Skeleton Tracking by the use of energy minimization

How we solve the correspondence problem

Norman Link

nlink@uni-koblenz.de

Institute for Computational Visualistics
Universität Koblenz-Landau

02. Mar. 2012

Motivation

- Analzing skeleton motion
- Tracking infant motions
- Solving the correspondence problem for adult-infant experiments
- independent code base for further research

COMPUTERVISUALISTIK

Motivation

- Analzing skeleton motion
- Tracking infant motions
- Solving the correspondence problem for adult-infant experiments
- independent code base for further research

COMPUTERVISUALISTIK

Motivation

- Analzing skeleton motion
- Tracking infant motions
- Solving the correspondence problem for adult-infant experiments
- independent code base for further research

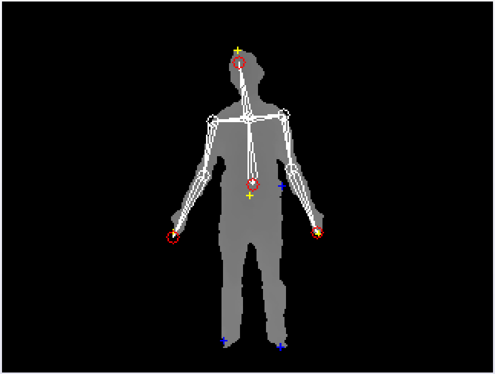
COMPUTERVISUALISTIK

Motivation

- Analzing skeleton motion
- Tracking infant motions
- Solving the correspondence problem for adult-infant experiments
- independent code base for further research

COMPUTERVISUALISTIK

Motivation



Pipeline

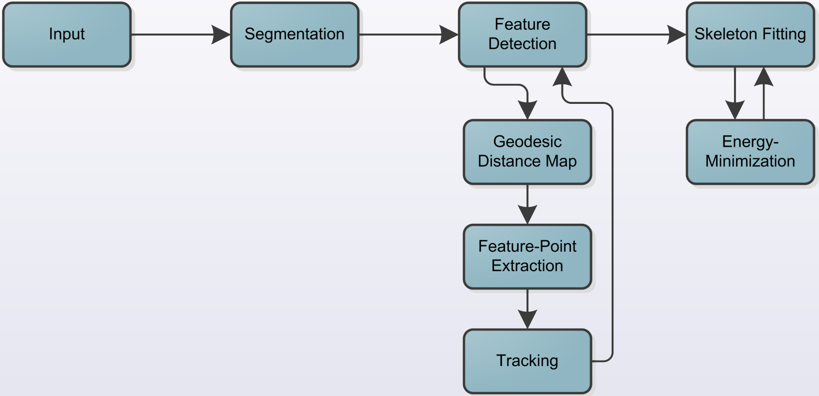


Figure: Pipeline

Pipeline Overview

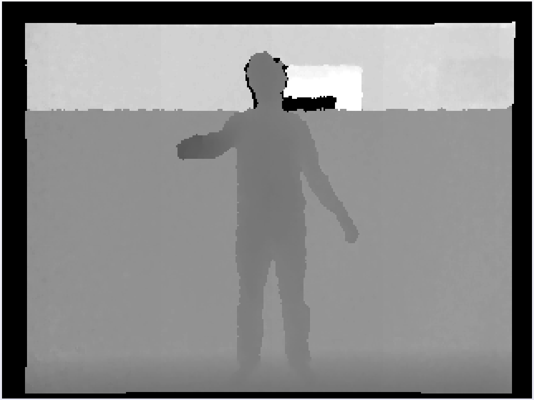


Figure: Depth Map

Pipeline Overview

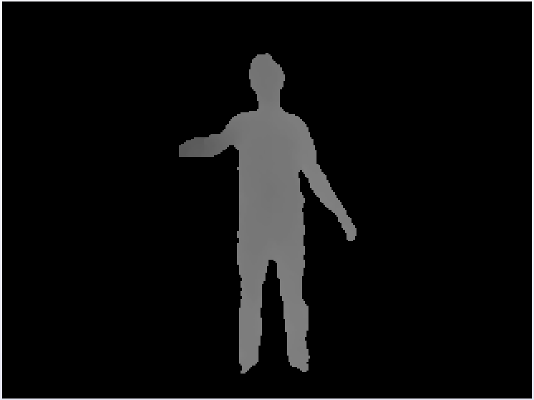


Figure: Segmented User

Geodesic Distance Map

Detect body points which are farthest from the body center ⇒ Head, Hands and Feet.

