Inferring human poses from a single depth image (Kinect Algorithm)

Hauptseminar Computer Vision & Visual Tracking for Robotic Applications SS2012

Lehrstuhl für Echtzeitsysteme und Robotik Fakultät für Informatik Technische Universität München

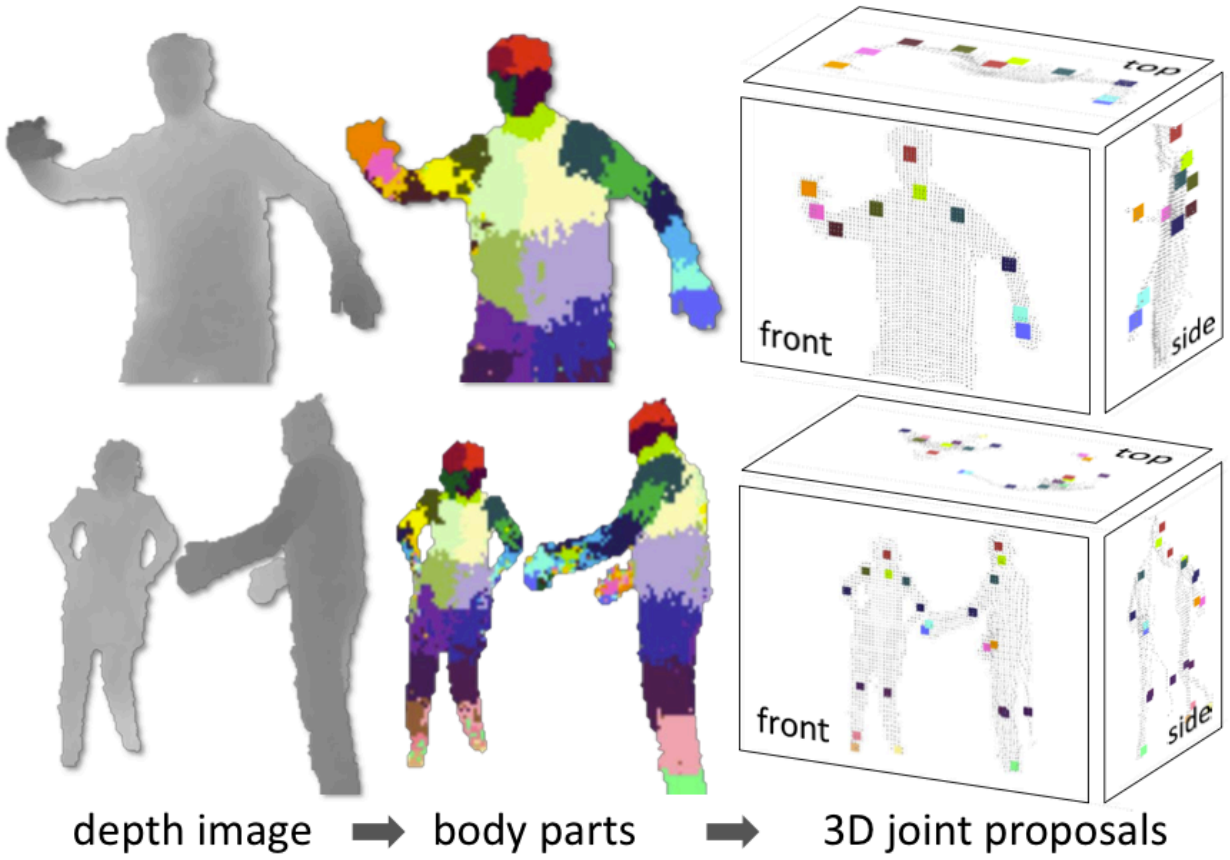
Benjamin Steer

Email: steer@in.tum.de

Gliederung – Kinect Algorithmus

1. Übersicht Kinect Algorithmus
2. Erzeugung synthetischer und realer Trainingsdaten
3. Tiefenbild Features
4. Randomized Decision Forests
5. Implementierung von Random Decision Trees in der GPU
6. Positionsvorschläge für Joints
7. Bedeutung der Ausgangsparameter/-daten

