Pose Estimation

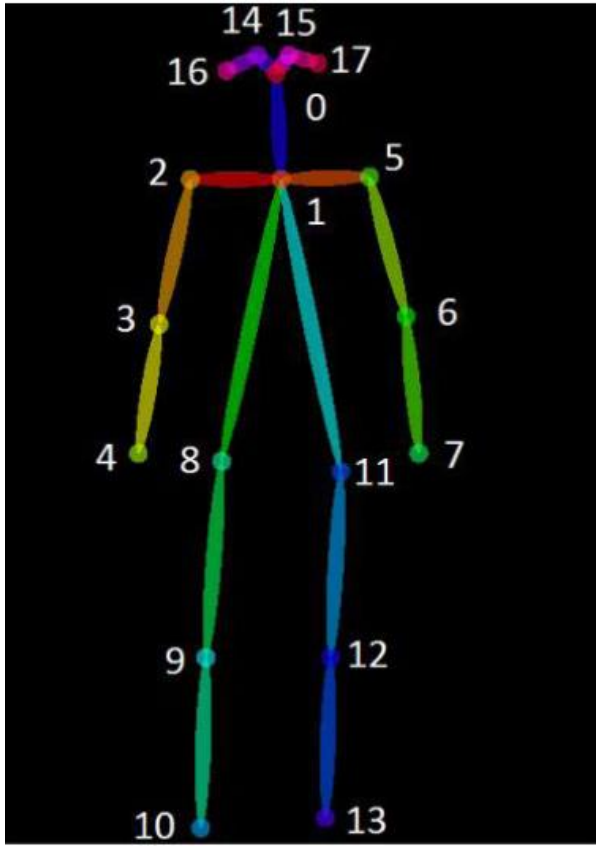
한양대학교

컴퓨터소프트웨어학부

박현준

Preview

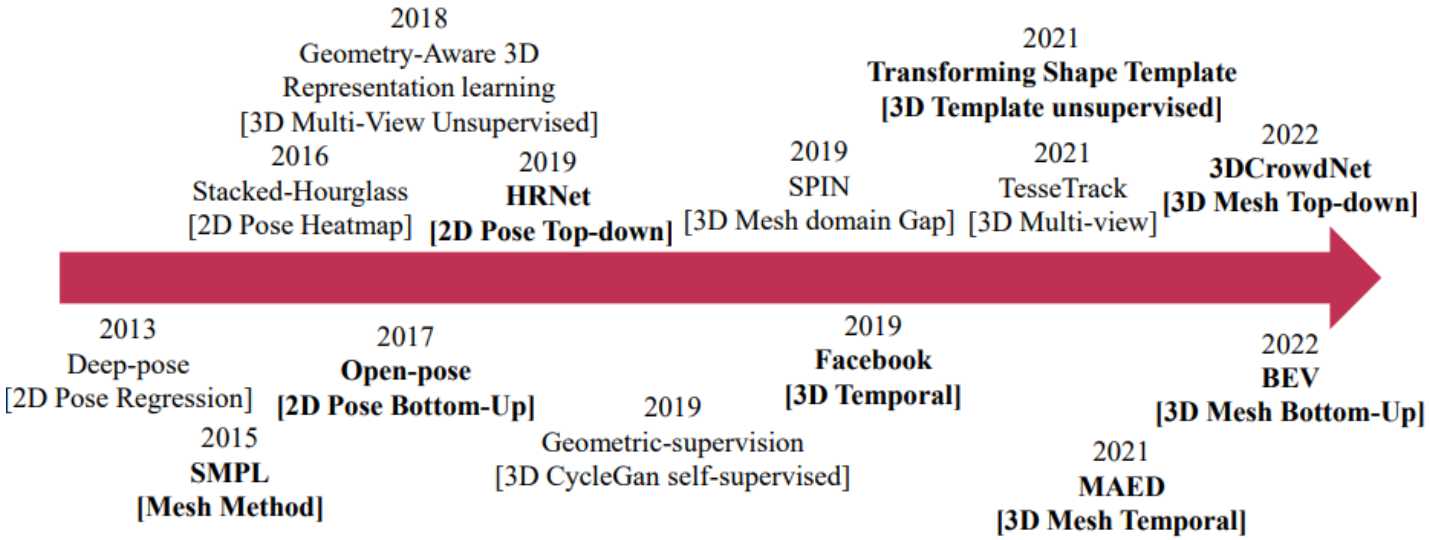
What is Pose Estimation?

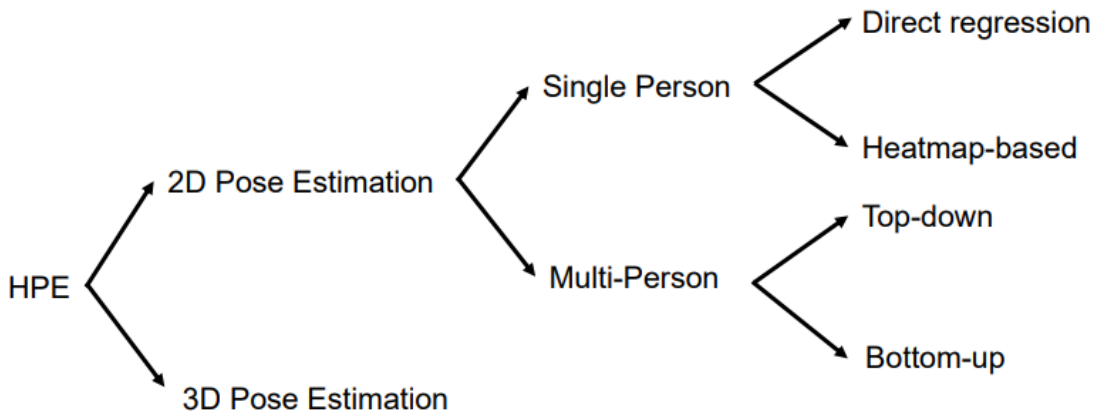


Pose Estimation is a matter of **localizing** and **estimating**
how a person's or object's joints are organized in an image or video

Preview

What is Pose Estimation?



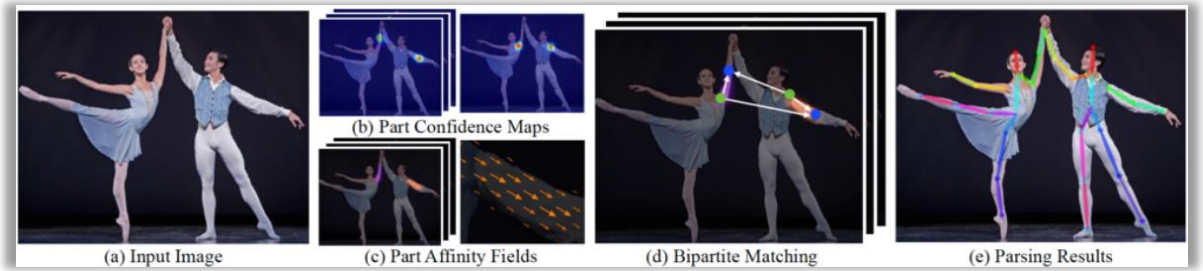


Preview

2D Pose Estimation

Top-down

- Detection first
- Cannot estimate if not detected
- Calculation ∝ #object

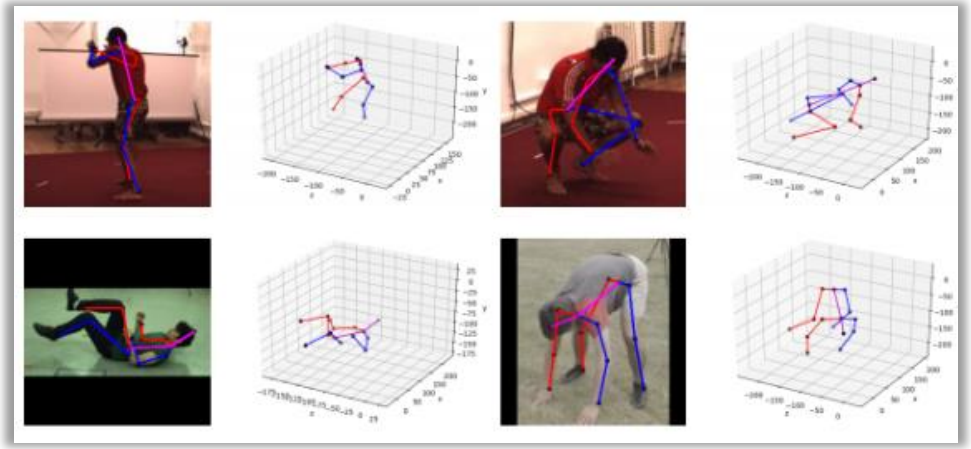


Bottom-up

- Estimation first
- Too many combination of matching joints and it is time consuming
- Difficult to enhance accuracy
- Can apply real-time

Preview

3D Pose Estimation



- Find joint position and depth using camera parameters
- Top-down, Bottom-up, Temporal, Monocular, …
- Unsupervised Learning

3D Mesh Estimation

- Estimate Mesh in image using SMPL Parameters
  - SMPL Parameters : $\Phi = \{\bar{T}, \mathcal{W}, \mathcal{S}, \mathcal{J}, \mathcal{P}\}$
- Top-down, Bottom-up, Temporal, …
- Try to reduce 2D, 3D benchmark domain gap

Preview

Metrics

*Correctly Predicted : $\|\sigma - \hat{\sigma}\| < \epsilon$

$\sigma = real,\ \hat{\sigma} = predicted,\ \epsilon = threshold$

Part

Keypoint

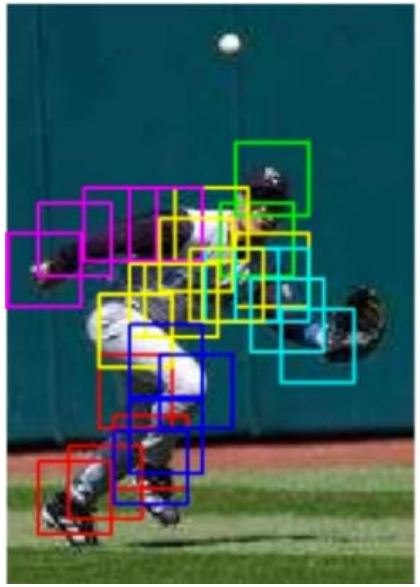
1. **PCP (Percentage of Correct Parts)**

$$PCP = \frac{\#correctly\ predicted\ parts}{\sum \#parts} \times 100$$

2. **PCK (Percentage of Correct Keypoints)**

$$PCK = \frac{\#correctly\ predicted\ keypoints}{\sum \#keypoints} \times 100$$

PCK_h : threshold of PCK

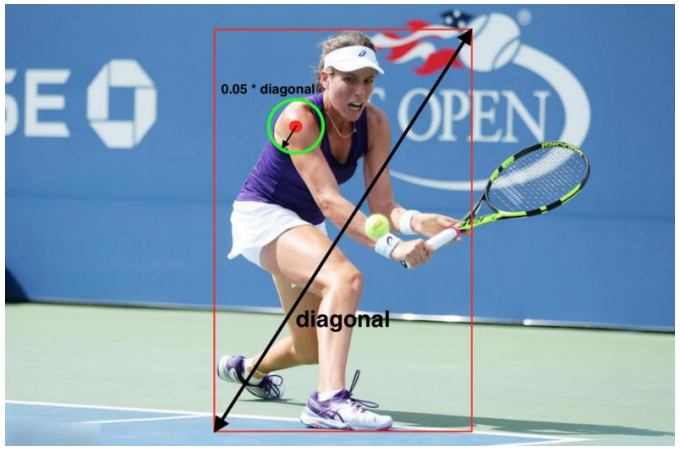


Preview

Metrics

3. PDJ (Percentage of Detected Joints)

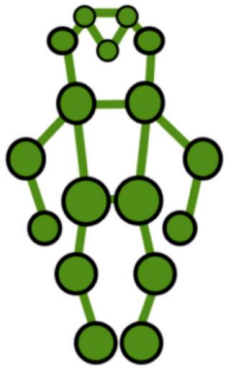
$$PDJ = \frac{\sum_{i=1}^{n} bool\,(d_i < 0.05 * diagonal)}{n}$$



4. OKS (Object Keypoint Similarity)

$$OKS = \frac{\sum_i \exp(-d_i^2/2s^2k_i^2)\delta(v_i > 0)}{\sum_i \delta(v_i > 0)}$$

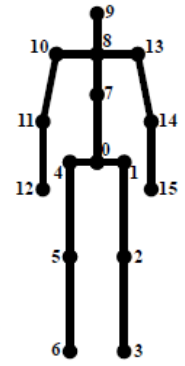
Keypoint	k_i
hips	0.107
ankles	0.089
knees	0.087
shoulders	0.079
elbows	0.072
wrists	0.062
ears	0.035
nose	0.026
eyes	0.025



5. MPJPE (Mean Per Joint Position Error)

$$MPJPE = \frac{1}{T}\frac{1}{N}\sum_{t=1}^{T}\sum_{i=1}^{N}\left\|\left(J_i^t - J_{root}^t\right) - \left(\hat{J}_i^t - \hat{J}_{root}^t\right)\right\|_2, root : Pelvis$$

0. Pelvis
1. R. Hip
2. R. Knee
3. R. Ankle
4. L. Hip
5. L. Knee
6. L. Ankle
7. Thorax
8. Neck
9. Head
10. L. Shoulder
11. L. Elbow
12. L. Wrist
13. R. Shoulder
14. R. Elbow
15. R. Wrist



Preview

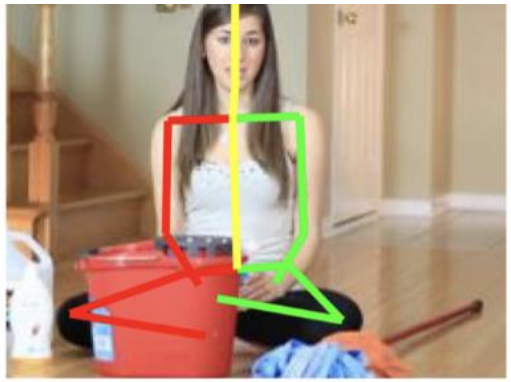
Dataset

2D Pose Estimation

- **LSP (Leeds Sports Pose)**
  - 2,000+ images
- **MPII Human Pose**
  - 25,000+ images with 40,000+ people
- **MS COCO**
  - 60,000+ images with 150,000+ people
- **AI Challenger**
  - 300,000+ images with 700,000+ people