- Geodesic distance: shortest distance between two points along the surface of the body
- Represent the body point cloud as a regular graph, each of the nodes connected to its 4 or 8 neighbors
- Edge weight = 3D (geodesic) distance between neighbors
- Dijkstra's algorithm computes shortest distance from center to every graph point

COMPUTERVISUALISTIK

Geodesic Distance Map

Detect body points which are farthest from the body center $\Rightarrow$ Head, Hands and Feet.

- Geodesic distance: shortest distance between two points along the surface of the body
- Represent the body point cloud as a regular graph, each of the nodes connected to its 4 or 8 neighbors
- Edge weight = 3D (geodesic) distance between neighbors
- Dijkstra's algorithm computes shortest distance from center to every graph point

Geodesic Distance Map

Detect body points which are farthest from the body center $\Rightarrow$ Head, Hands and Feet.

- Geodesic distance: shortest distance between two points along the surface of the body
- Represent the body point cloud as a regular graph, each of the nodes connected to its 4 or 8 neighbors
- Edge weight = 3D (geodesic) distance between neighbors
- Dijkstra's algorithm computes shortest distance from center to every graph point

COMPUTERVISUALISTIK

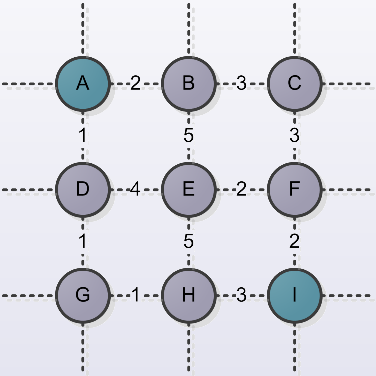
Geodesic Distance Map

Detect body points which are farthest from the body center $\Rightarrow$ Head, Hands and Feet.

- Geodesic distance: shortest distance between two points along the surface of the body
- Represent the body point cloud as a regular graph, each of the nodes connected to its 4 or 8 neighbors
- Edge weight = 3D (geodesic) distance between neighbors
- Dijkstra's algorithm computes shortest distance from center to every graph point

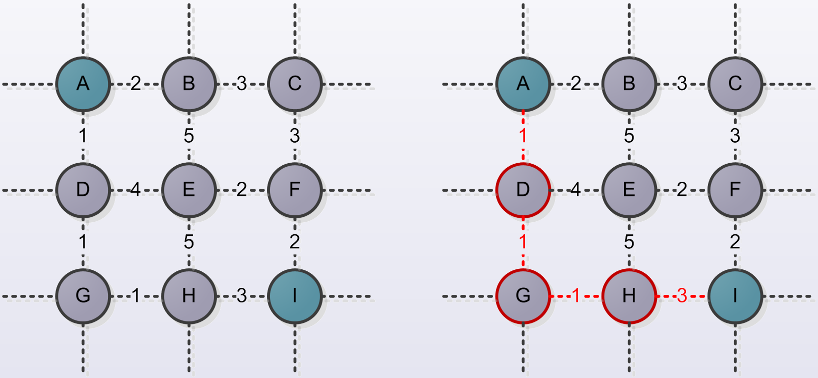
COMPUTERVISUALISTIK

Geodesic Distance Map



COMPUTERVISUALISTIK

Geodesic Distance Map



Pipeline Overview

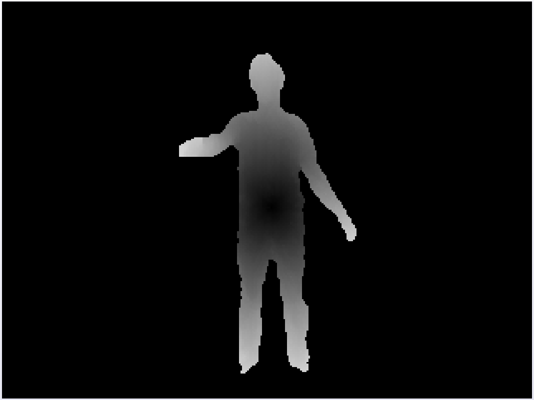


Figure: Geodesic Distance Map

Feature-Point Extraction

Find the most significant local extrema in the geodesic distance map.

