

Let the apple tea be ***on the other side*** *of the corny.*

Placing Objects

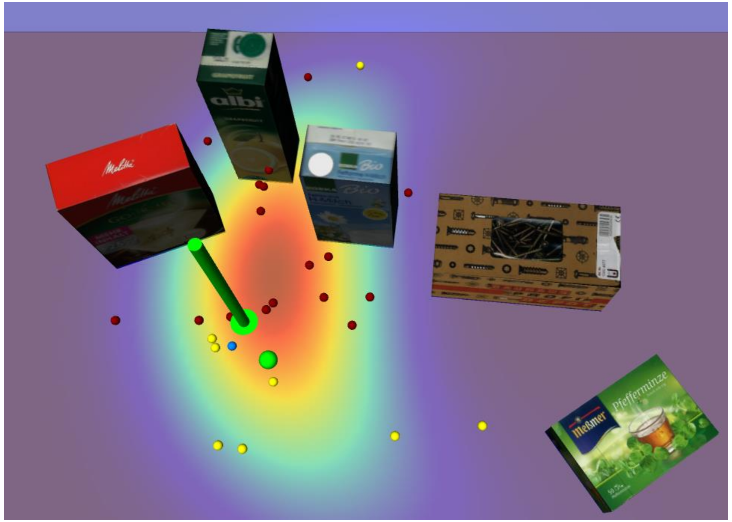


Kartmann, R., Zhou, Y., Liu, D., Paus, F., and Asfour, T., “Representing Spatial Object Relations as Parametric Polar Distribution for Scene Manipulation Based on Verbal Commands.” IROS, 2020

Placing Objects: Sample, Filter, Select

Goal: Adapt target position of movement primitive (MP).

- Sample N candidate target positions.
- **Discard infeasible** candidates.
  - Collision with other objects
  - Off the table
- Get candidates with **top 10% PDF value**.
- Pick candidate **closest to target object's current position**.



target object

Placing Objects: Execution (1)

Placing Objects: Execution (2)

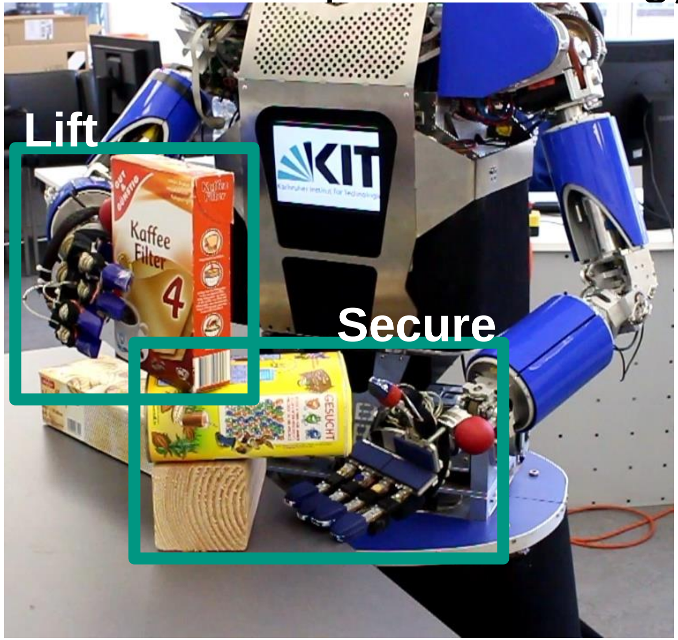
Overview

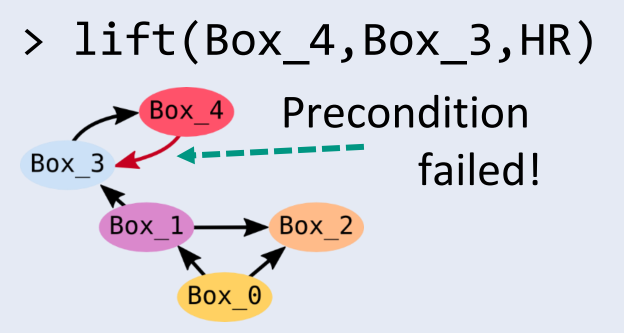
- Introduction
- Scene Representations
- Machine Learning for Object Relations
- **Leveraging Object Relations**
  - Bimanual Action Recognition
  - Placing Objects according to Verbal Commands
  - **Support Relations for Safe Bimanual Manipulation**