LSP	MPII	MS COCO	AI Challenger
Right ankle	Right ankle	Nose	Right shoulder
Right knee	Right knee	Left eye	Right elbow
Right hip	Right hip	Right eye	Right wrist
Left hip	Left hip	Left ear	Left shoulder
Left knee	Left knee	Right ear	Left elbow
Left ankle	Left ankle	Left shoulder	Left wrist
Right wrist	Pelvis	Right shoulder	Right hip
Right elbow	Thorax	Left elbow	Right knee
Right shoulder	Upper neck	Right elbow	Right ankle
Left shoulder	Head top	Left wrist	Left hip
Left elbow	Right wrist	Right wrist	Left knee
Left wrist	Right elbow	Left hip	Left ankle
Neck	Right shoulder	Right hip	Top of the head
Head top	Left shoulder	Left knee	Neck
-	Left elbow	Right knee	-
-	Left wrist	Left ankle	-
-	-	Right ankle	-

LSP [27]



MPII [28]

MS COCO [29]

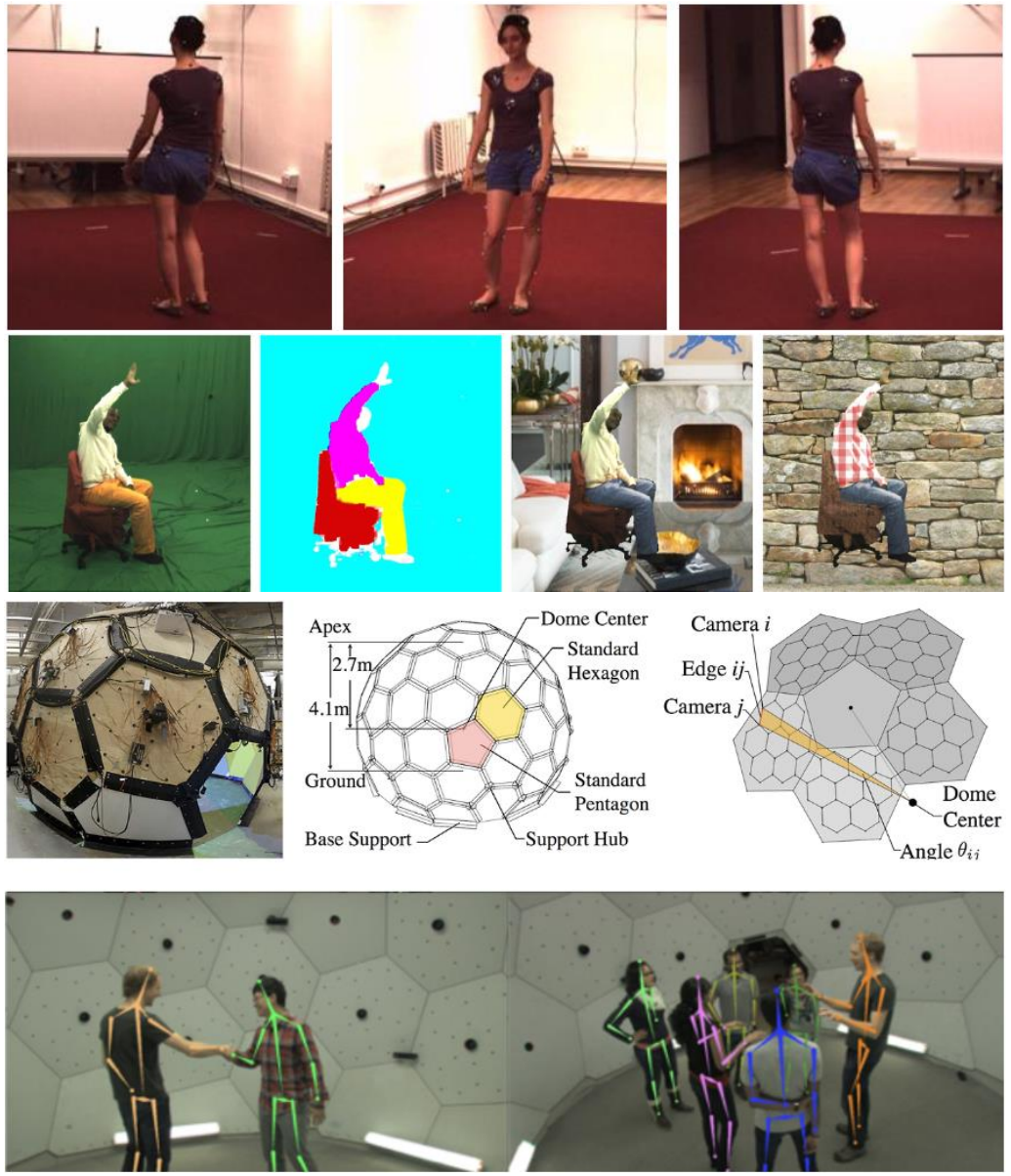
AI Challenger [30]

Preview

Dataset

3D Pose Estimation

- **Human3.6M**
  - 17 scenario (Smoking, Calling, Debate …)
  - 3.6M+ images
- **HumanEva-I**
  - 6 general movement (Walking, Jogging, …)
  - 40,000+ images
- **MPI-INF-3DHP**
  - Scalability because of green screen
  - 130M+ images
- **CMU Panoptic**
  - 65 sequence, 150M pose



Contents

DeepPose：Human Pose Estimation via Deep Neural Networks

OpenPose：Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields

SMPL：A Skinned Multi-Person Linear Model

DeepPose

: Human Pose Estimation via Deep Neural Networks

CVPR 2014

Introduction

"Localization of human joints"

Local detectors → modeling only a small subset of all interactions between body parts

⇒ Holistic view of human pose estimation … **Deep Neural Network**
(∵ DNN have shown outstanding performance on **visual classification tasks** and **object localization**)

[Strength of DNN]

1. Capability of capturing full context of each body joint
2. Substantially simpler to formulate than methods based on graphical models

   → No need to explicitly design
   - Feature representations and detectors for parts
   - Model topology and interactions between joints

Cascade of DNN-based pose predictors == Increased precision of joint localization

Pose Vector

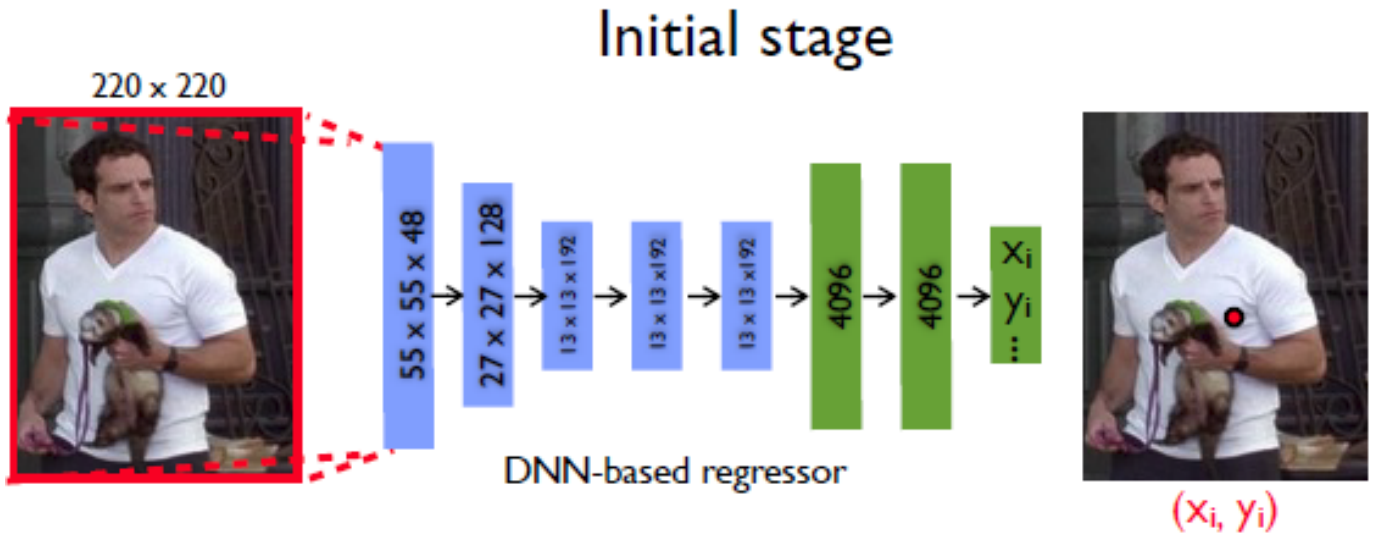
$y = (\dots, y_i^T, \dots)^T, i \in \{1, \dots, k\}$ where y_i contains x, y coordinates of the i^{th} joint

$$N(y_i; b) = \begin{pmatrix} 1/b_w & 0 \\ 0 & 1/b_h \end{pmatrix} (y_i - b_c) \quad \Rightarrow \quad N(y_i; b) = (\dots, N(y_i; b)^T, \dots)^T$$

$N(x; b)$: crop of the image x by the bounding box b

- $b = (b_c, b_w, b_h)$
- b_c : center of bounding box
- b_w : width of bounding box
- b_h : height of bounding box

Pose Estimation as DNN-based Regression



$$y^* = N^{-1}(\psi(N(x);\theta),\ \ \psi(x,\theta) \in \mathbb{R}^{2k}\ \ (\theta : model\ parameter)$$

- Architecture : AlexNet
- First Layer → Input : predefined size of an image
- Last Layer → Output : $2k$ joint coordinates
- #Parameters = 40M
- Mini-batch size = 128
- Data augmentation : Random translation & Left/Right flip

$$argmin_\theta \sum_{(x,y)\in D_N} \sum_{i=1}^{k} \|y_i - \psi_i(x;\theta)\|_2^2$$

Cascade of Pose Regressors

Figure 2. Left: schematic view of the DNN-based pose regression. We visualize the network layers with their corresponding dimensions, where convolutional layers are in blue, while fully connected ones are in green. We do not show the parameter free layers. Right: at stage s, a refining regressor is applied on a sub image to refine a prediction from the previous stage.

Stage 1: $y^1 \leftarrow N^{-1}(\psi(N(x;b^0);\theta_1);b^0)$

$$y_i^s \leftarrow y_i^{(s-1)}$$

Stage s: $y_i^s \leftarrow y_i^{(s-1)} + N^{-1}(\psi_i(N(x;b);\theta_s);b)$

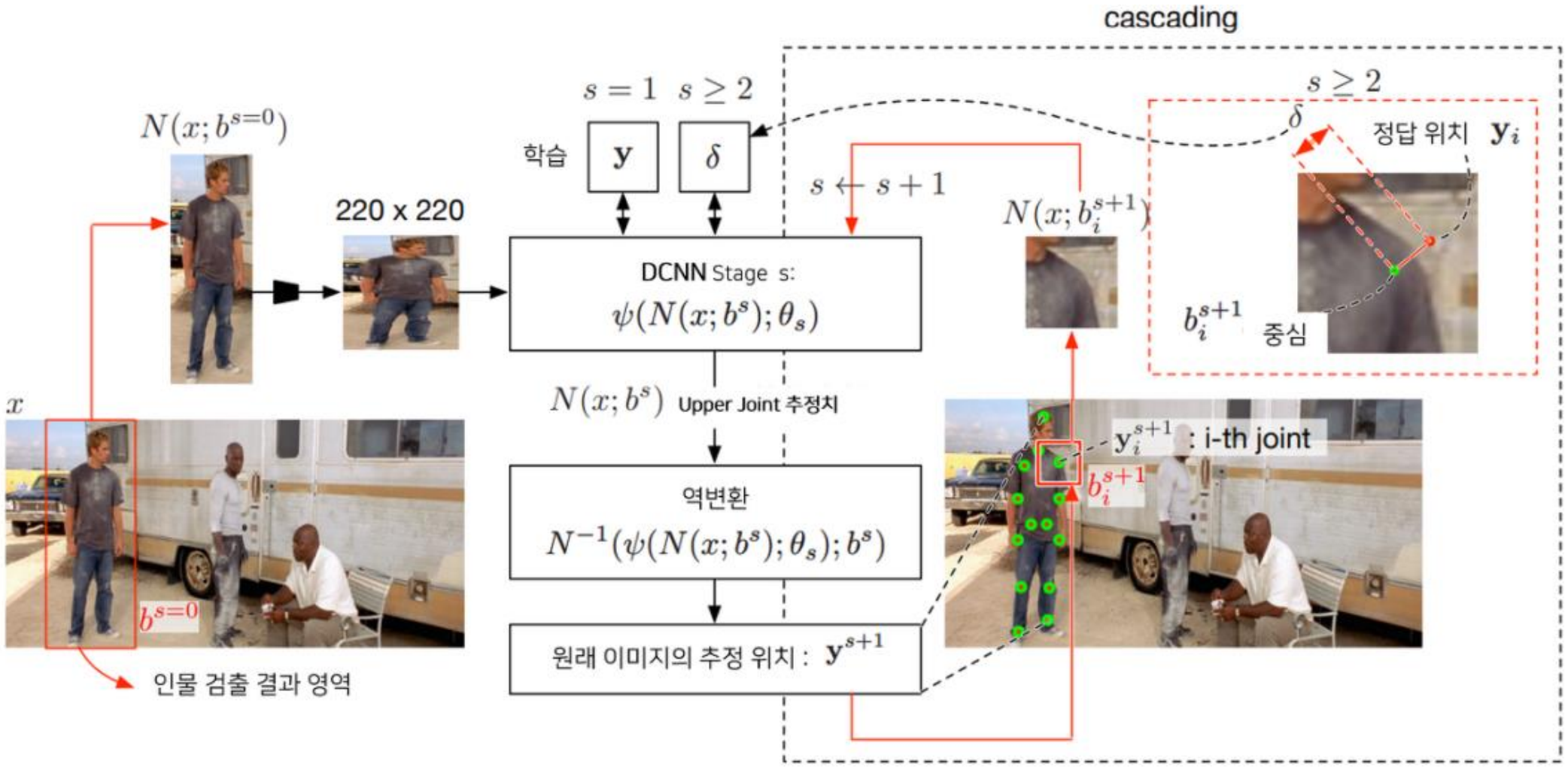
$$\text{for } b = b_i^{(s-1)}$$

$$b_i^s \leftarrow (y_i^s, \sigma diam(y^s), \sigma diam(y^s))$$

$$D_A^s = \{\big(N(x;b), N(y_i;b)\big)|$$
$$(x,y_i) \sim D, \delta \sim \mathcal{N}_i^{s-1},$$
$$b = \big(y_i + \delta, \sigma diam(y)\big)\}$$

$$\theta_s = argmin_\theta \sum_{(x,y_i)\in D_A^s} \|y_i - \psi_i(x;\theta)\|_2^2$$