- Subsample the distance map to create *Geodesic Iso-Patches*
  - each patch contains only points with $\pm x$ cm distance from the center (typical value $x \in [5 \text{ cm}, 15 \text{ cm}]$).
- identify patches that don't have neighboring patches with a higher distance
- detect local maxima within each detected patch

COMPUTERVISUALISTIK

Feature-Point Extraction

Find the most significant local extrema in the geodesic distance map.

- Subsample the distance map to create *Geodesic Iso-Patches*
  - each patch contains only points with $\pm\, x$ cm distance from the center (typical value $x \in [5\ \mathrm{cm}, 15\ \mathrm{cm}]$).
- identify patches that don't have neighboring patches with a higher distance
- detect local maxima within each detected patch

COMPUTERVISUALISTIK

Feature-Point Extraction

Find the most significant local extrema in the geodesic distance map.

- Subsample the distance map to create *Geodesic Iso-Patches*
  - each patch contains only points with $\pm\, x\ \mathrm{cm}$ distance from the center (typical value $x \in [5\ \mathrm{cm}, 15\ \mathrm{cm}]$).
- identify patches that don't have neighboring patches with a higher distance
- detect local maxima within each detected patch

COMPUTERVISUALISTIK

Feature-Point Extraction

Find the most significant local extrema in the geodesic distance map.

- Subsample the distance map to create *Geodesic Iso-Patches*
  - each patch contains only points with $\pm\, x\ \mathrm{cm}$ distance from the center (typical value $x \in [5\ \mathrm{cm}, 15\ \mathrm{cm}]$).
- identify patches that don't have neighboring patches with a higher distance
- detect local maxima within each detected patch

Pipeline Overview

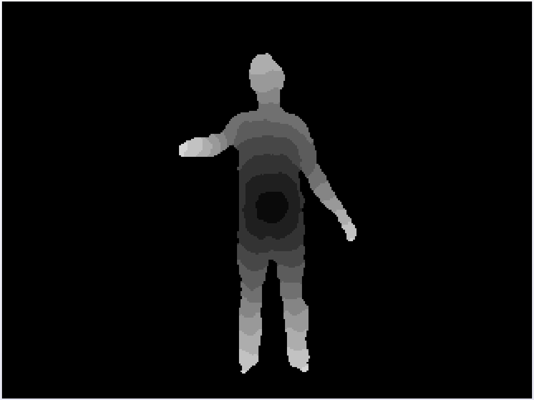


Figure: Geodesic Iso-Patches

Pipeline Overview

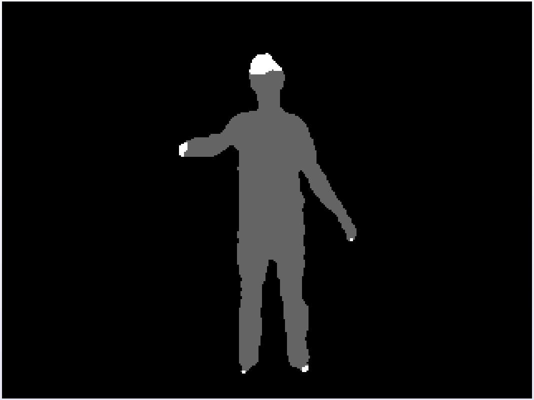


Figure: Geodesic End-Patches

Pipeline Overview

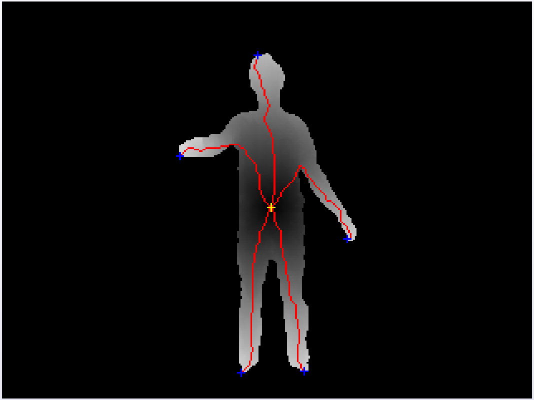


Figure: Feature Points

Tracking

Tracking of feature points for long-time analysis.

