Stereo Vision Approaches for Human to Robot Handover

Aleksej Logacjov

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

3. December 2018

Outline

1. Motivation
2. Basics
3. Stereo Correspondence Algorithms
4. Improvements for Human-to-Robot Handover
5. Future Work

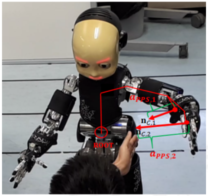
Why do we need Stereo Vision?

[Geiger, A. 2012]

More human like than TOF or phase shift

Why do we need Stereo Vision?

[Geiger, A. 2012]

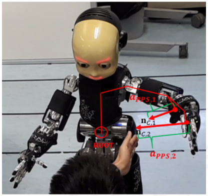


[Nguyen, P. D., et al. 2018]

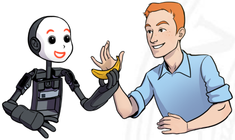
More human like than TOF or phase shift

Why do we need Stereo Vision?

[Geiger, A. 2012]



[Nguyen, P. D., et al. 2018]



Human to Robot Handover NICO robot [WTM]

More human like than TOF or phase shift

Simple Human to Robot Handover

3 main steps:

Simple Human to Robot Handover

3 main steps:

1. Detection of the position (x,y,z)

Simple Human to Robot Handover

3 main steps:

1. Detection of the position (x,y,z)

2. Move robot arm towards (under) the object

Simple Human to Robot Handover

3 main steps:

1. Detection of the position (x,y,z)

2. Move robot arm towards (under) the object

3. Detect the moment when object is in hand

Simple Human to Robot Handover

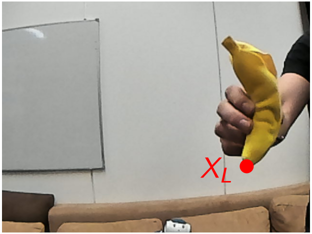
3 main steps:

1. Detection of the position (x,y,z)

Human to robot handover (cont.)

Left image

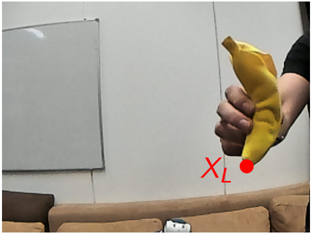
Human to robot handover (cont.)



Left image

- x,y coordinates easy with Object detection and tracking (later)

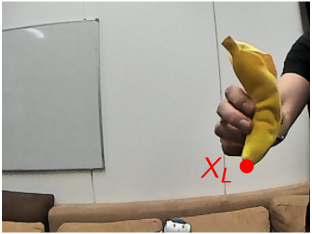
Human to robot handover (cont.)



Left image

- x,y coordinates easy with Object detection and tracking (later)
- But how to get z?

Human to robot handover (cont.)

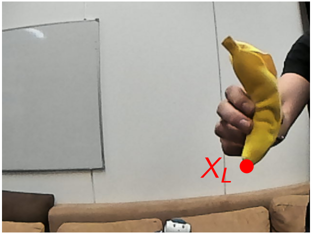


Left image

Right image

- x,y coordinates easy with Object detection and tracking (later)
- But how to get z?

Human to robot handover (cont.)

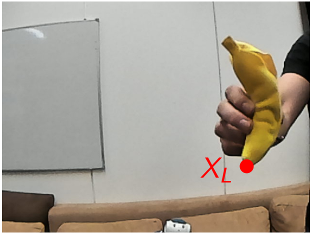


Left image

Right image

- x,y coordinates easy with Object detection and tracking (later)
- But how to get z?
- Displacement $d_P = X_L - X_R$

Human to robot handover (cont.)



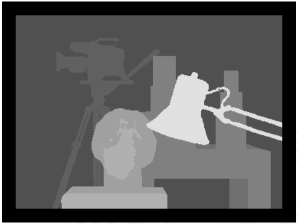
Left image

Right image

- x,y coordinates easy with Object detection and tracking (later)
- But how to get z?
- Displacement $d_P = X_L - X_R$
- Disparity

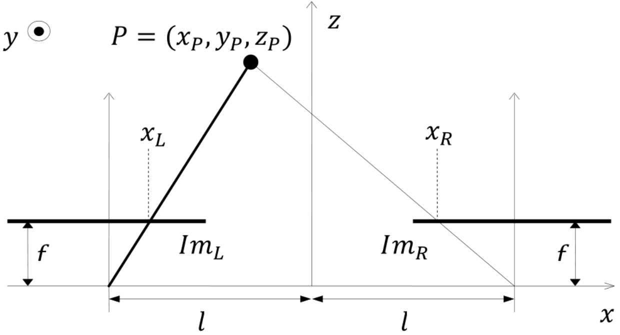
Disparity Map

Original Image [Middlebury Dataset]



Disparity Map [Middlebury Dataset]

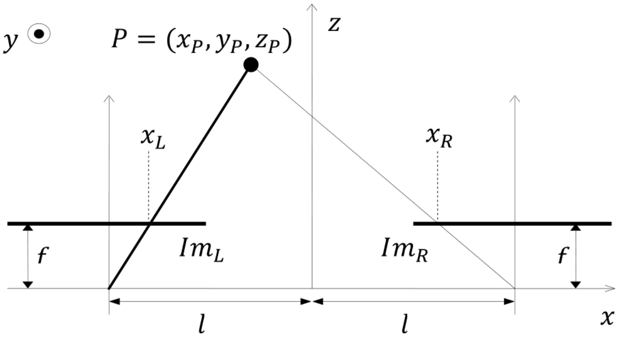
Depth Calculation



[Olofsson, A. 2010]

- Depth can be calculated by $z_P = \frac{T \cdot f}{d_P}$
- $T = 2l$ and $d_P = X_L - X_R$

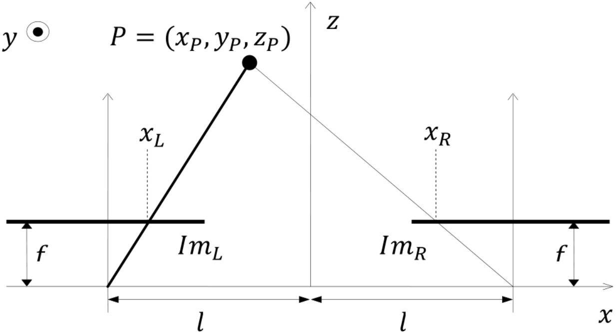
Depth Calculation



[Olofsson, A. 2010]

- Depth can be calculated by $z_P = \frac{T \cdot f}{d_P}$
- $T = 2l$ and $d_P = X_L - X_R$