Cascade of Pose Regressors



Evaluation

Dataset

- **FLIC (Frames Labeled In Cinema)**
  - 4,000 train / 1,000 test images obtained from popular Hollywood movies
  - Diverse poses and clothing
  - 10 upper body joints are labeled

- **LSD (Leeds Sports Dataset)**
  - 11,000 train / 1,000 test images obtained from sports activities
  - Majority of people have 150-pixel height
  - Full body is labeled with 14 joints

Evaluation

Metric

- **PCP (Percentage of Correct Parts)**
  - Measures detection rate of limbs
  - |two predicted joint locations - true limb joint locations|
  - Drawback of penalizing shorter limbs (hard to detect)

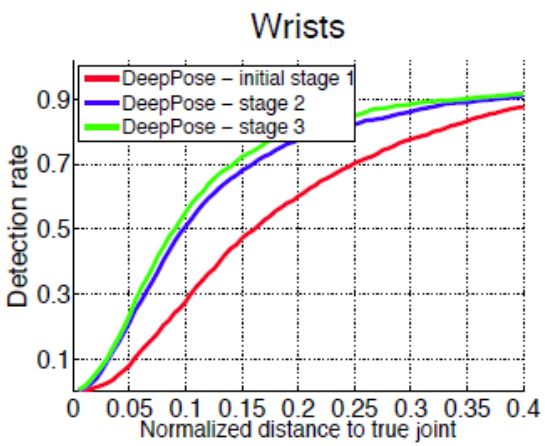
$$PCP = \frac{\#correctly\ predicted\ parts}{\sum \#parts} \times 100$$

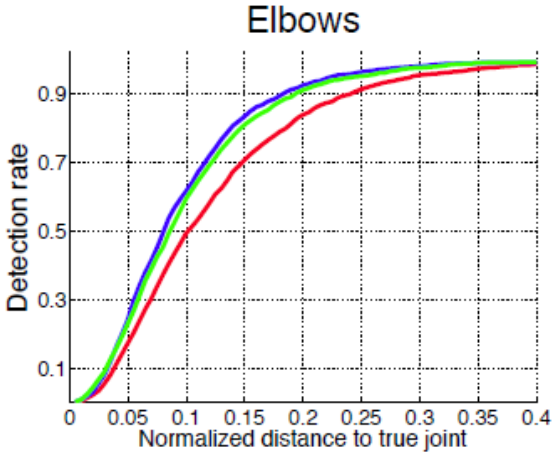
- **PDJ (Percentage of Detected Joints)**
  - $|Predicted - True| < torso\ diameter$
  - Majority of people have 150-pixel height
  - Alleviate the drawback of PCP
    (∵ detection criteria for all joints are based on the same distance threshold)

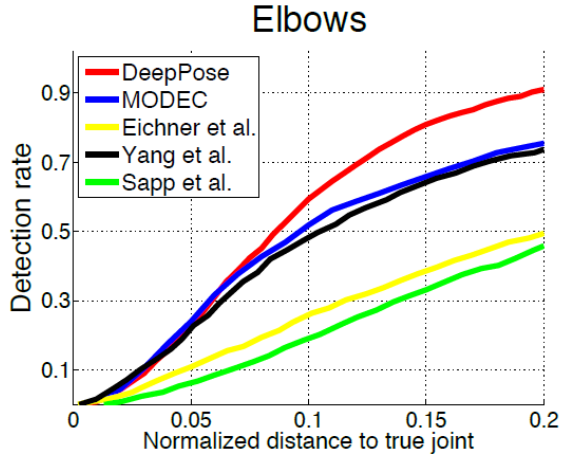
$$PDJ = \frac{\sum_{i=1}^{n} bool\,(d_i < 0.05 * diagonal)}{n}$$

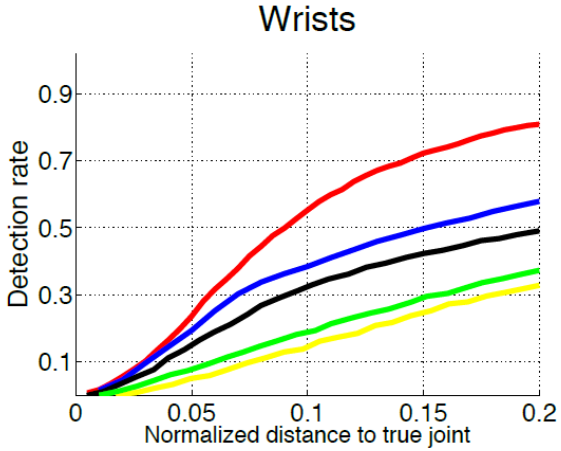
Result

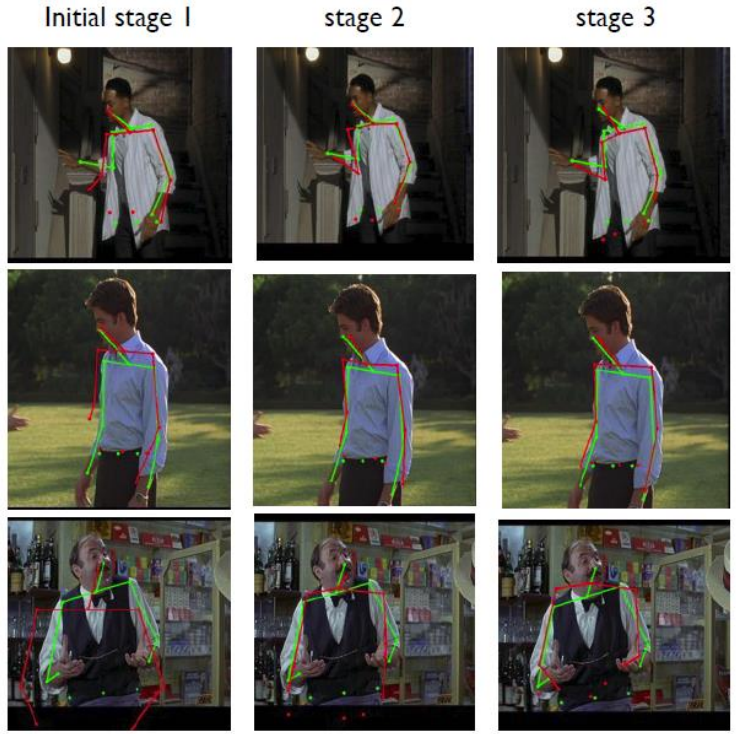
PDJ on FLIC











Method	Arm		Leg		Ave.
	Upper	Lower	Upper	Lower	
DeepPose-st1	0.5	0.27	0.74	0.65	0.54
DeepPose-st2	**0.56**	0.36	**0.78**	0.70	0.60
DeepPose-st3	**0.56**	**0.38**	0.77	**0.71**	**0.61**
Dantone et al. [2]	0.45	0.25	0.65	0.61	0.49
Tian et al. [24]	0.52	0.33	0.70	0.60	0.56
Johnson et al. [13]	0.54	**0.38**	0.75	0.66	0.58
Wang et al. [25]	**0.565**	0.37	0.76	0.68	0.59
Pishchulin [17]	0.49	0.32	0.74	0.70	0.56

Result

PCP on LSP

Method	Arm		Leg		Ave.
	Upper	Lower	Upper	Lower	
DeepPose-st1	0.5	0.27	0.74	0.65	0.54
DeepPose-st2	**0.56**	0.36	**0.78**	0.70	0.60
DeepPose-st3	**0.56**	**0.38**	0.77	**0.71**	**0.61**
Dantone et al. [2]	0.45	0.25	0.65	0.61	0.49
Tian et al. [24]	0.52	0.33	0.70	0.60	0.56
Johnson et al. [13]	0.54	**0.38**	0.75	0.66	0.58
Wang et al. [25]	**0.565**	0.37	0.76	0.68	0.59
Pishchulin [17]	0.49	0.32	0.74	0.70	0.56

Table 1. Percentage of Correct Parts (PCP) at 0.5 on LSP for DeepPose as well as five state-of-art approaches.

Method	Arm		Leg		Ave.
	Upper	Lower	Upper	Lower	
DeepPose	0.8	0.75	0.71	0.5	0.69
Pishchulin [17]	0.80	0.70	0.59	037	0.62
Johnson et al. [13]	0.75	0.67	0.67	0.46	0.64
Yang et al. [26]	0.69	0.64	0.55	0.35	0.56

Table 2. Percentage of Correct Parts (PCP) at 0.5 on Image Parse dataset for DeepPose as well as two state-of-art approaches on Image Parse dataset. Results obtained from [17].

PDJ on LSP

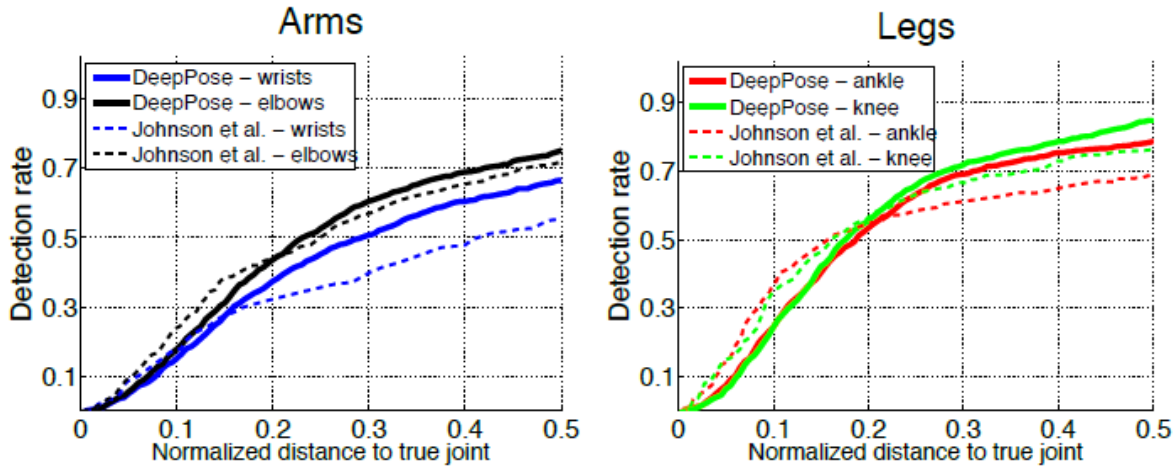


Figure 4. Percentage of detected joints (PDJ) on LSP for four limbs for DeepPose and Johnson et al. [13] over an extended range of distances to true joint: $[0, 0.5]$ of the torso diameter. Results of DeepPose are plotted with solid lines while all the results by [13] are plotted in dashed lines. Results for the same joint from both algorithms are colored with same color.

Result

FLIC

LSP

Conclusion

Strength

- First application of Deep Neural Networks to human pose estimation
- CNN which was originally designed for classification tasks can be applied to different task of localization
- Advantage of capturing context and reasoning about pose in a holistic manner
- Achieve state-of-the-art

Limitation

- When predicting joint, it does not consider spatial correlation between different joints
- Calculation inefficiency (Crop image iteratively and training CNN in each step)
- Solve problem solely using regression method

OpenPose

: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields

CVPR 2018

Introduction

Challenges

1. Each image may contain an unknown #people that can appear at any position or scale
2. Interactions between people induce complex spatial interference, making association of parts difficult (contact, occlusion, limb articulation …)
3. Run-time complexity $\propto$ #people in the image, making real-time performance difficult

Approach

Top-down approach

- Person detector
- Single-person pose estimation

⇒ Directly leverage existing techniques for single-person pose estimation

Bottom-up approach

- Do not directly use global contextual cues from other body parts and other people

Related Work

Single Person Pose Estimation

Tree-structured graphical models

Encode the spatial relationship between adjacent parts following a kinematic chain

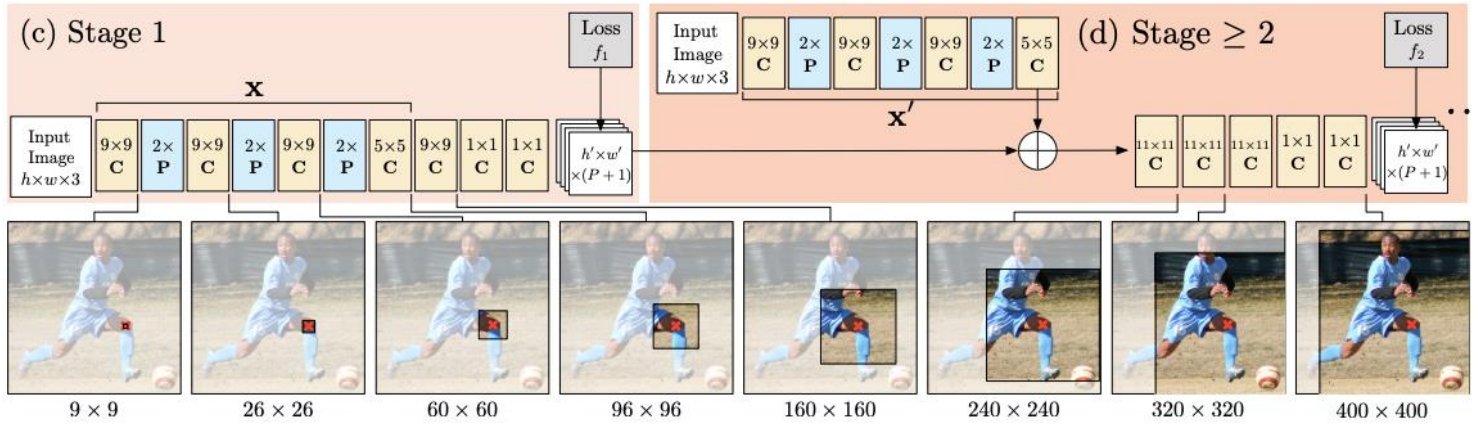
Non-tree models

Augment the tree structure with additional edges to capture occlusion, symmetry, and long-range relationships