- computing *temporal information*, i.e.:
  - current velocity
  - mean velocity vectors and speeds
  - feature point lifetime
- Tracking by assigning points from the previous frame to currently detected feature points
- extrapolating feature point positions during periods of uncertainty

Tracking

Tracking of feature points for long-time analysis.

- computing *temporal information*, i.e.:
  - current velocity
  - mean velocity vectors and speeds
  - feature point lifetime
- Tracking by assigning points from the previous frame to currently detected feature points
- extrapolating feature point positions during periods of uncertainty

COMPUTERVISUALISTIK

Tracking

Tracking of feature points for long-time analysis.

- computing *temporal information*, i.e.:
  - current velocity
  - mean velocity vectors and speeds
  - feature point lifetime
- Tracking by assigning points from the previous frame to currently detected feature points
- extrapolating feature point positions during periods of uncertainty

COMPUTERVISUALISTIK

Tracking

Tracking of feature points for long-time analysis.

- computing *temporal information*, i.e.:
  - current velocity
  - mean velocity vectors and speeds
  - feature point lifetime
- Tracking by assigning points from the previous frame to currently detected feature points
- extrapolating feature point positions during periods of uncertainty

COMPUTERVISUALISTIK

Tracking

Tracking of feature points for long-time analysis.

- computing *temporal information*, i.e.:
  - current velocity
  - mean velocity vectors and speeds
  - feature point lifetime
- Tracking by assigning points from the previous frame to currently detected feature points
- extrapolating feature point positions during periods of uncertainty

COMPUTERVISUALISTIK

Tracking

Tracking of feature points for long-time analysis.

- computing *temporal information*, i.e.:
  - current velocity
  - mean velocity vectors and speeds
  - feature point lifetime
- Tracking by assigning points from the previous frame to currently detected feature points
- extrapolating feature point positions during periods of uncertainty

COMPUTERVISUALISTIK

Pipeline Overview

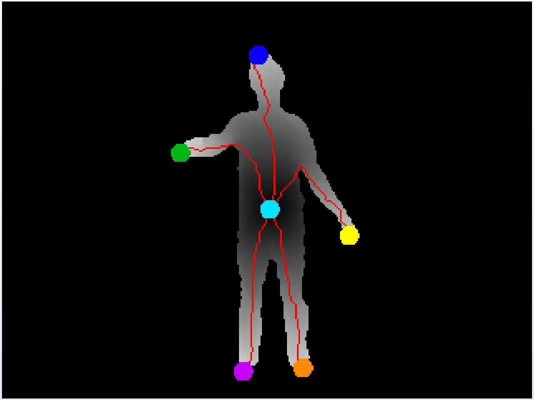


Figure: Tracked Feature Points

Energy-Functions

- Every joint in the skeleton model has to define an energy function
- The functions zero value is defined at the joint's optimal position
- Energy function: squared 3d distance from *current joint position* to *optimum position*

COMPUTERVISUALISTIK

Energy-Functions

- Every joint in the skeleton model has to define an energy function
- The functions zero value is defined at the joint's optimal position
- Energy function: squared 3d distance from *current joint position* to *optimum position*

COMPUTERVISUALISTIK

Energy-Functions

- Every joint in the skeleton model has to define an energy function
- The functions zero value is defined at the joint's optimal position
- Energy function: squared 3d distance from *current joint position* to *optimum position*

Skeleton Model

An arbitrary skeleton model to be tracked can be defined.

- 3 joint types:
  - Ball-And-Socket Joint (3 DOF)
  - Hinge Joint (1 DOF)
  - End-Affector Joint (0 DOF)
- upper body skeleton: 14 DOF (inner joints) + 4 DOF (root joint) = 18 DOF
- joint constraints reduce the search space and eliminate impossible poses
- every end-affector joint has to classify a feature point as its corresponding body part

COMPUTERVISUALISTIK

Skeleton Model

An arbitrary skeleton model to be tracked can be defined.