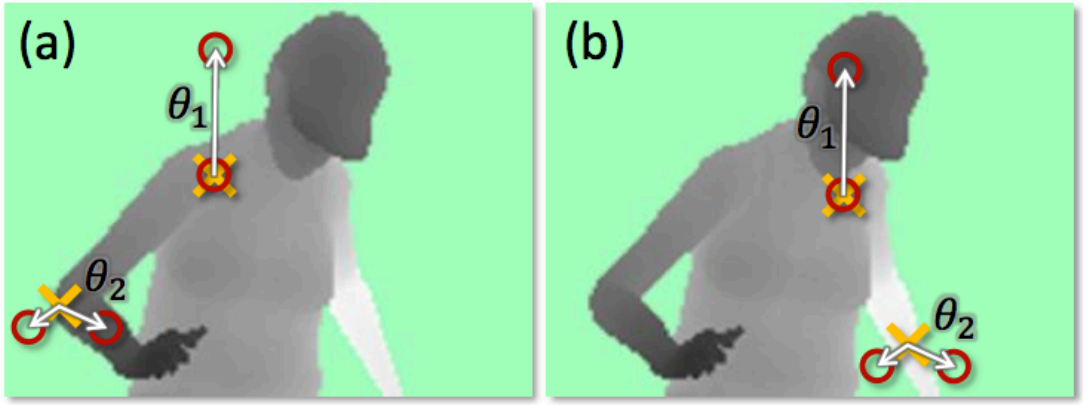
Übersicht Kinect Algorithmus



Erzeugung der Trainingsdaten

- Erzeugung mittels Motion Capturing (500k Frames an vers. Bewegungsmustern)
- Daraus werden 100k Posen erstellt (mit > 5cm Offset)
- Anwendung auf parametrisierte Körpermodelle (Größe, Gewicht, Haarform, Volumen möglicher Kleidungsstücke)

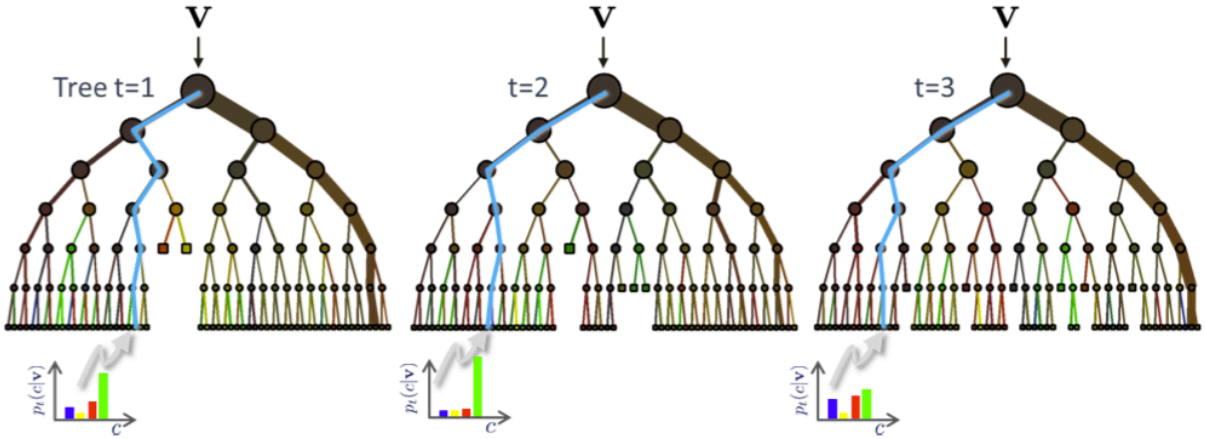
Tiefenbild Features



- Berechnung der Differenz der Tiefenwerte zweier Pixel
- Grundlage für die Entscheidungsfindung des Random Decision Forests

$$f_\theta(I, x) = d_I\left(x + \frac{u}{d_I(x)}\right) - d_I\left(x + \frac{v}{d_I(x)}\right)$$

Random Decision Forests



- Klassifikationsvorgang:

$$P(c|I,x) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|I,x)$$

Random Decision Forests - Training

- Training der Entscheidungsbäume:
  - Datensets aus verschiedenen Trainingsbildern
  - Zufällige Wahl von ca. 2000 Pixel aus jedem Bild
  - Willkürliche Wahl von Featureparametern und Schwellenwerten (Thresholds) für einen Entscheidungsknoten aus einem Featurepool
  - Feststellung der Pixelgruppen (ob Entscheidung zu linkem oder rechtem Kindknoten führt) und somit des Knoten mit dem höchsten Informationsgewinns