Related Work

Single Person Pose Estimation

CPM (Convolutional Pose Machines)



Implicitly capture
global spatial dependencies
by designing networks
with large receptive fields

Stacked Hourglass Architecture

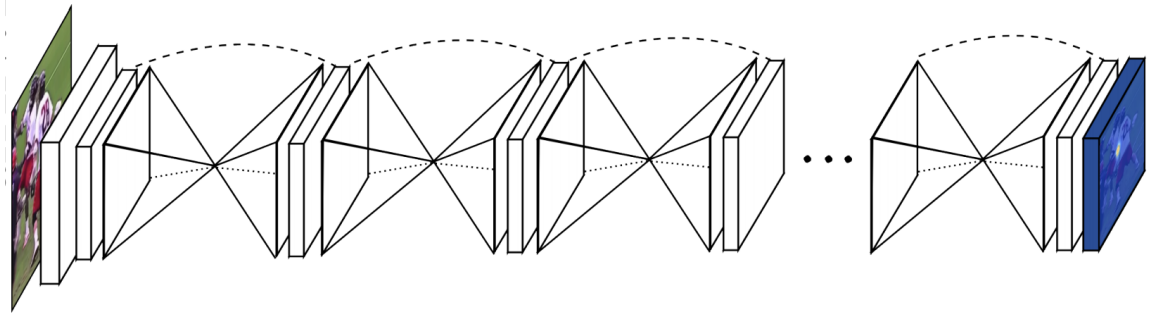
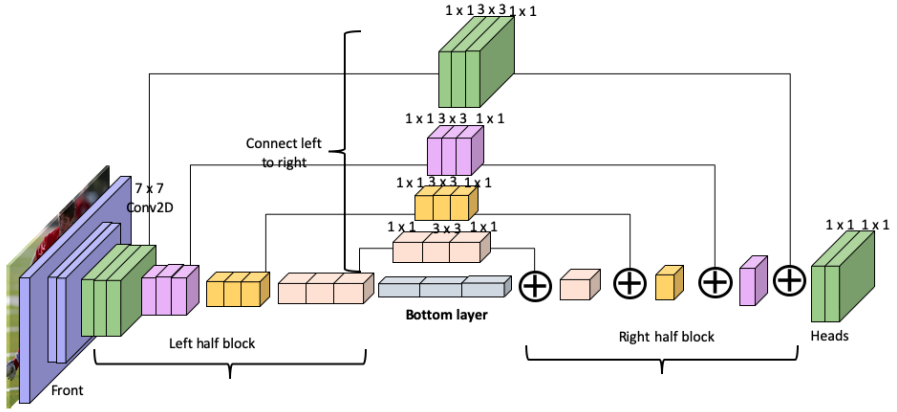


Fig. 1. Our network for pose estimation consists of multiple stacked hourglass modules which allow for repeated bottom-up, top-down inference.



Shih-En Wei, Varun Ramarkrishna, Takeo Kanade, Yaser Sheikh. 「***Convolutional Pose Machines***」*, 2016*
Alejandro Newell, Kaiyu Yang, Jia Deng. 「***Stacked Hourglass Networks for Human Pose Estimation***」*, 2016*

Related Work

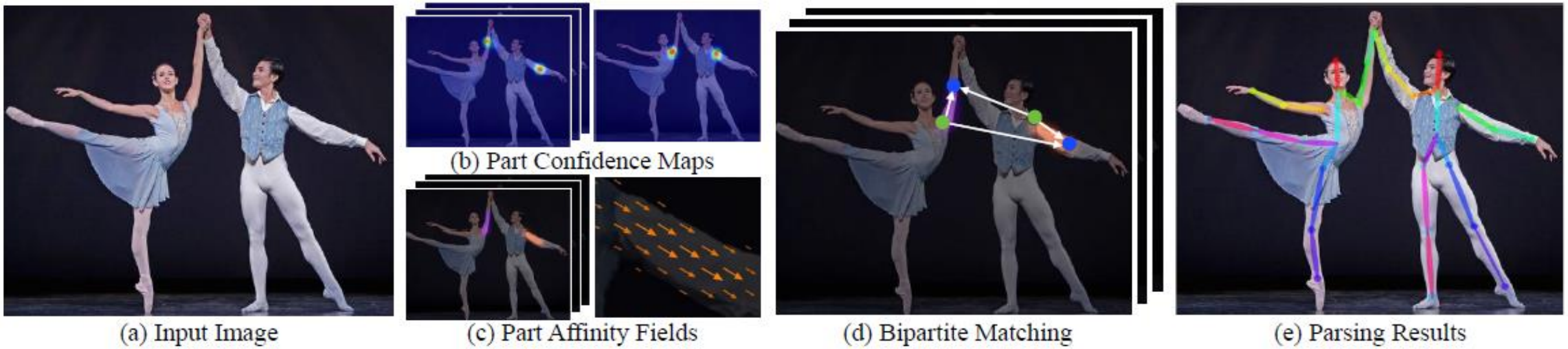
Multi-Person Pose Estimation

- Early commitment on person detection
- Capture the spatial dependencies across different people that require global reference

⇒ Consider inter-person dependencies == **Part Affinity Fields (PAFs)**

Consisting of a set of flow fields that encodes unstructured pairwise relationships between body parts of a variable number of people

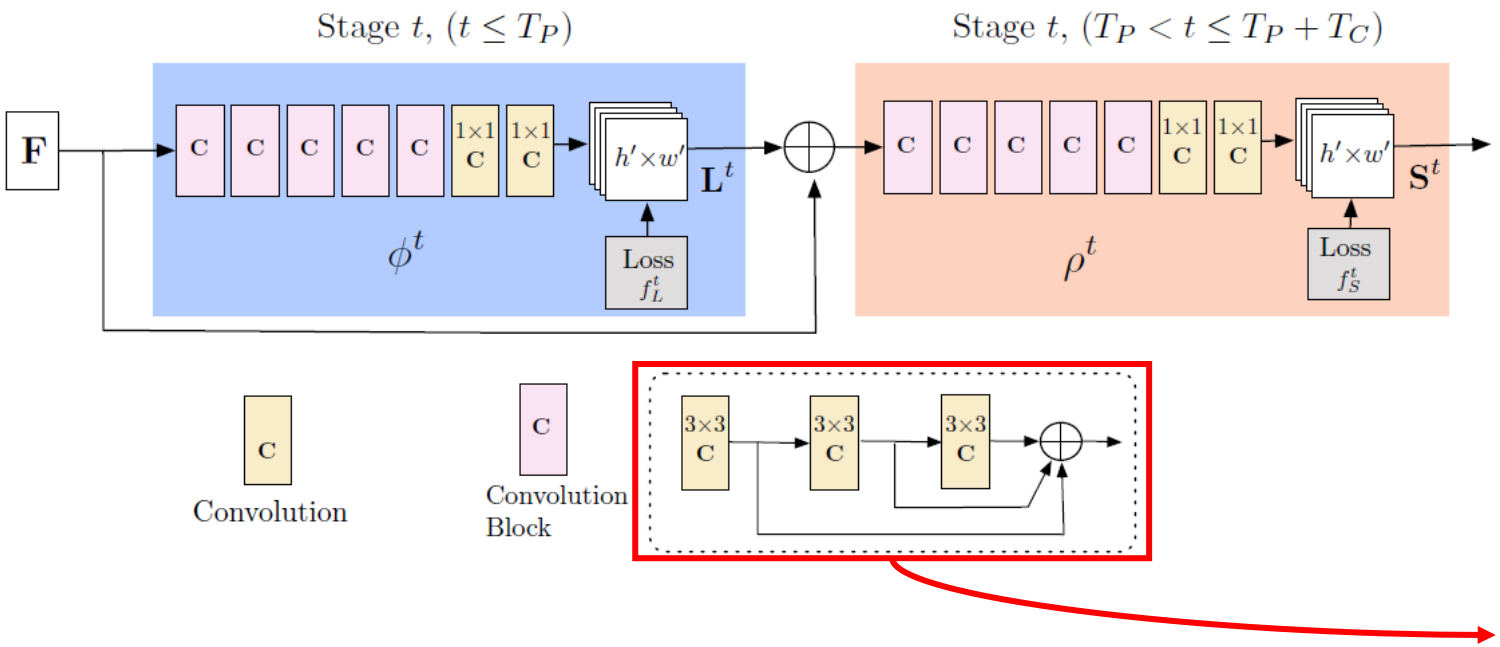
Method



- Input image (size $w \times h$)
- Predict a set of 2D confidence maps S and a set of 2D vector fields L of PAFs
- Confidence maps and the PAFs are parsed by greedy inference (Bipartite Matching)
- Produce 2D locations of anatomical keypoints for each person

$$S = (S_1, S_2, \ldots, S_J), \quad \text{where } S_j \in \mathbb{R}^{w \times h}, \; j \in \{1, \ldots, J\}$$
$$L = (L_1, L_2, \ldots, L_c), \quad \text{where } L_c \in \mathbb{R}^{w \times h \times 2}, \; c \in \{1, \ldots, C\}$$

Architecture



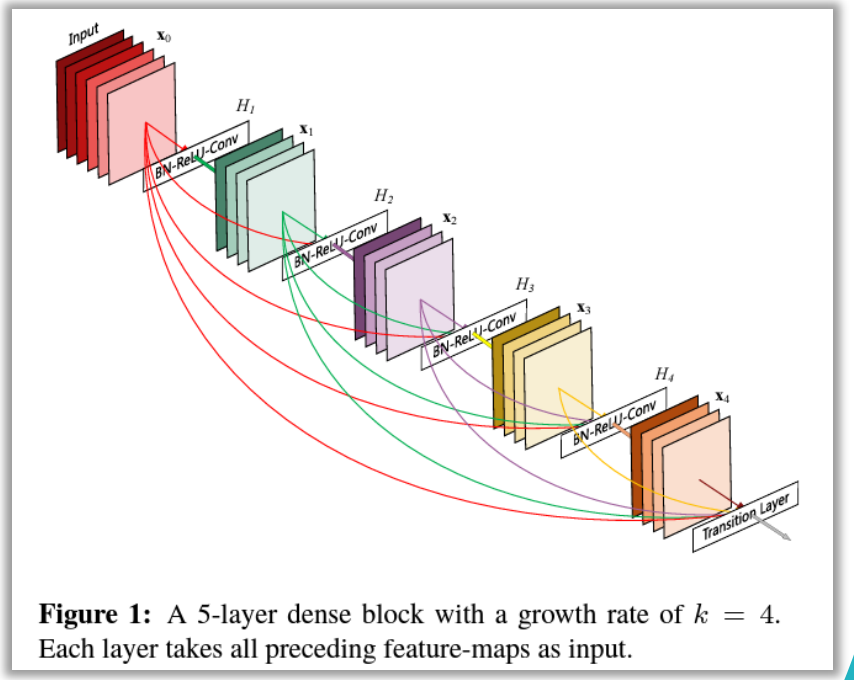


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

- Iteratively predicts PAFs that encode part-to-part association
- Receptive field is preserved while the computation is reduced, by replacing each 7x7 convolutional kernel by 3 consecutive 3x3 kernels
- Output of each one of the 3 convolutional kernels is concatenated following an approach similar to DenseNet

Gao Huang, Zhuang Liu, Laurens Van der Maaten. 「***Densely Connected Convolution Layers***」*, 2018*

Simultaneous Detection and Association

- $L^1 = \phi^1(F)$
- $L^t = \phi^t(F, L^{t-1}), \quad \forall 2 \leq t \leq T_P$
- $S^{T_P} = \rho^t(F, L^{T_P}), \ \forall t = T_P$
- $S^t = \rho^t(F, L^{T_P}, S^{t-1}), \ \forall T_P < t \leq T_P + T_C$

ϕ^t : CNNs for inference stage at stage t
T_P : Number of total PAF stages
ρ^t : CNNs for inference stage at stage t
T_c : Number of total confidence map stage

Loss Function

If there are no other information in the bunch of body parts,
it is impossible to parse them into different people

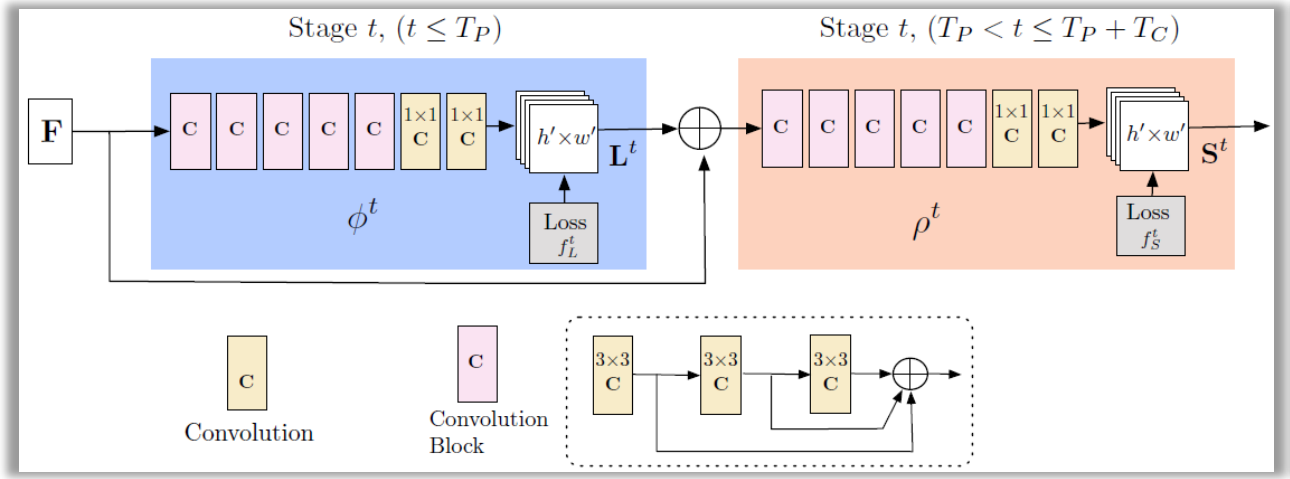
Apply loss function at the end of each stage to guide network to iteratively predict PAFs of body parts in the first branch and confidence maps in the second branch