- 3 joint types:
  - Ball-And-Socket Joint (3 DOF)
  - Hinge Joint (1 DOF)
  - End-Affector Joint (0 DOF)
- upper body skeleton: 14 DOF (inner joints) + 4 DOF (root joint) = 18 DOF
- joint constraints reduce the search space and eliminate impossible poses
- every end-affector joint has to classify a feature point as its corresponding body part

COMPUTERVISUALISTIK

Skeleton Model

An arbitrary skeleton model to be tracked can be defined.

- 3 joint types:
  - Ball-And-Socket Joint (3 DOF)
  - Hinge Joint (1 DOF)
  - End-Affector Joint (0 DOF)
- upper body skeleton: 14 DOF (inner joints) + 4 DOF (root joint) = 18 DOF
- joint constraints reduce the search space and eliminate impossible poses
- every end-affector joint has to classify a feature point as its corresponding body part

Skeleton Model

An arbitrary skeleton model to be tracked can be defined.

- 3 joint types:
  - Ball-And-Socket Joint (3 DOF)
  - Hinge Joint (1 DOF)
  - End-Affector Joint (0 DOF)
- upper body skeleton: 14 DOF (inner joints) + 4 DOF (root joint) = 18 DOF
- joint constraints reduce the search space and eliminate impossible poses
- every end-affector joint has to classify a feature point as its corresponding body part

COMPUTERVISUALISTIK

Skeleton Model

An arbitrary skeleton model to be tracked can be defined.

- 3 joint types:
  - Ball-And-Socket Joint (3 DOF)
  - Hinge Joint (1 DOF)
  - End-Affector Joint (0 DOF)
- upper body skeleton: 14 DOF (inner joints) + 4 DOF (root joint) = 18 DOF
- joint constraints reduce the search space and eliminate impossible poses
- every end-affector joint has to classify a feature point as its corresponding body part

Skeleton Model

An arbitrary skeleton model to be tracked can be defined.

- 3 joint types:
  - Ball-And-Socket Joint (3 DOF)
  - Hinge Joint (1 DOF)
  - End-Affector Joint (0 DOF)
- upper body skeleton: 14 DOF (inner joints) + 4 DOF (root joint) = 18 DOF
- joint constraints reduce the search space and eliminate impossible poses
- every end-affector joint has to classify a feature point as its corresponding body part

COMPUTERVISUALISTIK

Skeleton Model

An arbitrary skeleton model to be tracked can be defined.

- 3 joint types:
  - Ball-And-Socket Joint (3 DOF)
  - Hinge Joint (1 DOF)
  - End-Affector Joint (0 DOF)
- upper body skeleton: 14 DOF (inner joints) + 4 DOF (root joint) = 18 DOF
- joint constraints reduce the search space and eliminate impossible poses
- every end-affector joint has to classify a feature point as its corresponding body part

COMPUTERVISUALISTIK

Energy-Minimzation

Minimizing the global skeleton energy by IK method.

- Every DOF has to be optimized independently
- CCD inverse kinematics (cyclic coordinate descent):
  - optimizing joint positions recursively, beginning at end-affector joints
- gradient descent optimization:
  - determine energy gradient
  - step in gradient direction

COMPUTERVISUALISTIK

Energy-Minimzation

Minimizing the global skeleton energy by IK method.

- Every DOF has to be optimized independently
- CCD inverse kinematics (cyclic coordinate descent):
  - optimizing joint positions recursively, beginning at end-affector joints
- gradient descent optimization:
  - determine energy gradient
  - step in gradient direction

COMPUTERVISUALISTIK

Energy-Minimzation

Minimizing the global skeleton energy by IK method.

- Every DOF has to be optimized independently
- CCD inverse kinematics (cyclic coordinate descent):
  - optimizing joint positions recursively, beginning at end-affector joints
- gradient descent optimization:
  - determine energy gradient
  - step in gradient direction

COMPUTERVISUALISTIK

Energy-Minimzation

Minimizing the global skeleton energy by IK method.

- Every DOF has to be optimized independently
- CCD inverse kinematics (cyclic coordinate descent):
  - optimizing joint positions recursively, beginning at end-affector joints
- gradient descent optimization:
  - determine energy gradient
  - step in gradient direction

COMPUTERVISUALISTIK

Energy-Minimzation

Minimizing the global skeleton energy by IK method.