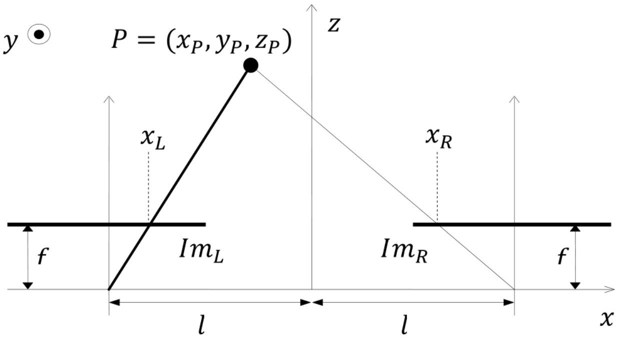
Disparity Map



[Olofsson, A. 2010]

- Problem: how do we know that X_L and X_R correspond to same Point P?
- **Solution: Stereo correspondence algorithms (SC)**

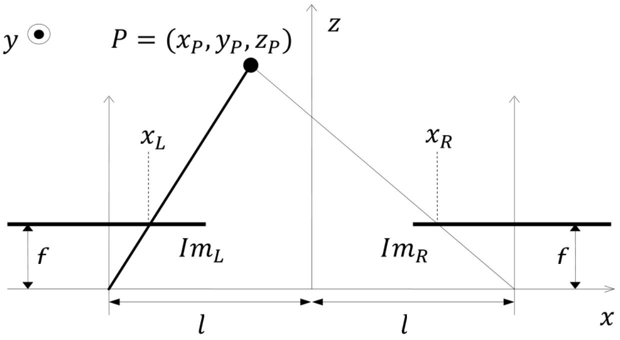
Disparity Map



[Olofsson, A. 2010]

- Problem: how do we know that X_L and X_R correspond to same Point P?
- **Solution: Stereo correspondence algorithms (SC)**

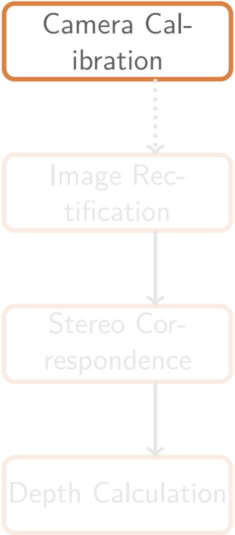
Disparity Map



[Olofsson, A. 2010]

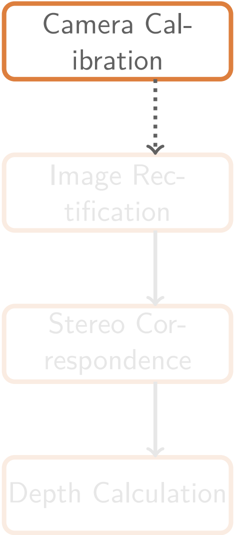
- Problem: how do we know that X_L and X_R correspond to same Point P?
- **Solution: Stereo correspondence algorithms (SC)**

Basics: Camera Calibration



- Intrinsic and extrinsic parameters known
- Has to be done ones

Basics: Camera Calibration



- Intrinsic and extrinsic parameters known
- Has to be done ones

Basics: Image Rectification

Camera Calibration

Image Rectification

Stereo Correspondence

Depth Calculation

- If cameras are not perfectly aligned
- Same point on 2 images are at the epipolar line [Kuhl, A., 2005]
- Making these parallel to baseline
- Reduce complexity

Basics: Image Rectification

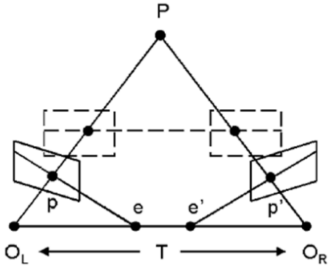
Camera Calibration

Image Rectification

Stereo Correspondence

Depth Calculation

- If cameras are not perfectly aligned
- Same point on 2 images are at the epipolar line [Kuhl, A., 2005]
- Making these parallel to baseline
- Reduce complexity



[Olofsson, A. 2010]

Basics: Image Rectification

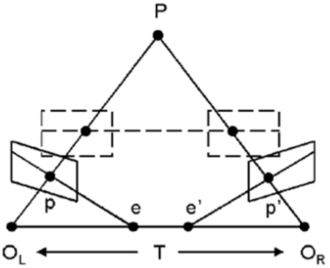
Camera Calibration

Image Rectification

Stereo Correspondence

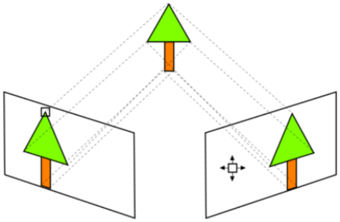
Depth Calculation

- If cameras are not perfectly aligned
- Same point on 2 images are at the epipolar line [Kuhl, A., 2005]
- Making these parallel to baseline
- Reduce complexity



[Olofsson, A. 2010]

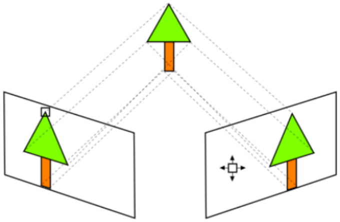
Basics: Image Rectification



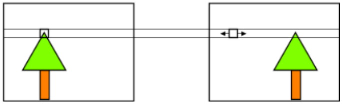
[Olofsson, A. 2010]

- Transform 2D search in 1D
- Linear time complexity [Kuhl, A., 2005]
- Has to be done for each image pair

Basics: Image Rectification



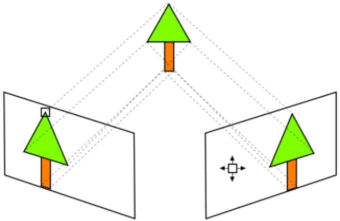
[Olofsson, A. 2010]



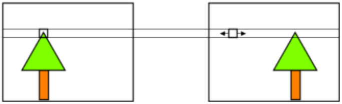
[Olofsson, A. 2010]

- Transform 2D search in 1D
- Linear time complexity [Kuhl, A., 2005]
- Has to be done for each image pair

Basics: Image Rectification



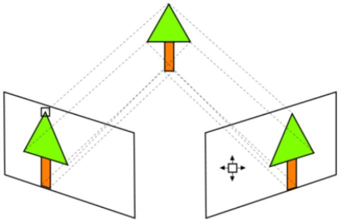
[Olofsson, A. 2010]

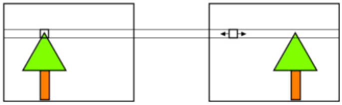


[Olofsson, A. 2010]

- Transform 2D search in 1D
- Linear time complexity [Kuhl, A., 2005]
- Has to be done for each image pair