Random Decision Forests - Training

1. Bei N Beobachtungen in der Trainingsmenge, werden n Objekte zufälllig mit Zurücklegen gezogen.
2. Bei M Merkmalen (Features oder Dimensionen) der Beispiele werden an jedem Knoten im Baum $m \ll M$ Merkmale zufälllig gewählt, die zur Betrachtung des Schnitts (Split) genommen werden.
3. Der Baum wird voll ausgebaut und nicht zurückgeschnitten.
4. Letztendlich Speicherung der a posteriori Wahrscheinlichkeiten der Klasse nach Anna Bosch.

Random Decision Forests - Training

Entscheidungsknoten:

$\phi = (\theta, \tau)$(Feature Parameter θ und Schwellenwert τ)

Gruppierung:

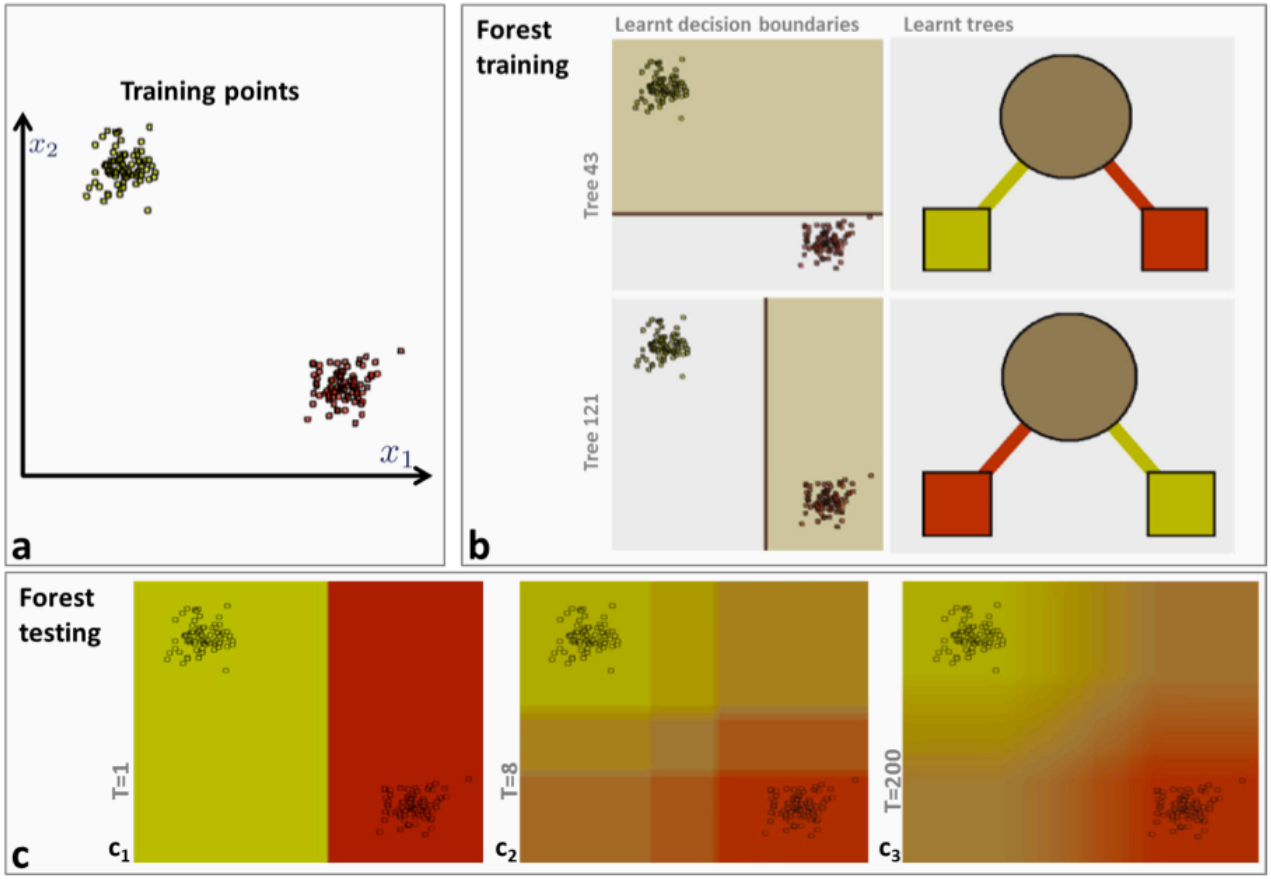
$$Q_l(\phi) = \{(I, x) \,|f_\theta(I, x) < \tau\}$$
$$Q_r(\phi) = Q \setminus Q_l(\phi)$$