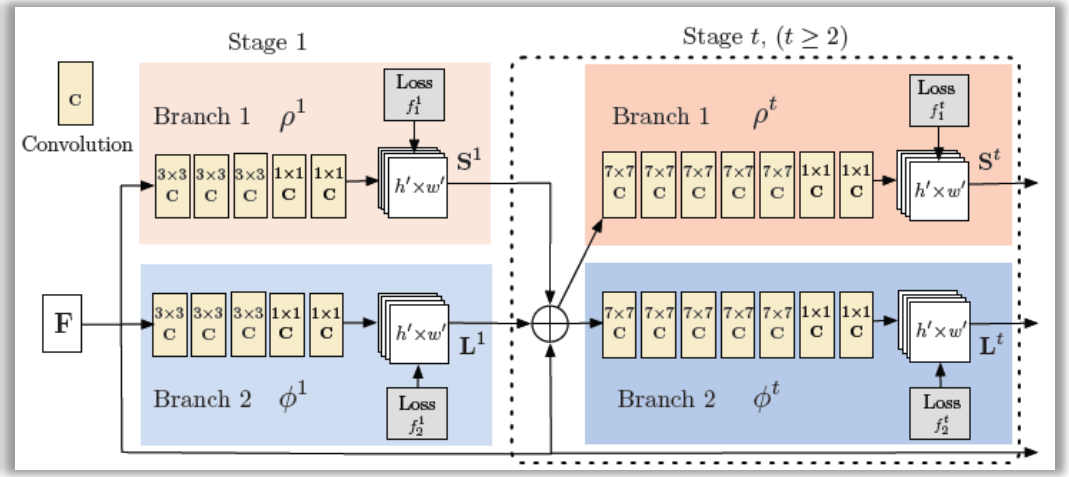
$$f_L^{t_i} = \sum_{c=1}^{C} \sum_{p} W(p) \cdot \left\| L_c^{t_i}(p) - L_c^*(p) \right\|_2^2$$

$$f_S^{t_k} = \sum_{j=1}^{J} \sum_{p} W(p) \cdot \left\| S_j^{t_k}(p) - S_j^*(p) \right\|_2^2$$

$$f = \sum_{t=1}^{T_P} f_L^t + \sum_{t=T_P+1}^{T_P+T_C} f_S^t$$

Realtime Multi-Person 2D Pose Estimation





[Difference]

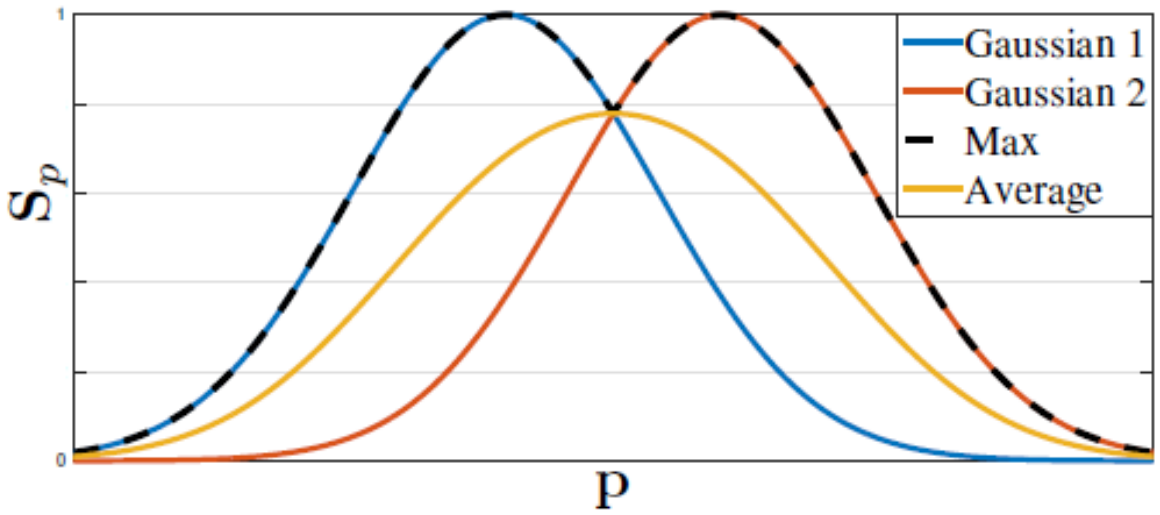
- Convolutions of kernel size 7 are replaced with 3 layers of convolutions of kernel 3
- Inference stage
  - $L^t = \phi^t(F, L^{t-1}), \quad \forall 2 \leq t \leq T_P$
  - $S^t = \rho^t(F, L^{T_P}, S^{t-1}), \quad \forall T_P < t \leq T_P + T_C$
  - $L^t = \phi^t(F, S^{t-1}, L^{t-1}), \quad \forall t \geq 2$
  - $S^t = \rho^t(F, S^{t-1}, L^{t-1}), \quad \forall t \geq 2$
- Loss Function

$$f = \sum_{t=1}^{T_P} f_L^t + \sum_{t=T_P+1}^{T_P+T_C} f_S^t \qquad f = \sum_{t=1}^{T} (f_S^t + f_L^t)$$

- Parallel → Serial ⇒ ½ computation cost per stage

Zhe Cao, Tomas Simon, Shih-En Wei, Yaser Sheikh. 「***Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields***」, *2017*

Confidence Maps for Part Detection



$$S^*_{j,k}(p) = \exp\left(-\frac{\|p - x_{j,k}\|_2^2}{\sigma^2}\right)$$

$$S^*_j(p) = \max_k S^*_{j,k}(p)$$

σ : controls spread of the peak
$S^*_{j,k}(p)$: confidence map for each person k
$S^*_j(p)$: aggregation of individual confidence maps

Predict confidence maps and obtain body part candidates by performing NMS

Part Affinity Fields for Part Association

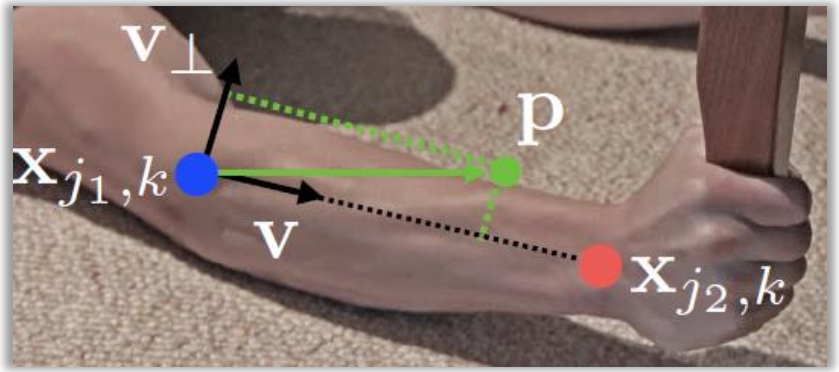
(a) (b) (c)

Fig. 5: Part association strategies. (a) The body part detection candidates (red and blue dots) for two body part types and all connection candidates (grey lines). (b) The connection results using the midpoint (yellow dots) representation: correct connections (black lines) and incorrect connections (green lines) that also satisfy the incidence constraint. (c) The results using PAFs (yellow arrows). By encoding position and orientation over the support of the limb, PAFs eliminate false associations.

How do we assemble detected body parts to form the full-body poses of an unknown number of people?

Detect an additional midpoint between each pair of parts on a limb and check for its incidence between candidate part detections

Part Affinity Fields for Part Association



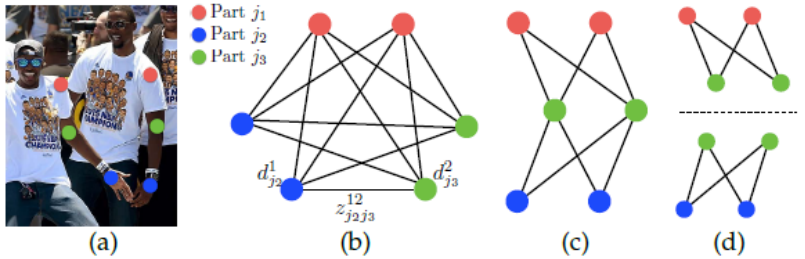


Fig. 6: Graph matching. (a) Original image with part detections. (b) K-partite graph. (c) Tree structure. (d) A set of bipartite graphs.

$$L^*_{c,k}(p) = \begin{cases} v & if\ p\ on\ limb\ c,k \\ 0 & otherwise \end{cases} \qquad v = (x_{j_2,k} - x_{j_1,k}) / \left\| x_{j_2,k} - x_{j_1,k} \right\|_2$$

$$0 \le v \cdot (p - x_{j_1,k}) \le l_{c,k} \text{ and } \left| v_\perp \cdot (p - x_{j_1,k}) \right| \le \sigma_l \ , l_{c,k} = \left\| x_{j_2,k} - x_{j_1,k} \right\|_2$$

$$L^*_c(p) = \frac{1}{n_c(p)} \sum_k L^*_{c,k}(p)$$

$$E = \int_{u=0}^{u=1} L_c(p(u)) \cdot \frac{d_{j_2} - d_{j_1}}{\left\| d_{j_2} - d_{j_1} \right\|} du\,, \qquad p(u) = (1-u) d_{j_1} + u d_{j_2}$$

E_{mn} : Part Affinity between $d^m_{j_1}$, $d^n_{j_2}$

PAFs preserve both location and orientation information across the region of support of the limb

Multi-Person Parsing using PAFs

$$\max_{\mathcal{Z}_c} E_c = \max_{\mathcal{Z}_c} \sum_{m \in \mathcal{D}_{j_1}} \sum_{n \in \mathcal{D}_{j_2}} E_{mn} \cdot z_{j_1 j_2}^{mn}$$

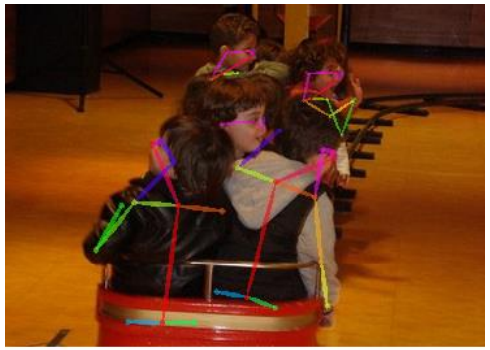
$$\text{s.t. } \forall m \in \mathcal{D}_{j_1}, \sum_{n \in \mathcal{D}_{j_2}} z_{j_1 j_2}^{mn} \leq 1, \ \forall n \in \mathcal{D}_{j_2}, \ \sum_{m \in \mathcal{D}_{j_1}} z_{j_1 j_2}^{mn} \leq 1$$

$$\max_{\mathcal{Z}} E = \sum_{c=1}^{C} \max_{\mathcal{Z}_c} E_c$$

⇒ Find a matching with maximum weight for the chosen edges

- Obtain limb connection candidates for each limb type independently
- Can assemble the connections that share the same part detection candidates into full-body poses of multiple structure
- Incorporate redundant PAF connections by sorting all pairwise possible connections based on their PAF score
  → Improves the accuracy in crowded images

(a) Original Person Parsing



(b) PAF-Redundant Parsing

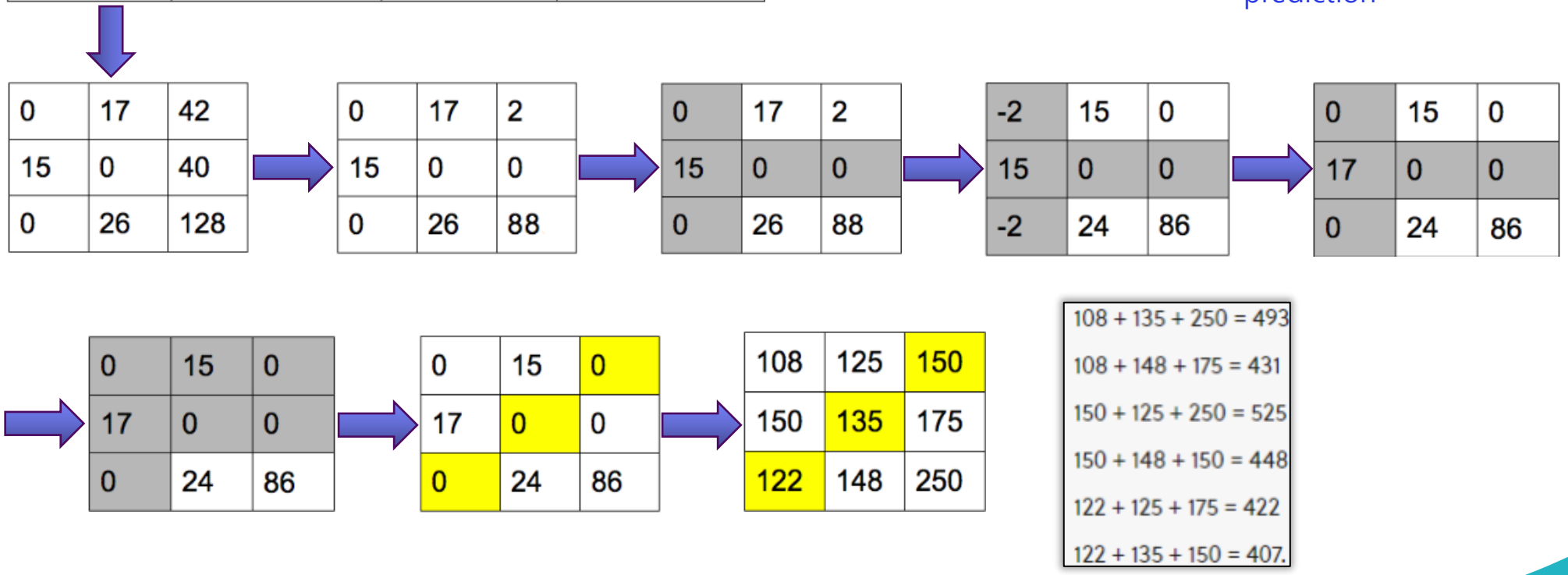
Multi-Person Parsing using PAFs

Hungarian Algorithm

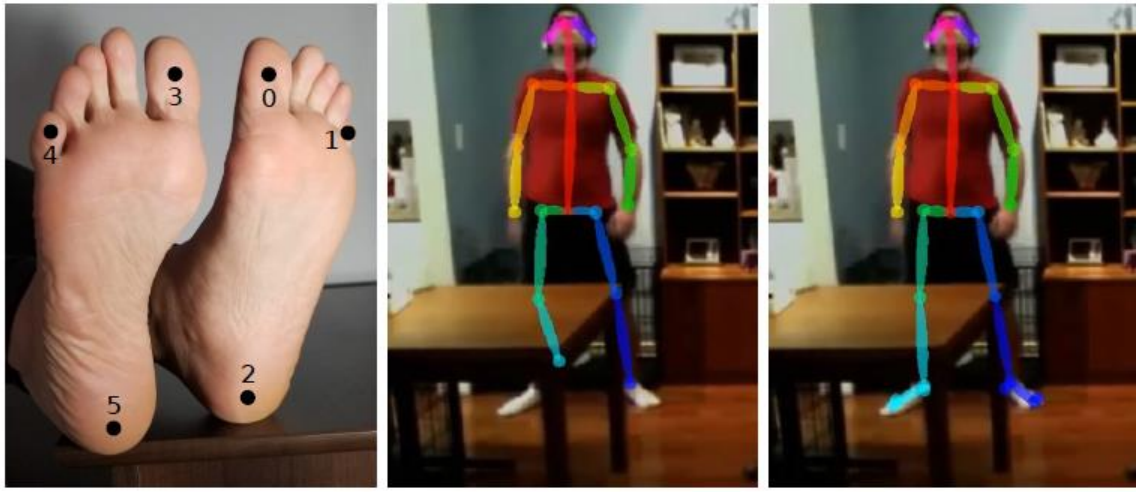
Company	Cost for Musician	Cost for Chef	Cost for Cleaners
Company A	$108	$125	$150
Company B	$150	$135	$175
Company C	$122	$148	$250