Basics: Image Rectification





[Olofsson, A. 2010]

[Olofsson, A. 2010]

- Transform 2D search in 1D
- Linear time complexity [Kuhl, A., 2005]
- Has to be done for each image pair

Basics: Image Rectification

Left image

Right image

Basics: Stereo Correspondence

Camera Calibration

↓

Image Rectification

↓

Stereo Correspondence

↓

Depth Calculation

- Find matching pixels in both images
- Calc. disparity $d = X_L - X_R$ f.e. pixel
- Problems: Occlusion, sensor noise ... [Olofsson, A. 2010]
- Still open research [Luo, W. et al., 2016]

Basics: Stereo Correspondence

Camera Calibration

↓

Image Rectification

↓

Stereo Correspondence

↓

Depth Calculation

- Find matching pixels in both images
- Calc. disparity $d = X_L - X_R$ f.e. pixel
- Problems: Occlusion, sensor noise ... [Olofsson, A. 2010]
- Still open research [Luo, W. et al., 2016]

Basics: Stereo Correspondence

Camera Calibration

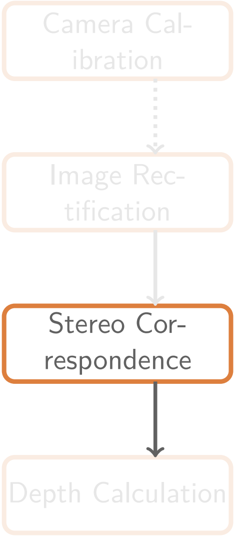
Image Rectification

Stereo Correspondence

Depth Calculation

- Find matching pixels in both images
- Calc. disparity $d = X_L - X_R$ f.e. pixel
- Problems: Occlusion, sensor noise ... [Olofsson, A. 2010]
- Still open research [Luo, W. et al., 2016]

Basics: Stereo Correspondence



- Find matching pixels in both images
- Calc. disparity $d = X_L - X_R$ f.e. pixel
- Problems: Occlusion, sensor noise … [Olofsson, A. 2010]
- Still open research [Luo, W. et al., 2016]

Basics: Depth Calculation

Camera Calibration

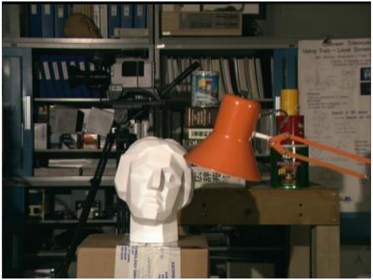
Image Rectification

Stereo Correspondence

Depth Calculation

- F.e. pixel in disparity map: calc. distance $z = \frac{T \cdot f}{d}$
- Straight forward approach with linear complexity

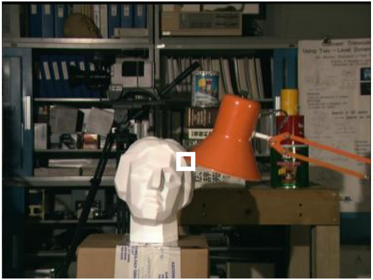
Basics: Stereo Correspondence

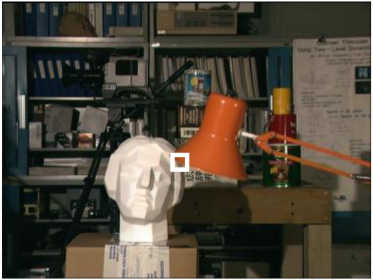


[Middleburry Dataset]

- **Find corresponding pixels**

Basics: Stereo Correspondence

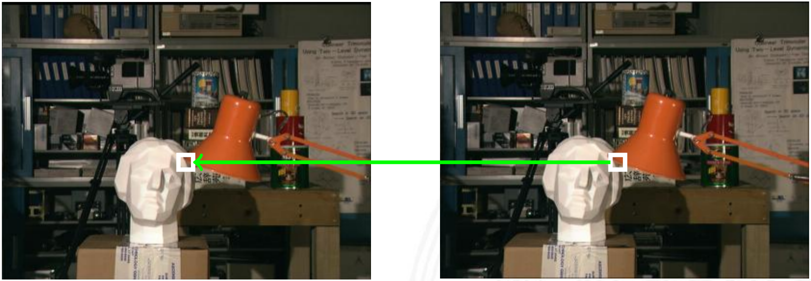




[Middleburry Dataset]

- **Find corresponding pixels**

Basics: Stereo Correspondence



[Middleburry Dataset]

- **Find corresponding pixels**

Basics: Stereo Correspondence

2 basic approaches:

- local

- global

Basics: Stereo Correspondence

2 basic approaches:

- local
- global

Basics: Stereo Correspondence

2 basic approaches:

- local
  - Window based
  - Fast, simple, sensitive to noise and occlusion [Olofsson, A. 2010]
- global

Basics: Stereo Correspondence

2 basic approaches:

- local
  - Window based
  - Fast, simple, sensitive to noise and occlusion [Olofsson, A. 2010]
- global
  - For whole image at once
  - Better in noise/occlusion handling [Olofsson, A. 2010]

Basics: Stereo Correspondence

2 basic approaches:

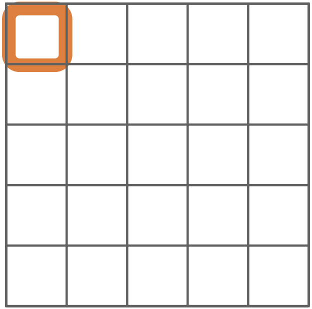
- local
  - Window based
  - Fast, simple, sensitive to noise and occlusion [Olofsson, A. 2010]
- global
  - For whole image at once
  - Better in noise/occlusion handling [Olofsson, A. 2010]