Ermittlung des maximalen Informationsgehalts:

$$\phi^* = \underset{\phi}{argmax}\, G(\phi)$$

$$G(\phi) = H(Q) - \sum_{s \in \{l,r\}} \frac{|Q_s(\phi)|}{|Q|} H(Q_s(\phi))$$

Random Decision Forests - Training

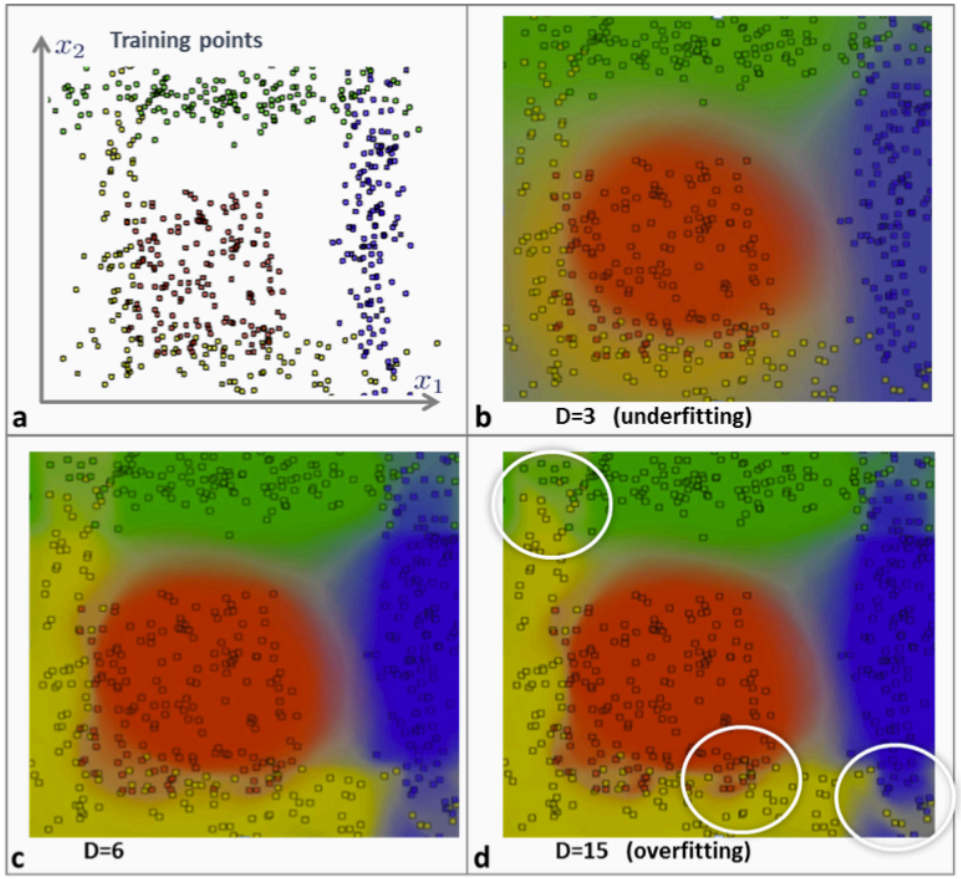


Random Decision Forests - Qualität

- Verschiedene Qualität der Ergebnisse abhängig von
  - gewählter maximaler Tiefe der Bäume
  - benutzter Weak Learner Methode