Find bipartite matching between ground-truth and prediction
→ enforce permutation-invariance and guarantee unique match

$$\mathcal{L}_{hungarian}(y, \hat{y}) = \sum_{i=1}^{N} \left[\underbrace{-\log \hat{p}_{\hat{\sigma}(i)}(c_i)}_{\text{Class prediction}} + \underbrace{\mathbb{I}_{\{c_i \neq \emptyset\}} \mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}}(i))}_{\text{Box loss}} \right]$$

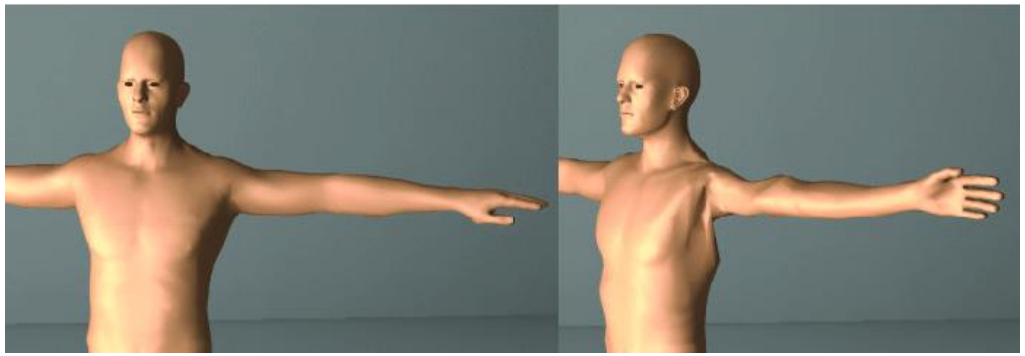


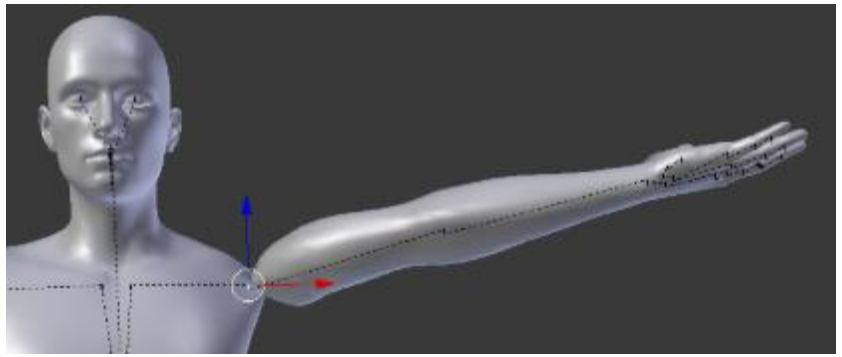
Extended Foot Keypoint Detection



(a) (b) (c)

“Candy wrapper effect”





Dataset

(a) MPII (b) COCO (c) COCO+Foot

MPII human multi-person dataset

- 3,844 training images
- 1,758 test images
- Groups of multiple interacting individuals in highly articulated poses with 14 body parts

COCO keypoint challenge dataset

- Require simultaneously detecting people and localizing 17 keypoints in each person (12 human body parts + 5 facial keypoints)

Foot dataset

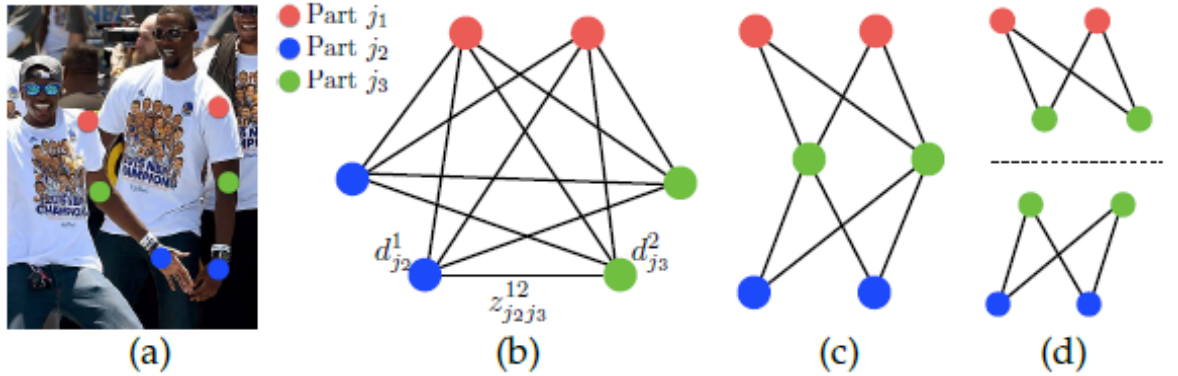
- Subset of 15K annotations out of the COCO keypoint dataset
- Collect images in diverse scenarios that contain many real-world challenges

Result

Method	Hea	Sho	Elb	Wri	Hip	Kne	Ank	mAP	s/image
Subset of 288 images as in [1]									
Deepcut [1]	73.4	71.8	57.9	39.9	56.7	44.0	32.0	54.1	57995
Iqbal et al. [41]	70.0	65.2	56.4	46.1	52.7	47.9	44.5	54.7	10
DeeperCut [2]	87.9	84.0	71.9	63.9	68.8	63.8	58.1	71.2	230
Newell et al. [48]	91.5	87.2	75.9	65.4	72.2	67.0	62.1	74.5	-
ArtTrack [47]	92.2	**91.3**	80.8	71.4	**79.1**	72.6	67.8	**79.3**	**0.005**
Fang et al. [6]	89.3	88.1	80.7	75.5	73.7	**76.7**	**70.0**	79.1	-
Ours	**92.9**	**91.3**	**82.3**	72.6	76.0	70.9	66.8	79.0	**0.005**
Full testing set									
DeeperCut [2]	78.4	72.5	60.2	51.0	57.2	52.0	45.4	59.5	485
Iqbal et al. [41]	58.4	53.9	44.5	35.0	42.2	36.7	31.1	43.1	10
Levinko et al. [71]	89.8	85.2	71.8	59.6	71.1	63.0	53.5	70.6	-
ArtTrack [47]	88.8	87.0	75.9	64.9	74.2	68.8	60.5	74.3	**0.005**
Fang et al. [6]	88.4	86.5	78.6	**70.4**	74.4	**73.0**	**65.8**	76.7	-
Newell et al. [48]	**92.1**	89.3	78.9	69.8	76.2	71.6	64.7	77.5	-
Fieraru et al. [72]	91.8	**89.5**	**80.4**	69.6	**77.3**	71.7	65.5	**78.0**	-
Ours (one scale)	89.0	84.9	74.9	64.2	71.0	65.6	58.1	72.5	0.005
Ours	91.2	87.6	77.7	66.8	75.4	68.9	61.7	75.6	**0.005**

TABLE 1: Results on the MPII dataset. Top: Comparison results on the testing subset defined in [1]. Middle: Comparison results on the whole testing set. Testing without scale search is denoted as "(one scale)".

Team	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}
Top-Down Approaches					
Megvii [43]	78.1	94.1	85.9	74.5	83.3
MRSA [44]	76.5	92.4	84.0	73.0	82.7
The Sea Monsters*	75.9	92.1	83.0	71.7	82.1
Alpha-Pose [6]	71.0	87.9	77.7	69.0	75.2
Mask R-CNN [5]	69.2	90.4	76.0	64.9	76.3
Bottom-Up Approaches					
METU [50]	70.5	87.7	77.2	66.1	77.3
TFMAN*	70.2	89.2	77.0	65.6	76.3
PersonLab [49]	68.7	89.0	75.4	64.1	75.5
Associative Emb. [48]	65.5	86.8	72.3	60.6	72.6
Ours	64.2	86.2	70.1	61.0	68.8
Ours [3]	61.8	84.9	67.5	57.1	68.2



Method	Hea	Sho	Elb	Wri	Hip	Kne	Ank	mAP	s/image
Fig. 6b	91.8	**90.8**	80.6	69.5	78.9	71.4	63.8	78.3	362
Fig. 6c	92.2	90.8	80.2	69.2	78.5	70.7	62.6	77.6	43
Fig. 6d	92.0	90.7	80.0	69.4	78.4	70.1	62.3	77.4	0.005
Fig. 6d (sep)	**92.4**	90.4	**80.9**	**70.8**	**79.5**	**73.1**	**66.5**	**79.1**	**0.005**

Method	AP	AP^{50}	AP^{75}	AP^{M}	AP^{L}	Stages
5 PAF - 1 CM	**65.3**	85.2	71.3	62.2	**70.7**	6
4 PAF - 2 CM	65.2	**85.3**	**71.4**	62.3	70.1	6
3 PAF - 3 CM	65.0	85.1	71.2	**62.4**	69.4	6
4 PAF - 1 CM	64.8	**85.3**	70.9	61.9	69.6	5
3 PAF - 1 CM	64.6	84.8	70.6	61.8	69.5	4
3 CM - 3 PAF	61.0	83.9	65.7	58.5	65.3	6

TABLE 5: Self-comparison experiments on the COCO validation set. CM refers to confidence map, while the numbers express the number of estimation stages for PAF and CM. *Stages* refers to the number of PAF and CM stages. Reducing the number of stages increases the runtime performance.

Result

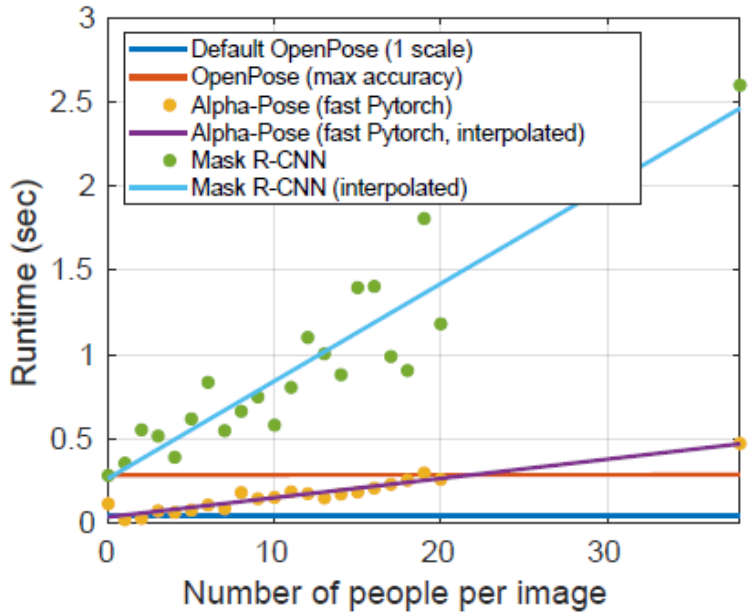


Fig. 12: Inference time comparison between OpenPose, Mask R-CNN, and Alpha-Pose (fast Pytorch version). While OpenPose inference time is invariant, Mask R-CNN and Alpha-Pose runtimes grow linearly with the number of people. Testing with and without scale search is denoted as "max accuracy" and "1 scale", respectively. This analysis was performed using the same images for each algorithm and a batch size of 1. Each analysis was repeated 1000 times and then averaged. This was all performed on a system with a Nvidia 1080 Ti and CUDA 8.

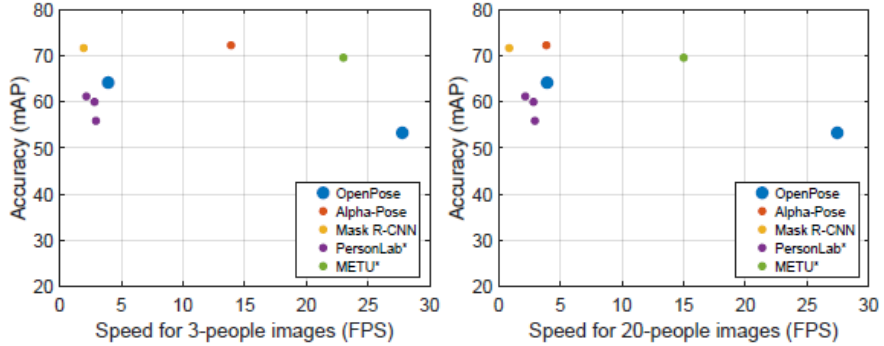


Fig. 13: Trade-off between speed and accuracy for the main entries of the COCO Challenge. We only consider those approaches that either release their runtime measurements (methods with an asterisk) or their code (rest). Algorithms with several values represent different resolution configurations. AlphaPose, METU, and single-scale OpenPose provide the best results considering the trade-off between speed and accuracy. The remaining methods are both slower and less accurate than at least one of these 3 approaches.