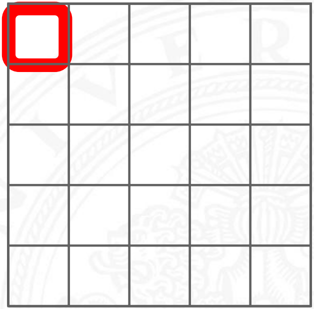
Focus on local

Basics: Stereo Correspondence

- Cost is needed
- F.e. pixel, compute cost accord. to each possible disparity

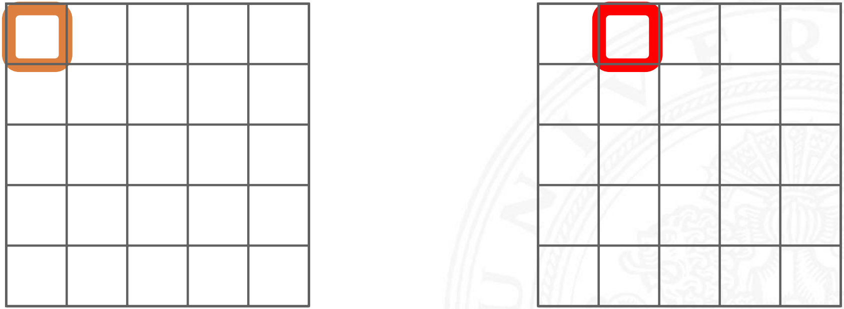
$C(x_1, y_1, 0)$





- Cost is needed
- F.e. pixel, compute cost accord. to each possible disparity

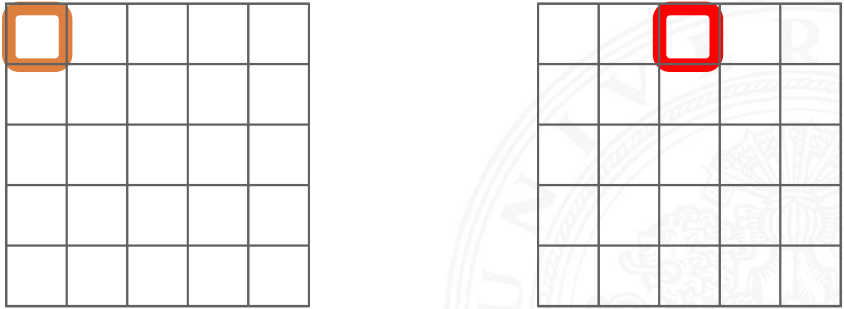
$C(x_1, y_1, 0), C(x_1, y_1, 1)$



Basics: Stereo Correspondence

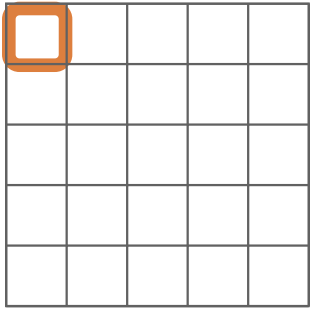
- Cost is needed
- F.e. pixel, compute cost accord. to each possible disparity

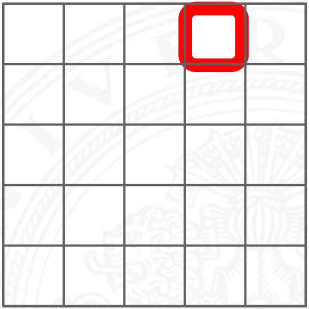
$C(x_1, y_1, 0), C(x_1, y_1, 1), C(x_1, y_1, 2)$



- Cost is needed
- F.e. pixel, compute cost accord. to each possible disparity

$C(x_1, y_1, 0), C(x_1, y_1, 1), C(x_1, y_1, 2), C(x_1, y_1, 3)$

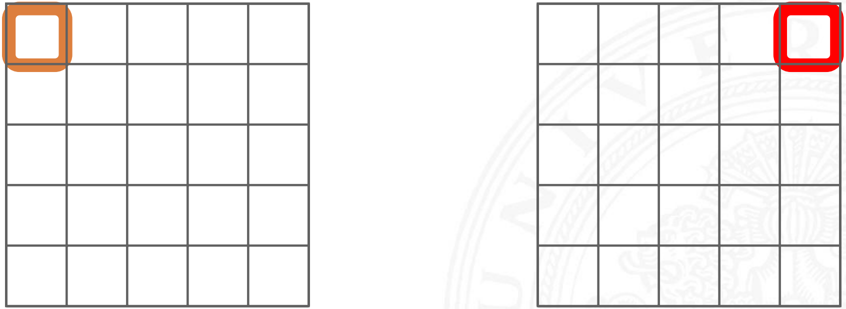




Basics: Stereo Correspondence

- Cost is needed
- F.e. pixel, compute cost accord. to each possible disparity

$$C(x_1, y_1, 0), C(x_1, y_1, 1), C(x_1, y_1, 2), C(x_1, y_1, 3), C(x_1, y_1, 4)$$



Basics: Stereo Correspondence

- Cost is needed
- F.e. pixel, compute cost accord. to each possible disparity

$$C(x_1, y_1, 0), C(x_1, y_1, 1), C(x_1, y_1, 2), C(x_1, y_1, 3), C(x_1, y_1, 4)$$
$$C(x_1, y_2, -1), C(x_1, y_2, 0), C(x_1, y_2, 1), C(x_1, y_2, 2), C(x_1, y_2, 3)$$

Basics: Stereo Correspondence

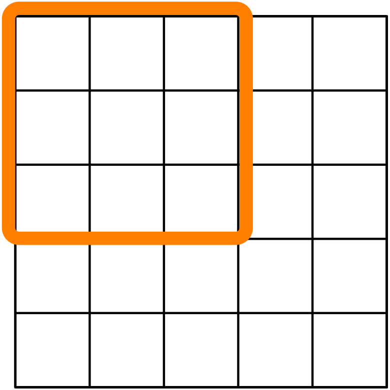
- Cost is needed
- F.e. pixel, compute cost accord. to each possible disparity

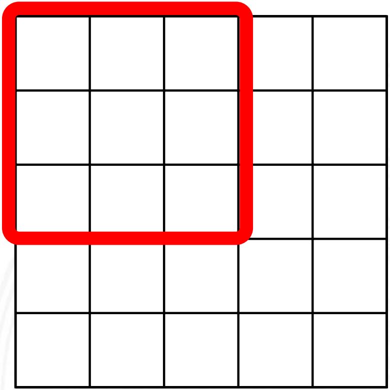
$C(x_1, y_1, 0), C(x_1, y_1, 1), C(x_1, y_1, 2), C(x_1, y_1, 3), C(x_1, y_1, 4)$
$C(x_1, y_2, -1), C(x_1, y_2, 0), C(x_1, y_2, 1), C(x_1, y_2, 2), C(x_1, y_2, 3)$

- **Cost Computation**

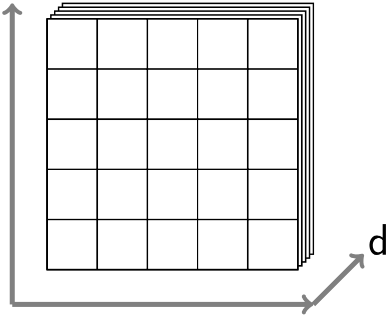
Basics: Stereo Correspondence

- Window centered at pixel
- Taking neighbors into account



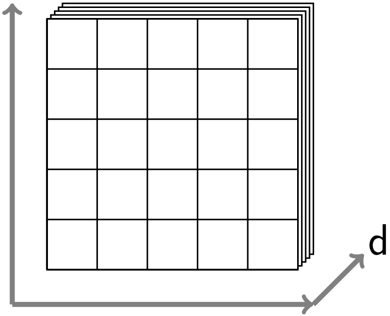


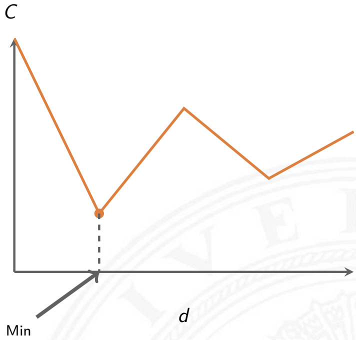
Basics: Stereo Correspondence



- Disparity Space Image (DSI)