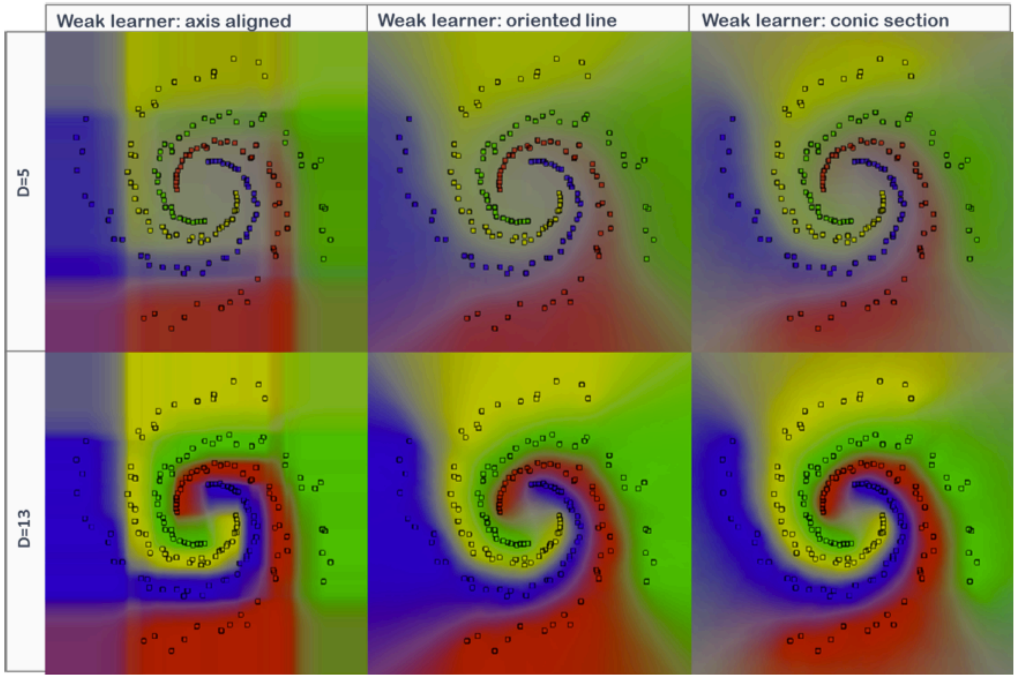
Random Decision Forests - Qualität

Auswirkung der Baumtiefe auf die Qualität der Klassifizierung



Random Decision Forests - Qualität

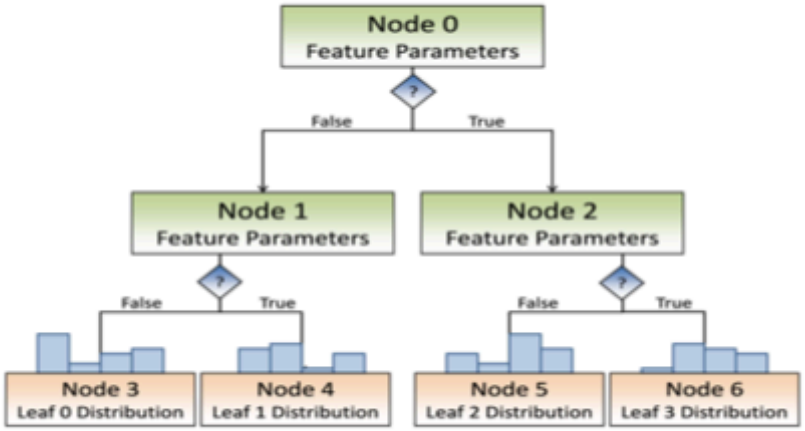
Auswirkungen der Weak Learner Methode auf die Qualität der Klassifizierung



Random Decision Forests - GPU

- Berechnung der Decision Trees auf der GPU
- 100-fache Geschwindigkeitssteigerung gegenüber Berechnung auf CPU
- Erstellung einer 2D-Textur aus Baumdatenstruktur
- Pro Entscheidungsknoten eine Zeile mit Informationen über linken Kindsknoten, Featureparameter, Schwellenwerte
- Blätterzeilen enthalten Informationen über die Klassenwahrscheinlichkeit