- Every DOF has to be optimized independently
- CCD inverse kinematics (cyclic coordinate descent):
  - optimizing joint positions recursively, beginning at end-affector joints
- gradient descent optimization:
  - determine energy gradient
  - step in gradient direction

COMPUTERVISUALISTIK

Energy-Minimzation

Minimizing the global skeleton energy by IK method.

- Every DOF has to be optimized independently
- CCD inverse kinematics (cyclic coordinate descent):
  - optimizing joint positions recursively, beginning at end-affector joints
- gradient descent optimization:
  - determine energy gradient
  - step in gradient direction

COMPUTERVISUALISTIK

Pipeline Overview

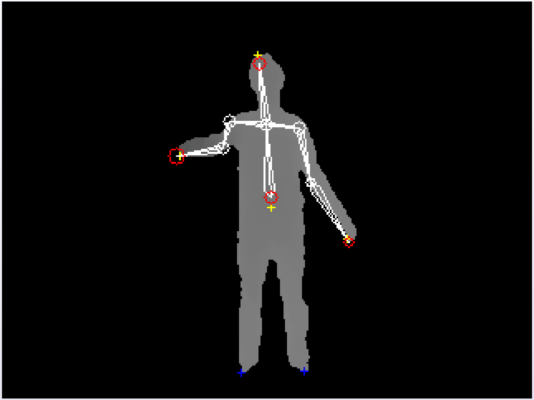


Figure: Skeleton

Special-Case Handling

1. Body parts directly in front of the body
   - Extrapolation of tracked feature points using nearest neighbors
2. Hands close to the body
   - resetting skeleton hierarchy into pre-defined standard pose
3. Hands behind the body

COMPUTERVISUALISTIK

Special-Case Handling

1. Body parts directly in front of the body
   - Extrapolation of tracked feature points using nearest neighbors
2. Hands close to the body
   - resetting skeleton hierarchy into pre-defined standard pose
3. Hands behind the body

Special-Case Handling

1. Body parts directly in front of the body
   - Extrapolation of tracked feature points using nearest neighbors
2. Hands close to the body
   - resetting skeleton hierarchy into pre-defined standard pose
3. Hands behind the body

Special-Case Handling

1. Body parts directly in front of the body
   - Extrapolation of tracked feature points using nearest neighbors
2. Hands close to the body
   - resetting skeleton hierarchy into pre-defined standard pose
3. Hands behind the body

COMPUTERVISUALISTIK

Special-Case Handling

1. Body parts directly in front of the body
   - Extrapolation of tracked feature points using nearest neighbors
2. Hands close to the body
   - resetting skeleton hierarchy into pre-defined standard pose
3. Hands behind the body
   - keeping previous position for x frames, then reset to standard pose

COMPUTERVISUALISTIK

Special-Case Handling

1. Body parts directly in front of the body
   - Extrapolation of tracked feature points using nearest neighbors
2. Hands close to the body
   - resetting skeleton hierarchy into pre-defined standard pose
3. Hands behind the body
   - keeping previous position for x frames, then reset to standard pose

Live Demonstration: Skeleton Tracking

Live Demonstration

Skeleton Tracking

COMPUTERVISUALISTIK

Explanation

The Goal-Oriented Correspondence Problem

- Matching skeleton topologies onto each other without prior knowledge
  - Matching different topologies (Elephant - Child)
  - Matching similar / same topologies (Adult - Child)
- $\Rightarrow$ Imitation learning

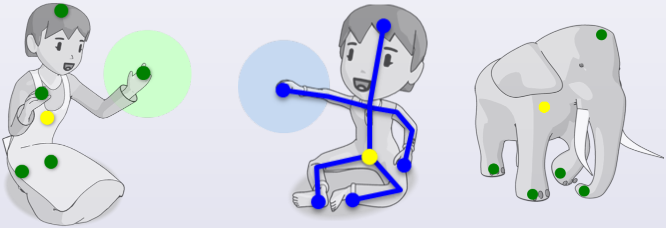


Figure: Correspondence Problem

Explanation

The Goal-Oriented Correspondence Problem

- Matching skeleton topologies onto each other without prior knowledge
  - Matching different topologies (Elephant - Child)
  - Matching similar / same topologies (Adult - Child)
- ⇒ Imitation learning