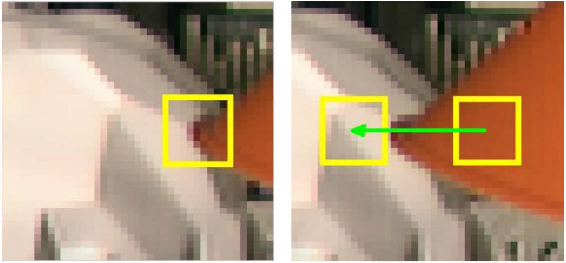
Basics: Stereo Correspondence

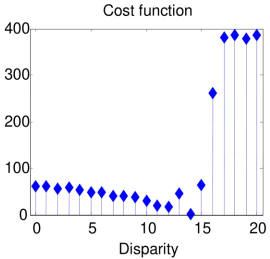




- Disparity Space Image (DSI)
- For each pixel

- Example window-based cost computation





[Olofsson, A. 2010]

3 different approaches to compute the **matching cost**:

- (Sum of) Absolute Intensity Difference $C_{SAD}(x, y, d)$
- Deep Learning Approach [Luo, W. et al., 2016]
- Efficient Large-Scale-Stereo-Matching (ELAS) [Geiger, A. 2012]

(Sum of) Absolute Intensity Difference

- Absolute Intensity Difference (AD):
  $C_{AD}(x, y, d) = |I_L(x, y) - I_R(x - d, y)|$

- (Sum of) = Window-based
- Sum of AD:
  $C_{SAD}(x, y, d) = \Sigma_{(u,v)\in N(x,y)} |I_L(u, v) - I_R(u - d, v)|$

- With Neighborhood N(x,y) of (x,y)

(Sum of) Absolute Intensity Difference

- Absolute Intensity Difference (AD):
  $C_{AD}(x, y, d) = |I_L(x, y) - I_R(x - d, y)|$

- (Sum of) = Window-based
- Sum of AD:
  $C_{SAD}(x, y, d) = \Sigma_{(u,v) \in N(x,y)} |I_L(u, v) - I_R(u - d, v)|$

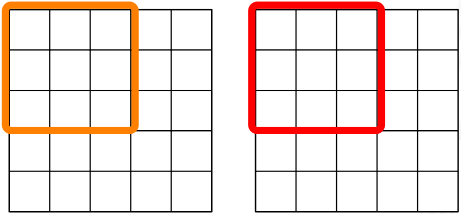
- With Neighborhood N(x,y) of (x,y)

(Sum of) Absolute Intensity Difference

- Absolute Intensity Difference (AD):
  $C_{AD}(x, y, d) = |I_L(x, y) - I_R(x - d, y)|$

- (Sum of) = Window-based
- Sum of AD:
  $C_{SAD}(x, y, d) = \Sigma_{(u,v)\in N(x,y)}|I_L(u, v) - I_R(u - d, v)|$

- With Neighborhood N(x,y) of (x,y)



(Sum of) Absolute Intensity Difference

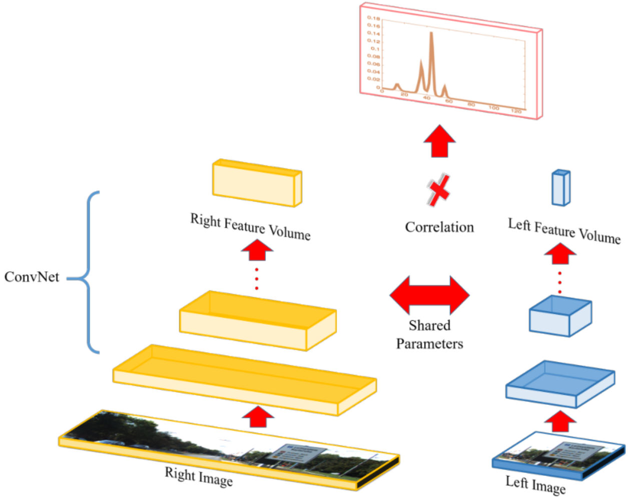
- Pros: Fast, simple. According to [Scharstein, D. et al., 2002] one of the fastest classical approach.
- Cons: Bad accuracy (place 8 of 20 according to [Scharstein, D. et al., 2002])

(Sum of) Absolute Intensity Difference

- Pros: Fast, simple. According to [Scharstein, D. et al., 2002] one of the fastest classical approach.

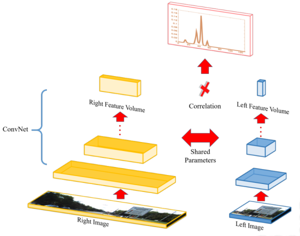
- Cons: Bad accuracy (place 8 of 20 according to [Scharstein, D. et al., 2002])

Deep Learning Approach [Luo, W. et al., 2016]



[Luo, W. et al., 2016]

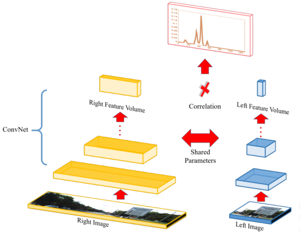
Deep Learning Approach [Luo, W. et al., 2016]



[Luo, W. et al., 2016]

- Put one image patch, centered at pixel (x,y) as input (9x9)
- Put an image patch, of size (max_disparity,9) as sec. input
- Network computes in **one iteration**, the cost for **all** given disparities

Deep Learning Approach [Luo, W. et al., 2016]



[Luo, W. et al., 2016]

- Put one image patch, centered at pixel (x,y) as input (9x9)
- Put an image patch, of size (max_disparity,9) as sec. input
- Network computes in **one iteration**, the cost for **all** given disparities

Deep Learning Approach [Luo, W. et al., 2016]

- Training:
  - Random image patches from the Kitti dataset
  - Cross entropy loss for multi class classification (disparities)
  - 6.5 hours training
- Testing/Benchmarking:
  - On Kitti and Middlebury dataset

Deep Learning Approach [Luo, W. et al., 2016]

- Training:
  - Random image patches from the Kitti dataset
  - Cross entropy loss for multi class classification (disparities)
  - 6.5 hours training