Avoiding Side-Effects in Bimanual Manipulation

Top-down support detected

⇒ Use safer bimanual manipulation strategy





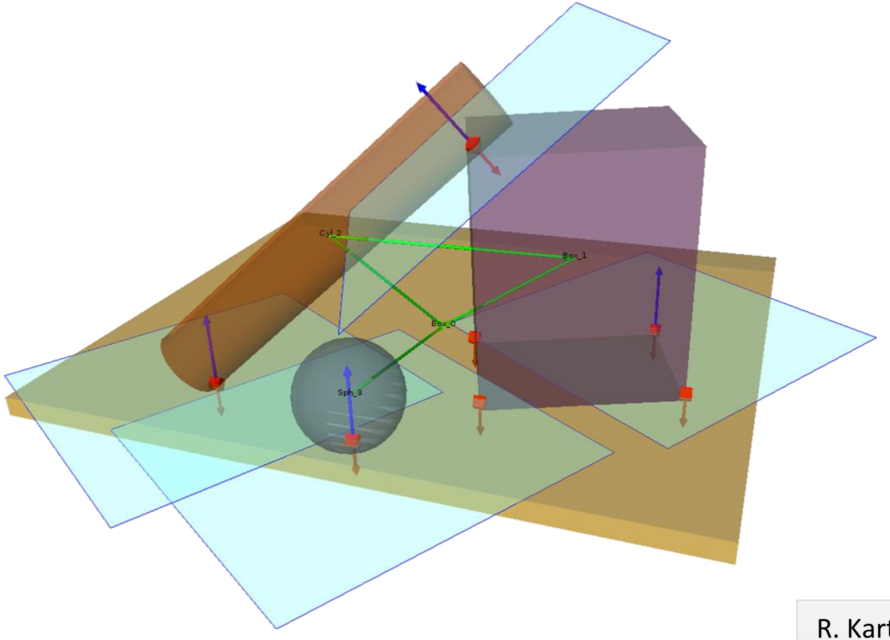
Extracting Support Relations through Force Analysis

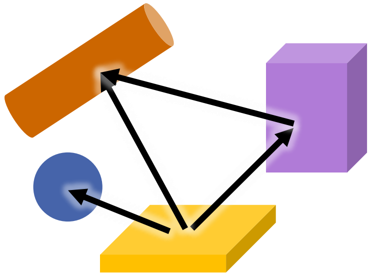
Contacts

Separating Planes

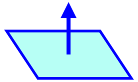
Acting Force

Support Relation





 Contact with Normal

 Separating Plane with Normal

R. Kartmann, F. Paus, M. Grotz and T. Asfour, "Extraction of Physically Plausible Support Relations to Predict and Validate Manipulation Action Effects," RA-L (2018)

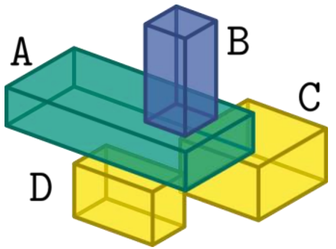
Support Polygon Analysis

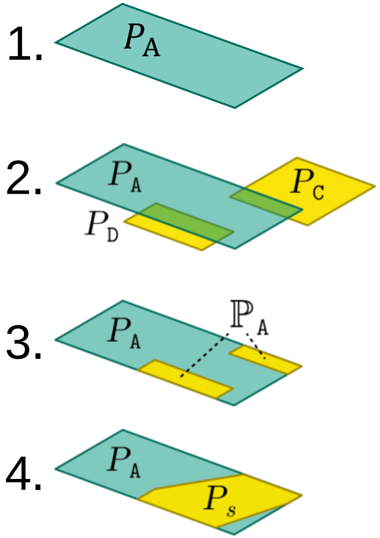
For each support relation $\mathrm{SUPP}(A, B)$:

1. Project A to the ground plane ➔ 2D polygon P_A
2. For each object K where $\mathrm{SUPP}(K, A)$:

   2.1 Project K to the ground plane

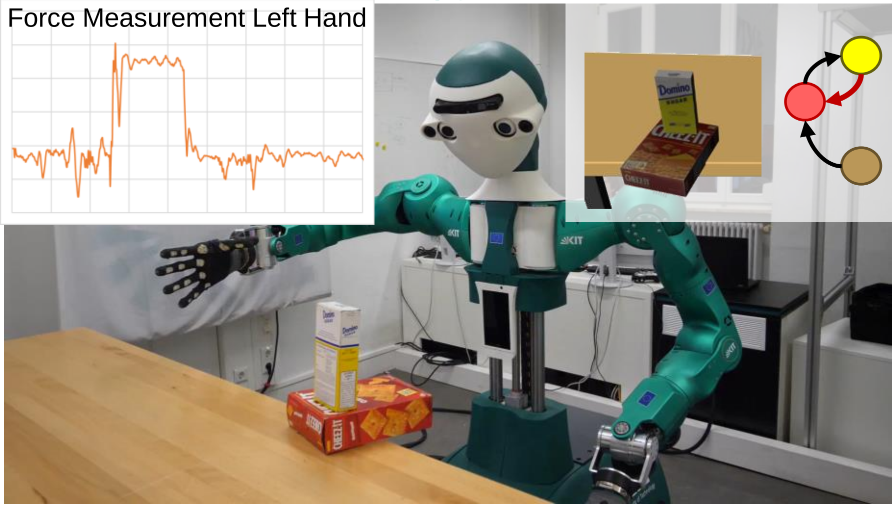
   2.2 Construct intersection with P_A
3. Build set of polygons: $\mathbb{P}_\mathrm{A} = \{P_\mathrm{K} \cap P_\mathrm{A} | \mathrm{SUPP}(K, A)\}$
4. Construct support polygon P_s from $\mathbb{P}_\mathrm{A}$
5. Compute support area ratio $r_s = \dfrac{\mathrm{area}(P_s)}{\mathrm{area}(P_A)}$
6. Assume $\mathrm{SUPP}(B, A)$ if $r_s < r_{s,\min}$





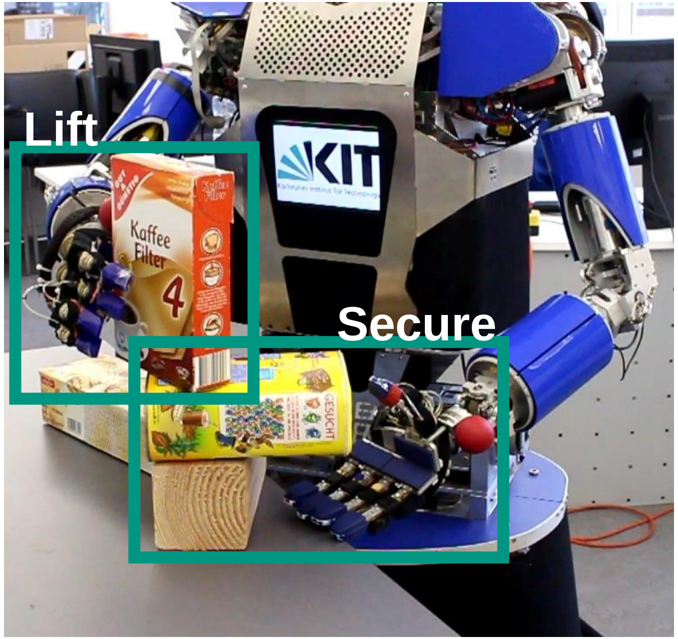
Interactive Exploration of Support Relations

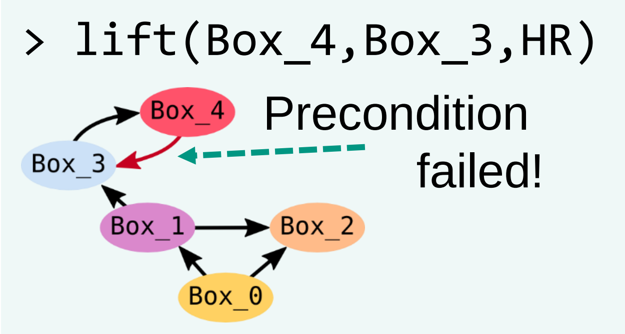
- Top-down support relations depend on **physical properties** of the involved objects (e.g., mass distribution and friction coefficients)
- Interact with the scene to determine top-down support
  ➔ **Bimanual manipulation strategy**



Avoiding Side-Effects in Bimanual Manipulation

Top-down support detected ⇒ Use safer bimanual manipulation strategy





Safe Bimanual Manipulation Strategy

Support Relations: Experiments on ARMAR-6

Overview

- Introduction
- Scene Representations
- Machine Learning for Object Relations
- Leveraging Object Relations