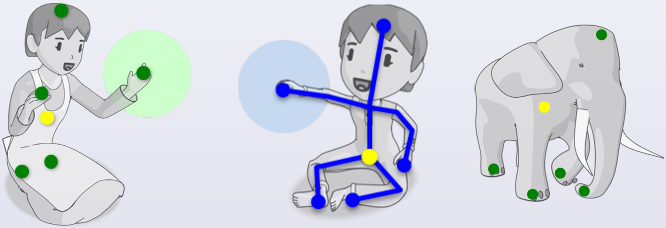


Figure: Correspondence Problem

COMPUTERVISUALISTIK

Explanation

The Goal-Oriented Correspondence Problem

- Matching skeleton topologies onto each other without prior knowledge
  - Matching different topologies (Elephant - Child)
  - Matching similar / same topologies (Adult - Child)
- $\Rightarrow$ Imitation learning

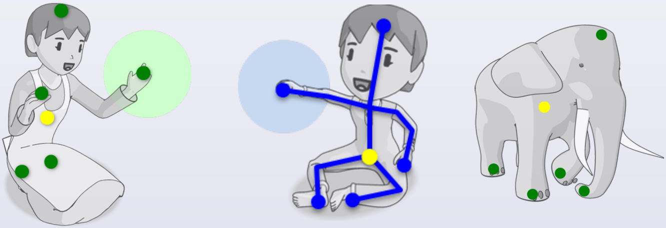


Figure: Correspondence Problem

COMPUTERVISUALISTIK

Explanation

The Goal-Oriented Correspondence Problem

- Matching skeleton topologies onto each other without prior knowledge
  - Matching different topologies (Elephant - Child)
  - Matching similar / same topologies (Adult - Child)
- $\Rightarrow$ Imitation learning

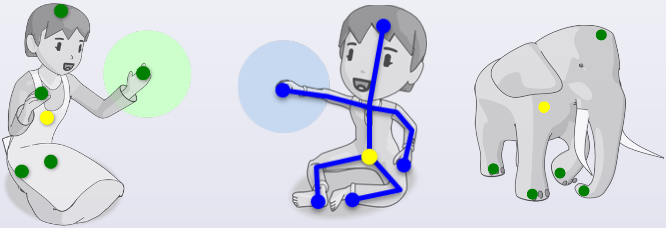


Figure: Correspondence Problem

COMPUTERVISUALISTIK

Change in Classificator Functions

- Idea: identifying corresponding body parts by motion information
- **a body part is more important than another one, if it can cause a higher effect on the environment**
  - most moving body parts are most important
- every joint defines a classificator function to classify a feature point as its corresponding body part

COMPUTERVISUALISTIK

Change in Classificator Functions

- Idea: identifying corresponding body parts by motion information
- **a body part is more important than another one, if it can cause a higher effect on the environment**
  - most moving body parts are most important
- every joint defines a classificator function to classify a feature point as its corresponding body part

COMPUTERVISUALISTIK

Change in Classificator Functions

- Idea: identifying corresponding body parts by motion information
- **a body part is more important than another one, if it can cause a higher effect on the environment**
  - most moving body parts are most important
- every joint defines a classificator function to classify a feature point as its corresponding body part

COMPUTERVISUALISTIK

Change in Classificator Functions

- Idea: identifying corresponding body parts by motion information
- **a body part is more important than another one, if it can cause a higher effect on the environment**
  - most moving body parts are most important
- every joint defines a classificator function to classify a feature point as its corresponding body part

COMPUTERVISUALISTIK

Live Demonstration: Correspondence Simulation

Live Demonstration

Correspondence Simulation

Summary

- feature-based Skeleton Tracking framework
- Tracking of arbitrary skeleton hierarchies using generic modelling
- in principle realtime capable (based on desired quality)
- independent framework for future development
- Application to correspondence problem

COMPUTERVISUALISTIK

Summary

- feature-based Skeleton Tracking framework
- Tracking of arbitrary skeleton hierarchies using generic modelling
- in principle realtime capable (based on desired quality)
- independent framework for future development
- Application to correspondence problem