- Testing/Benchmarking:
  - On Kitti and Middlebury dataset

Deep Learning Approach [Luo, W. et al., 2016]

- Pros:
  - Very fast, compared to other learning approaches (1sec on NVIDIA Titan-X)
  - As accurate as other learning approaches

- Cons:
  - No comparison to non learning state-of-the art approaches
  - After calculation, cost aggregation, smoothing done (time consuming)
  - No CPU runtime

Deep Learning Approach [Luo, W. et al., 2016]

- Pros:
  - Very fast, compared to other learning approaches (1sec on NVIDIA Titan-X)
  - As accurate as other learning approaches

- Cons:
  - No comparison to non learning state-of-the art approaches
  - After calculation, cost aggregation, smoothing done (time consuming)
  - No CPU runtime

Efficient Large-Scale-Stereo-Matching (ELAS) [Geiger, A. 2012]

- Not trying to find matching f.e. pixel in first run
- They can be ambiguous
- Find robust matchings with **matching support points** algorithm
- These pixels: **support points**

- Find support points by calc. L1-distance between feature vectors

- Using this support points to calc. the remaining matchings

Efficient Large-Scale-Stereo-Matching (ELAS) [Geiger, A. 2012]

- Not trying to find matching f.e. pixel in first run
- They can be ambiguous
- Find robust matchings with **matching support points** algorithm
- These pixels: **support points**
- Find support points by calc. L1-distance between feature vectors
- Using this support points to calc. the remaining matchings

Efficient Large-Scale-Stereo-Matching (ELAS) [Geiger, A. 2012]

- Not trying to find matching f.e. pixel in first run
- They can be ambiguous
- Find robust matchings with **matching support points** algorithm
- These pixels: **support points**

- Find support points by calc. L1-distance between feature vectors

- Using this support points to calc. the remaining matchings

Efficient Large-Scale-Stereo-Matching (ELAS) [Geiger, A. 2012]

- Pros:
  - Very fast, (0.7 sec on i7 CPU with 2.66 GHz)
  - Performs well on higher resolution images (900x750)
  - Better accuracy than other state-of-the-art approaches

- Cons:
  - Non trivial algorithm
  - 0.7 sec maybe to slow for human-robot-handover

Efficient Large-Scale-Stereo-Matching (ELAS) [Geiger, A. 2012]

iCub using ELAS [Nguyen, P. D., et al. 2018]

Improvements for Human-to-Robot Handover

- All 3 approaches can be used in real-time applications
- With powerful hardware (except SAD)
- Maybe this is not given for some humanoid robots

- Focusing on creating a good disparity map for each pixel
- They are also focusing on handling noise (except SAD)

- But we don't need both (at least not so much)
- We **only** need the pixels **corresponding to object**
- We don't need to consider a lot of different disparities
- Only objects which are nearer than approx. 30cm

Improvements for Human-to-Robot Handover

- All 3 approaches can be used in real-time applications
- With powerful hardware (except SAD)
- Maybe this is not given for some humanoid robots

- Focusing on creating a good disparity map for each pixel
- They are also focusing on handling noise (except SAD)

- But we don't need both (at least not so much)
- We **only** need the pixels **corresponding to object**
- We don't need to consider a lot of different disparities
- Only objects which are nearer than approx. 30cm

Improvements for Human-to-Robot Handover

- All 3 approaches can be used in real-time applications
- With powerful hardware (except SAD)
- Maybe this is not given for some humanoid robots

- Focusing on creating a good disparity map for each pixel
- They are also focusing on handling noise (except SAD)

- But we don't need both (at least not so much)
- We **only** need the pixels **corresponding to object**
- We don't need to consider a lot of different disparities
- Only objects which are nearer than approx. 30cm

Improvements for Human-to-Robot Handover

We need 3 things:

1. Object detection and tracking (create **bounding box**)

2. Cutting out the bounding box

3. Use $z = \frac{T \cdot f}{d}$ to calc. disparity boundaries

Improvements for Human-to-Robot Handover

We need 3 things:

1. Object detection and tracking (create **bounding box**)

2. Cutting out the bounding box

3. Use $z = \frac{T \cdot f}{d}$ to calc. disparity boundaries

Object Detection and Object Tracking

- Object Detection (YOLOv3)[Redmon, J., 2018]
  - Has to be done once at the beginning
  - After that, tracking
- Object Tracking (CSRT)[OpenCV]
  - Fast and accurate tracking

NICO example

Object Detection and Object Tracking

- Object Detection (YOLOv3)[Redmon, J., 2018]
  - Has to be done once at the beginning
  - After that, tracking
- Object Tracking (CSRT)[OpenCV]
  - Fast and accurate tracking

NICO example

Object Detection and Object Tracking

- Object Detection (YOLOv3)[Redmon, J., 2018]
  - Has to be done once at the beginning
  - After that, tracking
- Object Tracking (CSRT)[OpenCV]
  - Fast and accurate tracking

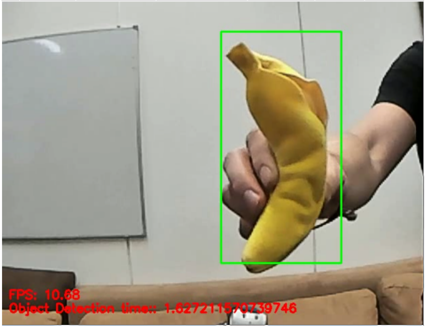
NICO example

Object Detection and Object Tracking

- Object Detection (YOLOv3)[Redmon, J., 2018]
  - Has to be done once at the beginning
  - After that, tracking
- Object Tracking (CSRT)[OpenCV]
  - Fast and accurate tracking