Method	CUDA	CPU-only
Original MPII model	73 ms	**2309 ms**
Original COCO model	74 ms	2407 ms
Body+foot model	**36 ms**	10396 ms

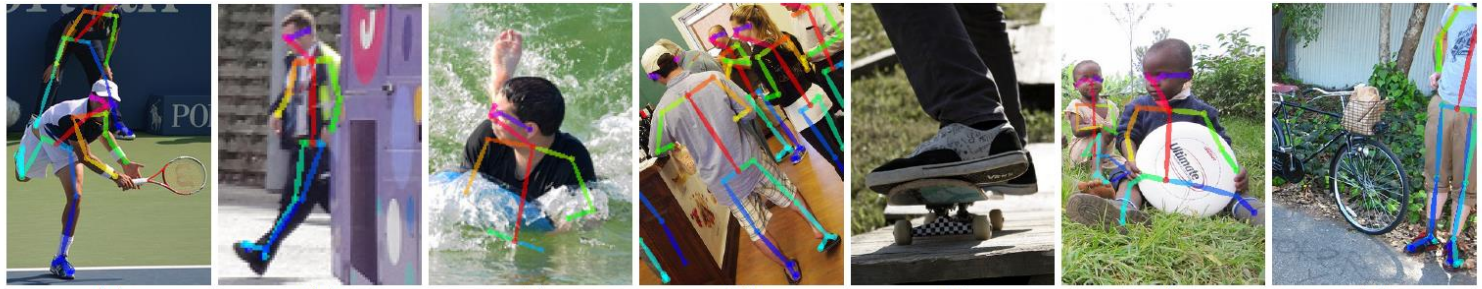
TABLE 6: Runtime difference between the 3 models released in OpenPose with CUDA and CPU-only versions, running in a NVIDIA GeForce GTX-1080 Ti GPU and a i7-6850K CPU. MPII and COCO models refer to our work in [3].

Result

Fig. 14: Vehicle keypoint detection examples from the validation set. The keypoint locations are successfully estimated under challenging scenarios, including overlapping between cars, cropped vehicles, and different scales.

(a) (b) (c) (d) (e) (f)

Fig. 15: Common failure cases: (a) rare pose or appearance, (b) missing or false parts detection, (c) overlapping parts, i.e., part detections shared by two persons, (d) wrong connection associating parts from two persons, (e-f): false positives on statues or animals.

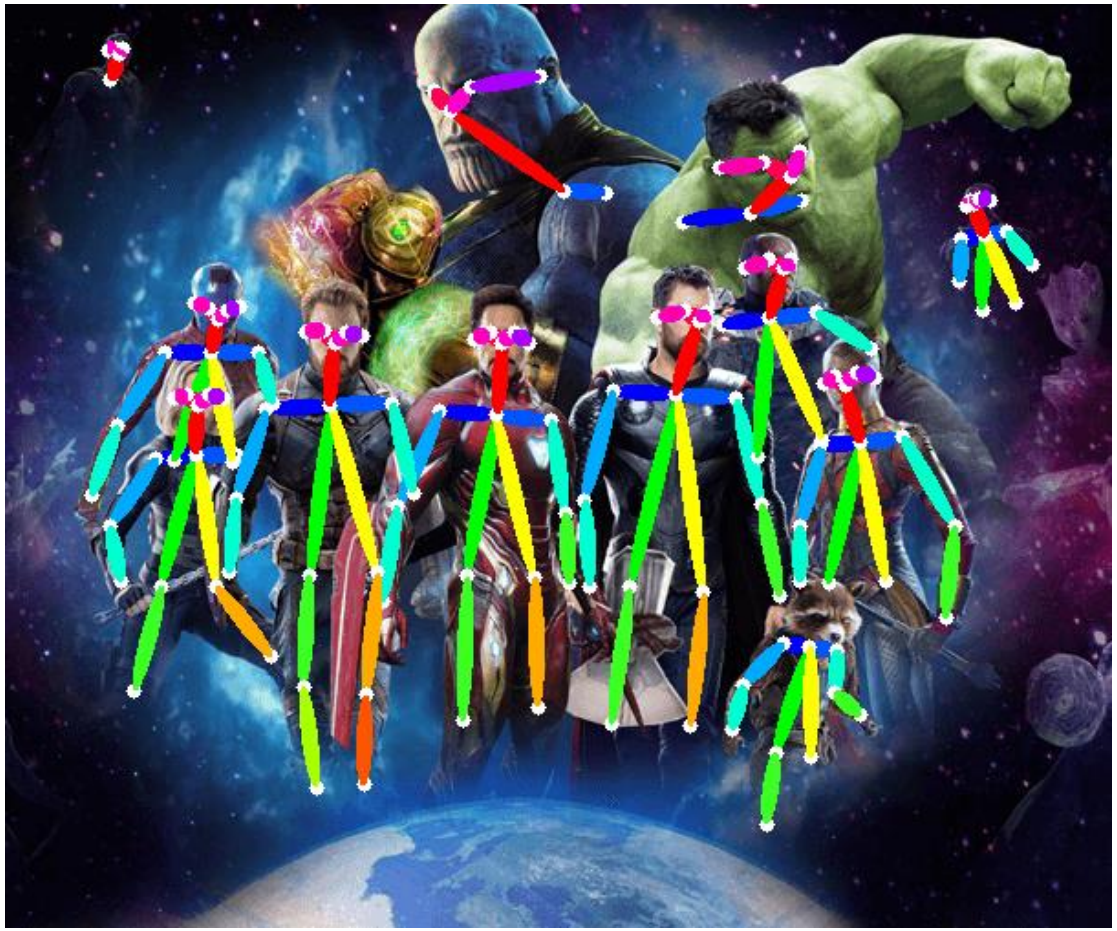


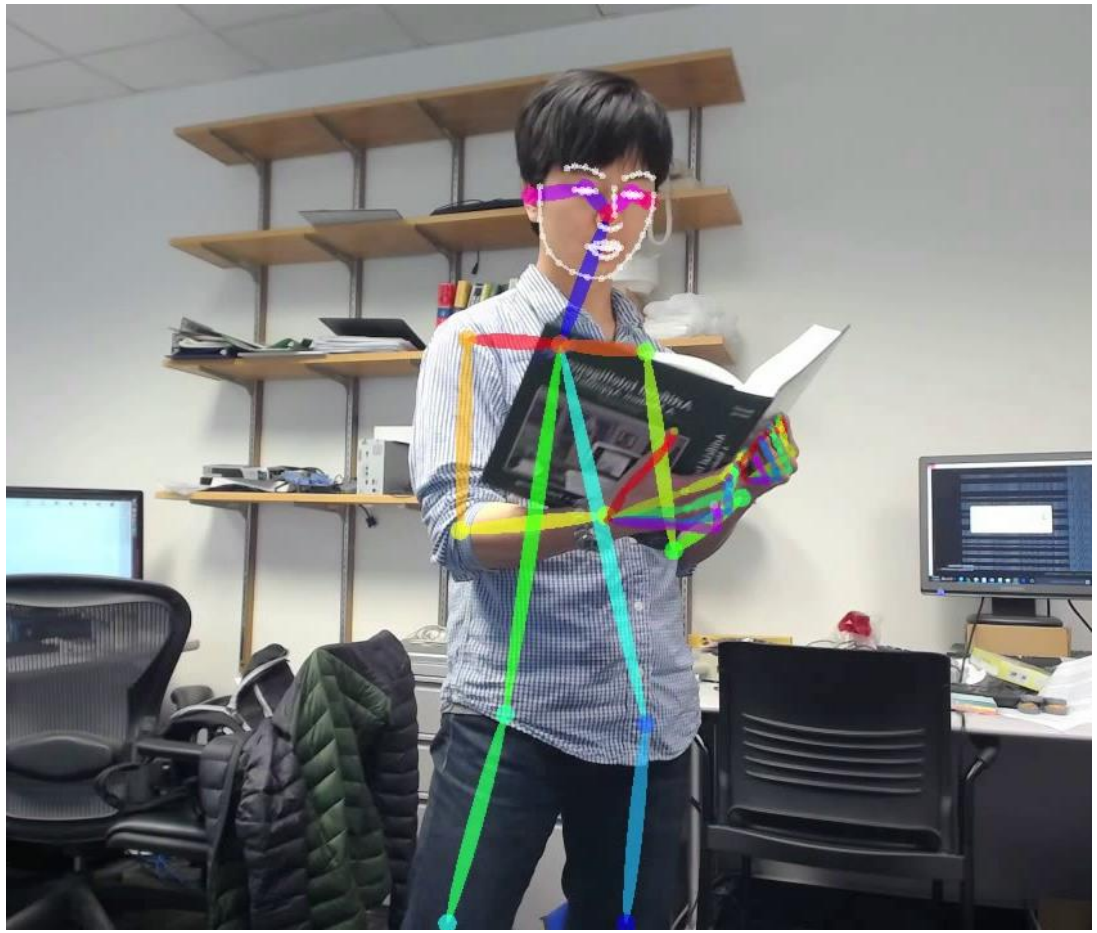
(a) (b) (c) (d) (e) (f) (g)

Fig. 16: Common foot failure cases: (a) foot or leg occluded by the body, (b) foot or leg occluded by another object, (c) foot visible but leg occluded, (d) shoe and foot not aligned, (e): false negatives when foot visible but rest of the body occluded, (f): soles of their feet are usually not detected (rare in training), (g): swap between right and left body parts.