COMPUTERVISUALISTIK

Summary

- feature-based Skeleton Tracking framework
- Tracking of arbitrary skeleton hierarchies using generic modelling
- in principle realtime capable (based on desired quality)
- independent framework for future development
- Application to correspondence problem

Summary

- feature-based Skeleton Tracking framework
- Tracking of arbitrary skeleton hierarchies using generic modelling
- in principle realtime capable (based on desired quality)
- independent framework for future development
- Application to correspondence problem

COMPUTERVISUALISTIK

Summary

- feature-based Skeleton Tracking framework
- Tracking of arbitrary skeleton hierarchies using generic modelling
- in principle realtime capable (based on desired quality)
- independent framework for future development
- Application to correspondence problem

COMPUTERVISUALISTIK

Conclusion & Outlook

- speed improvements:
  - performance optimization of algorithms
  - multithreading
  - GPU computing (CUDA, OpenCL)
- quality improvements: more exact energy, classificator and extrapolator functions
- feature point tracking by solving global assignment (Kuhn-Munkres algorithm: Hungarian method)
- reassigning lost feature points using local depth image features

COMPUTERVISUALISTIK

Conclusion & Outlook

- speed improvements:
  - performance optimization of algorithms
  - multithreading
  - GPU computing (CUDA, OpenCL)
- quality improvements: more exact energy, classificator and extrapolator functions
- feature point tracking by solving global assignment (Kuhn-Munkres algorithm: Hungarian method)
- reassigning lost feature points using local depth image features

COMPUTERVISUALISTIK

Conclusion & Outlook

- speed improvements:
  - performance optimization of algorithms
  - multithreading
  - GPU computing (CUDA, OpenCL)
- quality improvements: more exact energy, classificator and extrapolator functions
- feature point tracking by solving global assignment (Kuhn-Munkres algorithm: Hungarian method)
- reassigning lost feature points using local depth image features

COMPUTERVISUALISTIK

Conclusion & Outlook

- speed improvements:
  - performance optimization of algorithms
  - multithreading
  - GPU computing (CUDA, OpenCL)
- quality improvements: more exact energy, classificator and extrapolator functions
- feature point tracking by solving global assignment (Kuhn-Munkres algorithm: Hungarian method)
- reassigning lost feature points using local depth image features

COMPUTERVISUALISTIK

Conclusion & Outlook

- speed improvements:
  - performance optimization of algorithms
  - multithreading
  - GPU computing (CUDA, OpenCL)
- quality improvements: more exact energy, classificator and extrapolator functions
- feature point tracking by solving global assignment (Kuhn-Munkres algorithm: Hungarian method)
- reassigning lost feature points using local depth image features

COMPUTERVISUALISTIK

Conclusion & Outlook

- speed improvements:
  - performance optimization of algorithms
  - multithreading
  - GPU computing (CUDA, OpenCL)
- quality improvements: more exact energy, classificator and extrapolator functions
- feature point tracking by solving global assignment (Kuhn-Munkres algorithm: Hungarian method)
- reassigning lost feature points using local depth image features

COMPUTERVISUALISTIK

Conclusion & Outlook

- speed improvements:
  - performance optimization of algorithms
  - multithreading
  - GPU computing (CUDA, OpenCL)
- quality improvements: more exact energy, classificator and extrapolator functions
- feature point tracking by solving global assignment (Kuhn-Munkres algorithm: Hungarian method)
- reassigning lost feature points using local depth image features

Conclusion & Outlook

Thank you for your attention!

COMPUTERVISUALISTIK

- SHOTTON, Jamie ; FITZGIBBON, Andrew W. ; COOK, Mat ; SHARP, Toby ; FINOCCHIO, Mark ; MOORE, Richard ; KIPMAN, Alex ; BLAKE, Andrew:
  Real-time human pose recognition in parts from single depth images.
  In: *CVPR*, IEEE, 2011, 1297-1304
- SCHWARZ, Loren ; MKHYTARYAN, Artashes ; MATEUS, Diana ; NAVAB, Nassir:
  Estimating Human 3D Pose from Time-of-Flight Images Based on Geodesic Distances and Optical Flow.
  In: *IEEE Conference on Automatic Face and Gesture Recognition (FG)* (2011), March

COMPUTERVISUALISTIK