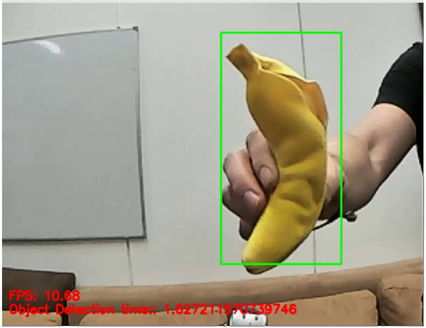
NICO example

Cutting out the bounding box

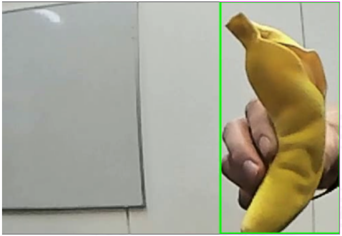


Full view

Cutting out the bounding box

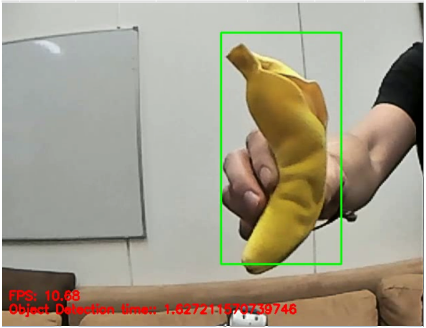


Full view

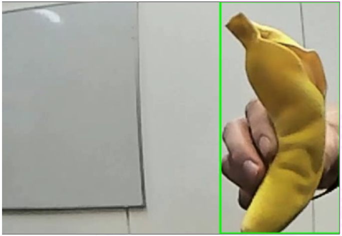


Cut out view

Cutting out the bounding box



Full view



Cut out view

- Detection runtime: approx 2 sec
- x coordinates important for disparity

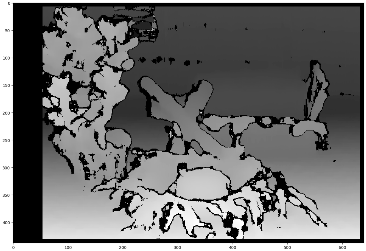
Cutting out the BB (Disparity Comparison)

Original image left [Middlebury Dataset]

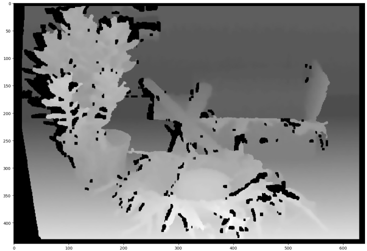
Original image right [Middlebury Dataset]

- size: 640x438

Cutting out the BB (Disparity Comparison)

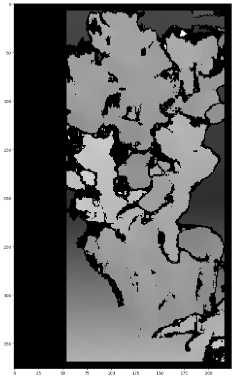


SAD

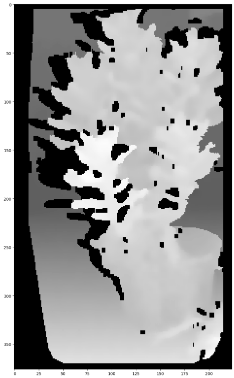


ELAS [Geiger, A. 2012]

Cutting out the BB (Disparity Comparison)



SAD



ELAS [Geiger, A. 2012]

- size: 224x376 (approx 3 times smaller)

Cutting out the BB (Disparity Comparison)

- SAD:
  - Big: 0.014 sec
  - Small: 0.006 sec (2 times faster)
- ELAS:
  - Big: 0.24 sec
  - Small: 0.06 sec (4 times faster)
- Time to cut out: 0.0005 sec

Calc. the disparity boundaries

- $z = \frac{T \cdot f}{d}$
- $d = \frac{T \cdot f}{z}$

- Knowing T, f and max. reachable dist. z_{max}:
- Calc. smallest disparity $d_{min} = \frac{T \cdot f}{z_{max}}$
- Similar to smallest distance z_{min}

- Lead to a smaller DSI

Calc. the disparity boundaries

- $z = \frac{T \cdot f}{d}$
- $d = \frac{T \cdot f}{z}$

- Knowing T, f and max. reachable dist. z_{max}:
- Calc. smallest disparity $d_{min} = \frac{T \cdot f}{z_{max}}$
- Similar to smallest distance z_{min}

- Lead to a smaller DSI

Calc. the disparity boundaries

- $z = \frac{T \cdot f}{d}$
- $d = \frac{T \cdot f}{z}$

- Knowing T, f and max. reachable dist. z_{max}:
- Calc. smallest disparity $d_{min} = \frac{T \cdot f}{z_{max}}$
- Similar to smallest distance z_{min}

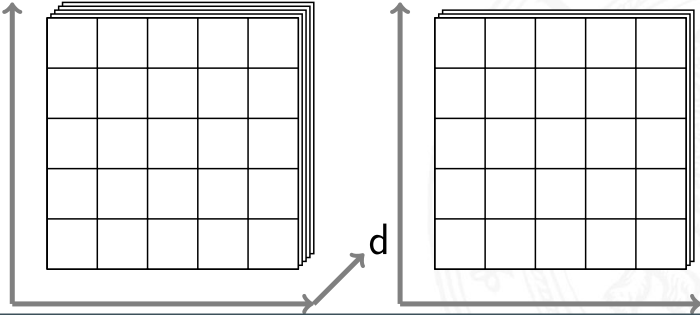
- Lead to a smaller DSI

Calc. the disparity boundaries

- $z = \frac{T \cdot f}{d}$
- $d = \frac{T \cdot f}{z}$

- Knowing T, f and max. reachable dist. z_{max}:
- Calc. smallest disparity $d_{min} = \frac{T \cdot f}{z_{max}}$
- Similar to smallest distance z_{min}

- Lead to a smaller DSI



Improvements for Human-to-Robot Handover

1. (*)Calibrate the cameras
2. **Get both video streams** from the cams
3. **Rectify** the frames
4. (*)At some frame, **detect the object** in **both** frames (YOLOv3 approx. 2 sec)
5. **Track** it (CSRT approx. 0.05 sec)
6. **Cut out** the BB (approx. 0.0005 sec)
7. **Stereo Correspondence** Algorithm (SAD: between 0.006 and 0.014 sec)
8. Calculate the depth of the nearest pixels

(*) = Only 1x

Improvements for Human-to-Robot Handover

1. (*)Calibrate the cameras
2. **Get both video streams** from the cams
3. **Rectify** the frames
4. (*)At some frame, **detect the object** in **both** frames (YOLOv3 approx. 2 sec)
5. **Track** it (CSRT approx. 0.05 sec)
6. **Cut out** the BB (approx. 0.0005 sec)
7. **Stereo Correspondence** Algorithm (SAD: between 0.006 and 0.014 sec)
8. Calculate the depth of the nearest pixels

(*) = Only 1x

Improvements for Human-to-Robot Handover

1. (*)Calibrate the cameras
2. **Get both video streams** from the cams
3. **Rectify** the frames
4. (*)At some frame, **detect the object** in **both** frames (YOLOv3 approx. 2 sec)
5. **Track** it (CSRT approx. 0.05 sec)
6. **Cut out** the BB (approx. 0.0005 sec)
7. **Stereo Correspondence** Algorithm (SAD: between 0.006 and 0.014 sec)
8. Calculate the depth of the nearest pixels