Random Decision Forests - GPU



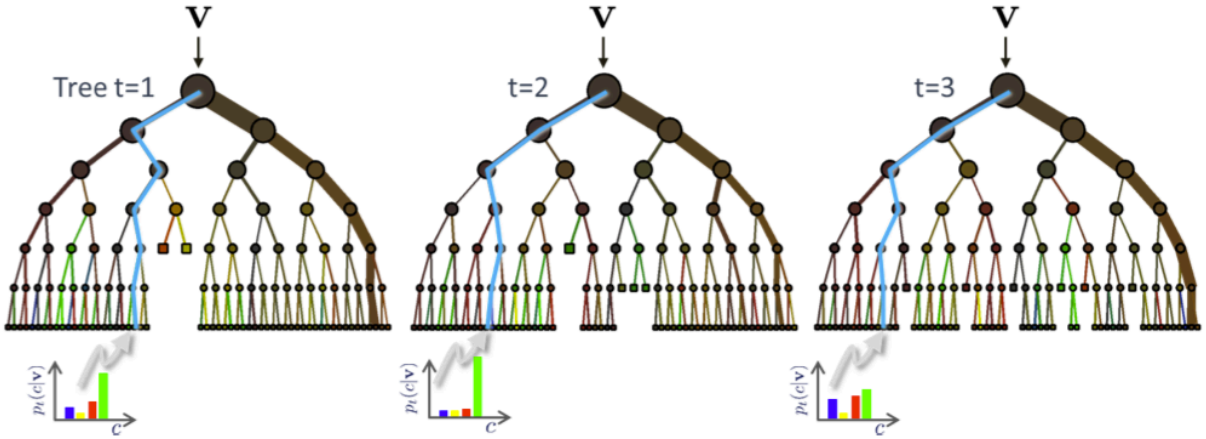
Tree 0	LeftChild LeafNode	Feature Parameters					Distribution
		Rect1	Rect2	Channel1	Channel2	Thresholds	
Node 0	1 -1	X X X X	X X X X	X X X X	X X X X	X X	
Node 1	3 -1	X X X X	X X X X	X X X X	X X X X	X X	
Node 2	5 -1	X X X X	X X X X	X X X X	X X X X	X X	
Node 3	-1 0						X X X X
Node 4	-1 1						X X X X
Node 5	-1 2						X X X X
Node 6	-1 3						X X X X

Random Decision Forests - GPU

Auswertung des Entscheidungsbaums mit Hilfe eines Pixel Shaders

```
float4 Evaluate(uniform sampler2D Forest, uniform sampler2D Input,
         uniform float2 PixSize, in float2 TexCoord : TEXCOORD0) : COLOR0
{
    float2 NodeCoord = PixSize * 0.5f;
    // Iterate over the levels of the tree, from the root down...
    [unroll] for (int nLevel = 1; nLevel < MAX_DEPTH; nLevel++)
    {
        float LeftChild = tex2D(Forest, NodeCoord).x;
        // Read the feature parameters for this node...
        Parameters Params = ReadParams(Forest, NodeCoord, PixSize);
        // Perform the user-supplied Boolean test for this node...
        bool TestResult = TestFeature(Input, TexCoord, Params);
        // Move the node coordinate according to the result of the test...
        NodeCoord.y = LeftChild + TestResult * PixSize.y;
    }
    // Read the output distribution associated with this leaf node...
    return Distribution(Forest, NodeCoord);
}
```

Random Decision Forests - Auswertung



- Klassifikationsvorgang:

$$P(c|I,x) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|I,x)$$

Positionsvorschläge für Joints

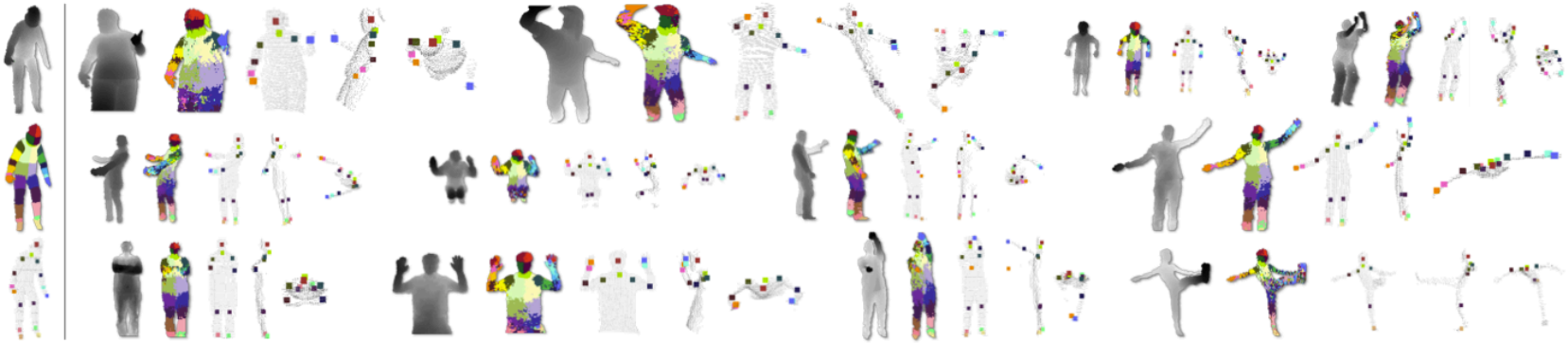
- Berechnung der Joints mit Hilfe der Mean Shift Methode
- Ermittlung des dichtesten Bereichs im Umfeld des Startpunkts der Mean Shift Methode
- Wiederholung bis ein bestimmter Konvergenzwert erreicht wird
- Pixel mit hoher Wahrscheinlichkeit als Startpunkte

$$f_c(\hat{x}) \propto \sum_{i=1}^{N} w_{ic} \exp\left(-\left\| \frac{\hat{x}-\hat{x}_i}{b_c} \right\|^2\right)$$

$$w_{ic} = P(c|I, x_i) \cdot d_I(x_i)^2$$

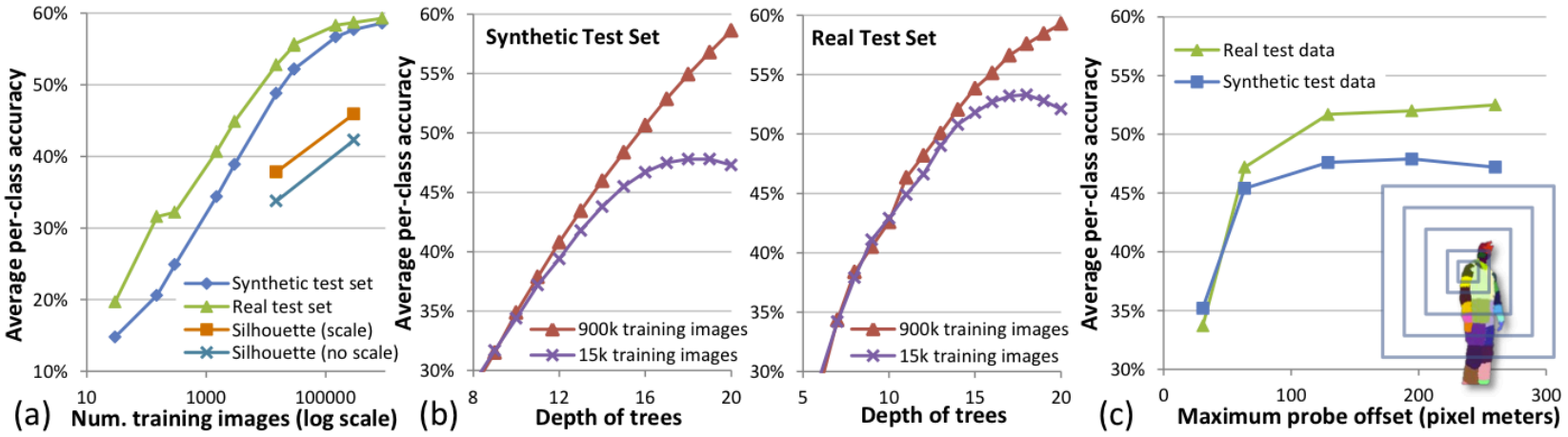
Positionsvorschläge für Joints

- Punkt mit der höchsten Dichte liegt auf Oberfläche des Körpers
- z-Offset ins Körperinnere mittels im voraus ermittelter, optimierter Werte



Bedeutung der Ausgangsparameter/-daten

- Anzahl der Trainingsbilder und Tiefe der Bäume ausschlaggebend für Qualität
- Nur wenn beide in ausreichender Zahl vorhandenen sind werden Ergebnisse verbessert
- Ansonsten Vorkommnisse von Overfitting



Ende

Vielen Dank für die Aufmerksamkeit!