(*) = Only 1x

Improvements for Human-to-Robot Handover

1. (*)Calibrate the cameras
2. **Get both video streams** from the cams
3. **Rectify** the frames
4. (*)At some frame, **detect the object** in **both** frames (YOLOv3 approx. 2 sec)
5. **Track** it (CSRT approx. 0.05 sec)
6. **Cut out** the BB (approx. 0.0005 sec)
7. **Stereo Correspondence** Algorithm (SAD: between 0.006 and 0.014 sec)
8. Calculate the depth of the nearest pixels

(*) = Only 1x

Improvements for Human-to-Robot Handover

1. (*)Calibrate the cameras
2. **Get both video streams** from the cams
3. **Rectify** the frames
4. (*)At some frame, **detect the object** in **both** frames (YOLOv3 approx. 2 sec)
5. **Track** it (CSRT approx. 0.05 sec)
6. **Cut out** the BB (approx. 0.0005 sec)
7. **Stereo Correspondence** Algorithm (SAD: between 0.006 and 0.014 sec)
8. Calculate the depth of the nearest pixels

(*) = Only 1x

Improvements for Human-to-Robot Handover

1. (*)Calibrate the cameras
2. **Get both video streams** from the cams
3. **Rectify** the frames
4. (*)At some frame, **detect the object** in **both** frames (YOLOv3 approx. 2 sec)
5. **Track** it (CSRT approx. 0.05 sec)
6. **Cut out** the BB (approx. 0.0005 sec)
7. **Stereo Correspondence** Algorithm (SAD: between 0.006 and 0.014 sec)
8. Calculate the depth of the nearest pixels

(*) = Only 1x

Improvements for Human-to-Robot Handover

1. (*)Calibrate the cameras
2. **Get both video streams** from the cams
3. **Rectify** the frames
4. (*)At some frame, **detect the object** in **both** frames (YOLOv3 approx. 2 sec)
5. **Track** it (CSRT approx. 0.05 sec)
6. **Cut out** the BB (approx. 0.0005 sec)
7. **Stereo Correspondence** Algorithm (SAD: between 0.006 and 0.014 sec)
8. Calculate the depth of the nearest pixels

(*) = Only 1x

Improvements for Human-to-Robot Handover

1. (*)Calibrate the cameras
2. **Get both video streams** from the cams
3. **Rectify** the frames
4. (*)At some frame, **detect the object** in **both** frames (YOLOv3 approx. 2 sec)
5. **Track** it (CSRT approx. 0.05 sec)
6. **Cut out** the BB (approx. 0.0005 sec)
7. **Stereo Correspondence** Algorithm (SAD: between 0.006 and 0.014 sec)
8. Calculate the depth of the nearest pixels

(*) = Only 1x

Future Work

- Calibrate the cameras of NICO
- Apply different SC algorithms
  - Check for **runtime**
  - Check for **distance accuracy**
  - Check for different resolutions
- Try to implement a handover
  - Detect object in scene
  - Calculate (x,y,z) coordinates of object
  - Implement inverse kinematic to determine motor positions
  - Detect the moment the object is placed in the hand
  - Close the hand

Future Work

- Calibrate the cameras of NICO
- Apply different SC algorithms
  - Check for **runtime**
  - Check for **distance accuracy**
  - Check for different resolutions
- Try to implement a handover
  - Detect object in scene
  - Calculate (x,y,z) coordinates of object
  - Implement inverse kinematic to determine motor positions
  - Detect the moment the object is placed in the hand
  - Close the hand

Future Work

- Calibrate the cameras of NICO
- Apply different SC algorithms
  - Check for **runtime**
  - Check for **distance accuracy**
  - Check for different resolutions
- Try to implement a handover
  - Detect object in scene
  - Calculate (x,y,z) coordinates of object
  - Implement inverse kinematic to determine motor positions
  - Detect the moment the object is placed in the hand
  - Close the hand

Future Work

- Calibrate the cameras of NICO
- Apply different SC algorithms
  - Check for **runtime**
  - Check for **distance accuracy**
  - Check for different resolutions
- Try to implement a handover
  - Detect object in scene
  - Calculate (x,y,z) coordinates of object
  - Implement inverse kinematic to determine motor positions
  - Detect the moment the object is placed in the hand
  - Close the hand

Future Work

- Calibrate the cameras of NICO

- Apply different SC algorithms
  - Check for **runtime**
  - Check for **distance accuracy**
  - Check for different resolutions

- Try to implement a handover
  - Detect object in scene
  - Calculate (x,y,z) coordinates of object
  - Implement inverse kinematic to determine motor positions
  - Detect the moment the object is placed in the hand
  - Close the hand

Future Work

- Calibrate the cameras of NICO

- Apply different SC algorithms
  - Check for **runtime**
  - Check for **distance accuracy**
  - Check for different resolutions

- Try to implement a handover
  - Detect object in scene
  - Calculate (x,y,z) coordinates of object
  - Implement inverse kinematic to determine motor positions
  - Detect the moment the object is placed in the hand
  - Close the hand

Future Work

- Calibrate the cameras of NICO

- Apply different SC algorithms
  - Check for **runtime**
  - Check for **distance accuracy**
  - Check for different resolutions

- Try to implement a handover
  - Detect object in scene
  - Calculate (x,y,z) coordinates of object
  - Implement inverse kinematic to determine motor positions
  - Detect the moment the object is placed in the hand
  - Close the hand

Literature

- Geiger, A., Lenz, P., Urtasun, R. (2012, June). Are we ready for autonomous driving? the kitti vision benchmark suite. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on (pp. 3354-3361). IEEE.
- Nguyen, D. H. P., Hoffmann, M., Roncone, A., Pattacini, U., Metta, G. (2018, February). Compact Real-time avoidance on a Humanoid Robot for Human-robot Interaction. In Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction (pp. 416-424). ACM.
- Olofsson, A. (2010). Modern stereo correspondence algorithms: investigation and evaluation.
- Scharstein, D., Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International journal of computer vision, 47(1-3), 7-42.

Literature

- Fang, J., Varbanescu, A. L., Shen, J., Sips, H., Saygili, G., Van Der Maaten, L. (2012, December). Accelerating cost aggregation for real-time stereo matching. In 2012 IEEE 18th International Conference on Parallel and Distributed Systems (pp. 472-481). IEEE.
- Geiger, A., Roser, M., Urtasun, R. (2010). Efficient large-scale stereo matching. In Computer Vision–ACCV 2010 (pp. 25-38). Springer, Berlin, Heidelberg.
- Luo, W., Schwing, A. G., Urtasun, R. (2016). Efficient deep learning for stereo matching. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5695-5703).
- Redmon, J., Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.
- https://www.inf.uni-hamburg.de/en/inst/ab/wtm/research/neurobotics/nico.html, 30.11.18

Thank you for your attention.
Any Questions?