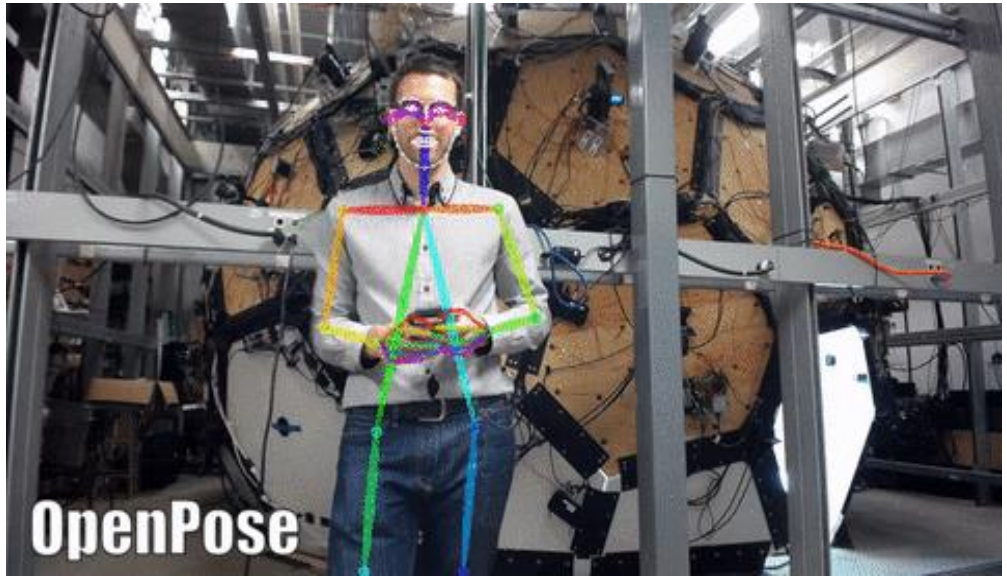
Result

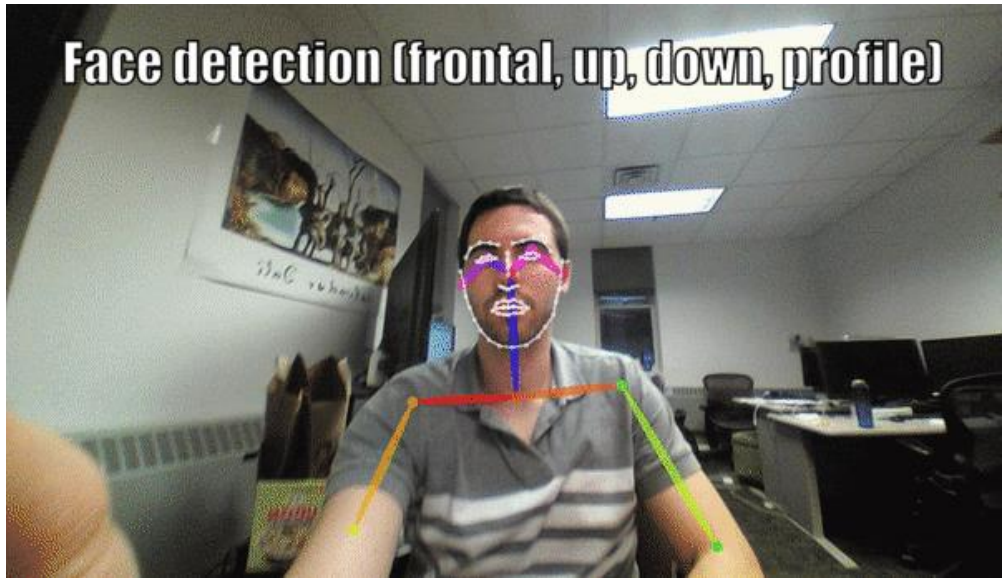
Result





Result

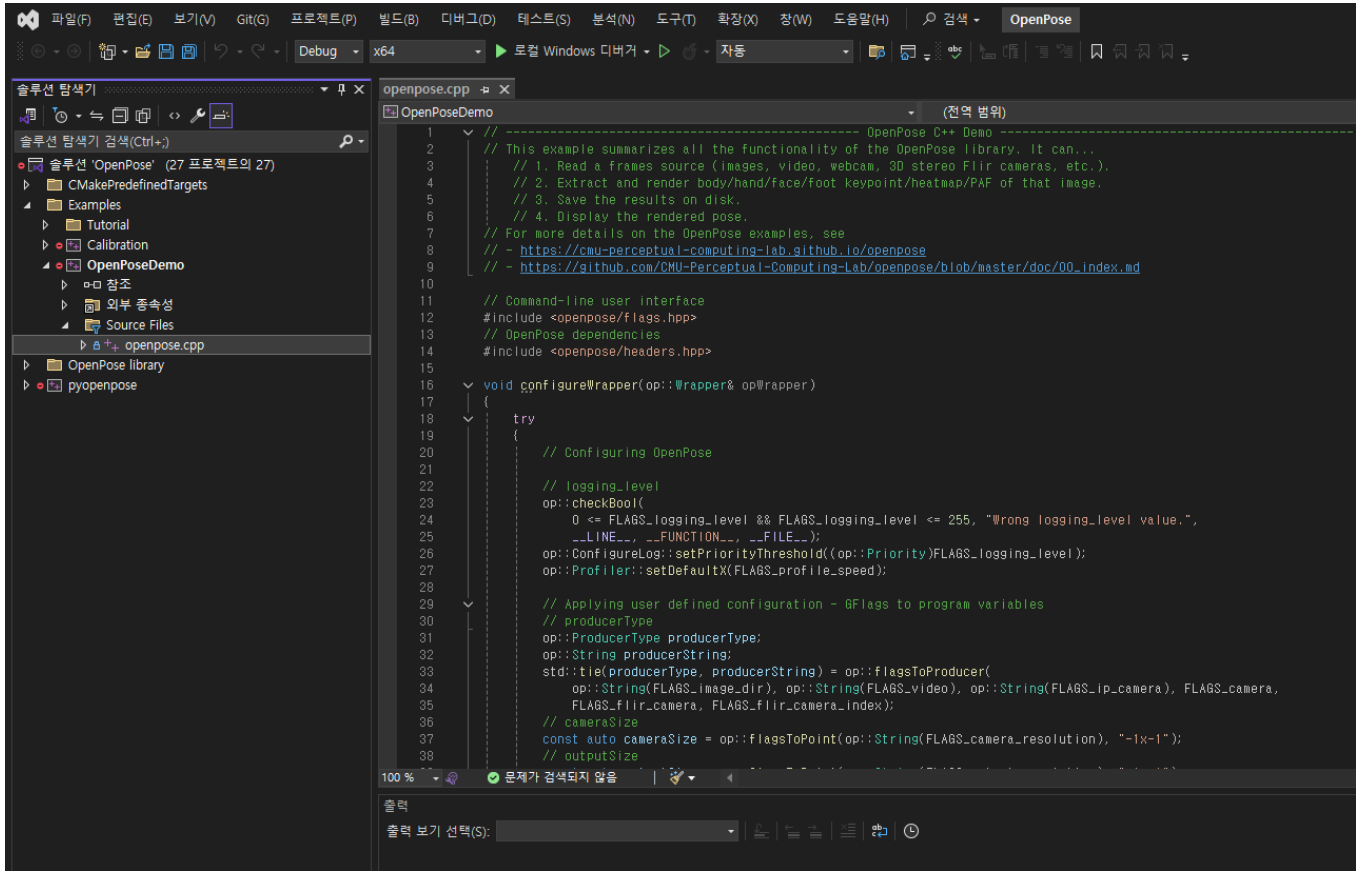

OpenPose

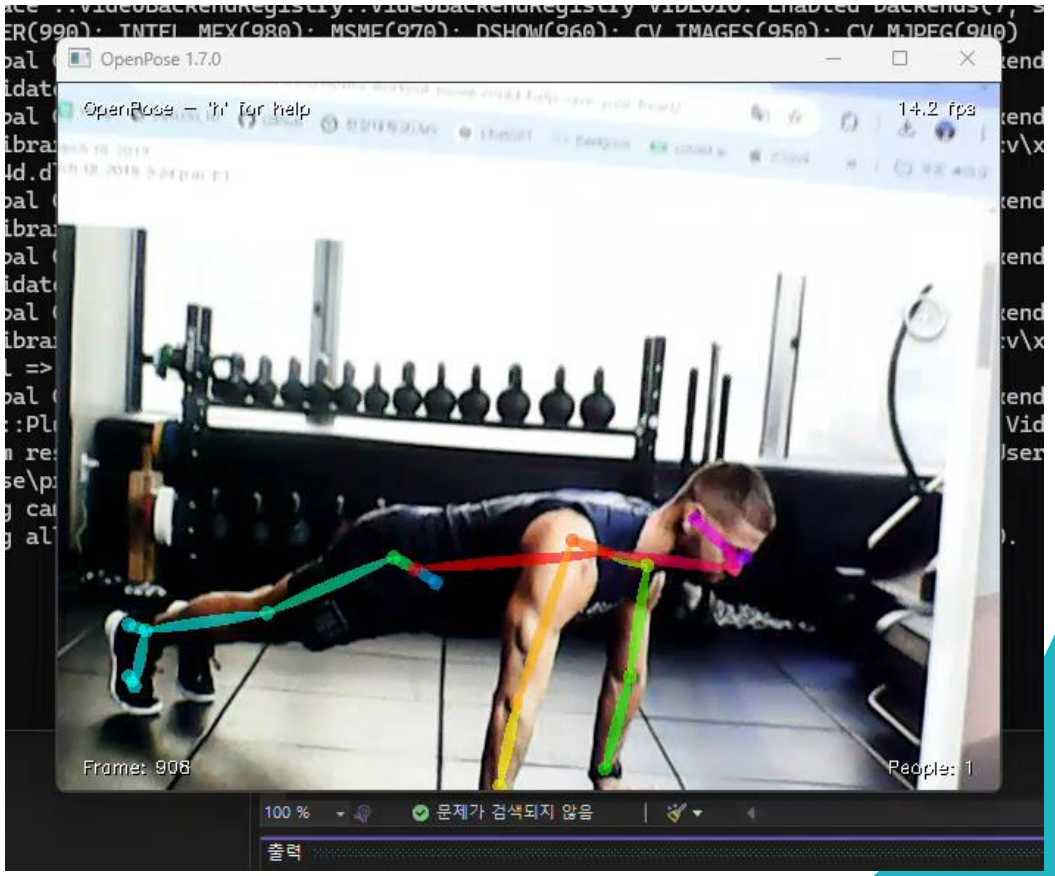

Face detection (frontal, up, down, profile)


8.8 fps
Realtime Detections

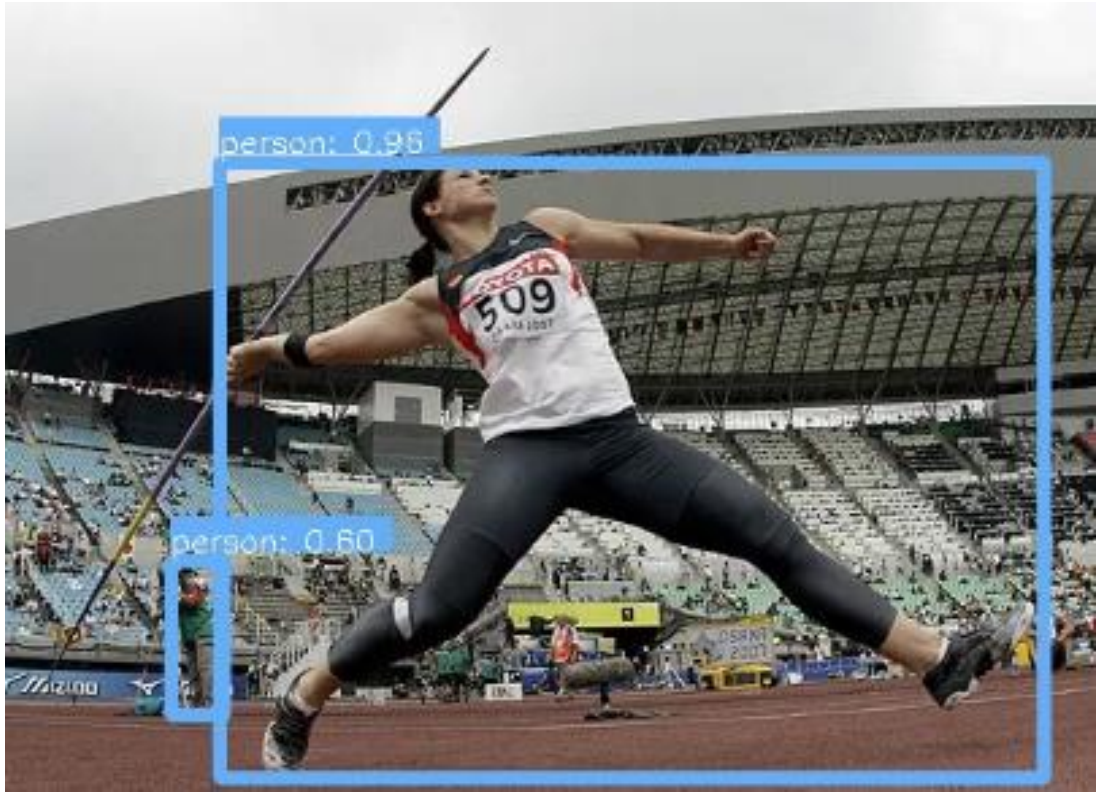
Implementation

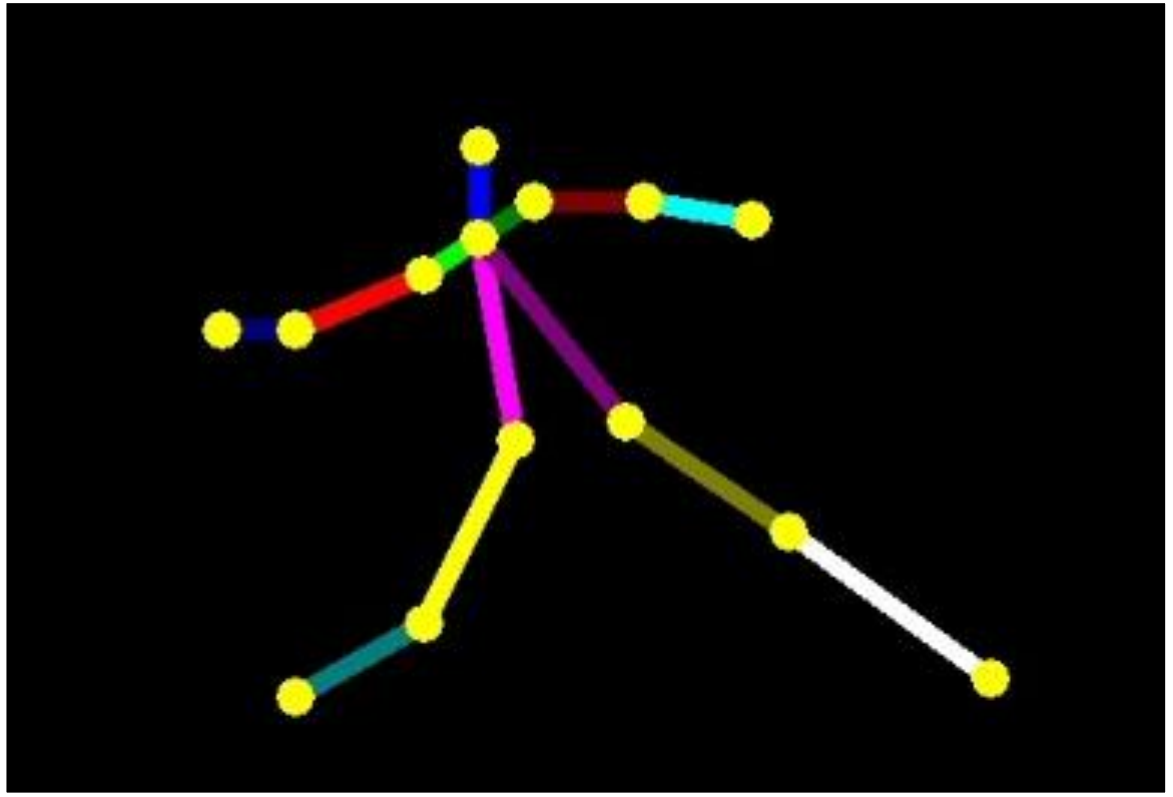
https://github.com/CMU-Perceptual-Computing-Lab/openpose





Implementation

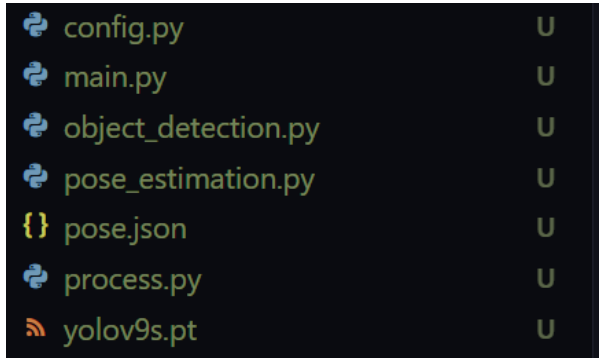




Implementation

```
{
  "BODY_PARTS": {
      "Head": 0,
      "Neck": 1,
      "RShoulder": 2,
      "RElbow": 3,
      "RWrist": 4,
      "LShoulder": 5,
      "LElbow": 6,
      "LWrist": 7,
      "RHip": 8,
      "RKnee": 9,
      "RAnkle": 10,
      "LHip": 11,
      "LKnee": 12,
      "LAnkle": 13
  },
  "POSE_PAIRS": [
      ["Head", "Neck"],
      ["Neck", "RShoulder"],
      ["RShoulder", "RElbow"],
      ["RElbow", "RWrist"],
      ["Neck", "LShoulder"],
      ["LShoulder", "LElbow"],
      ["LElbow", "LWrist"],
      ["Neck", "RHip"],
      ["RHip", "RKnee"],
      ["RKnee", "RAnkle"],
      ["Neck", "LHip"],
      ["LHip", "LKnee"],
      ["LKnee", "LAnkle"]
  ],
  "colors": [
      [255, 0, 0],
      [0, 255, 0],
      [0, 0, 255],
      [128, 0, 0],
      [0, 128, 0],
      [0, 0, 128],
      [255, 255, 0],
      [255, 0, 255],
      [0, 255, 255],
      [128, 128, 0],
      [128, 0, 128],
      [0, 128, 128],
      [255, 255, 255],
      [128, 128, 128]
  ],
  "pose_protoFile": "models/pose/coco/pose_deploy_linevec.prototxt",
  "pose_weightsFile": "models/pose/coco/pose_iter_440000.caffemodel",

  "face_protoFile": "models/face/pose_deploy.prototxt",
  "face_weightsFile": "models/face/pose_iter_116000.caffemodel",

  "hand_protoFile": "models/hand/pose_deploy.prototxt",
  "hand_weightsFile": "models/hand/pose_iter_102000.caffemodel",

  "image_path": "/mnt/storage1/Hyunjoon/openpose/input/image",
  "video_path": "/mnt/storage1/Hyunjoon/openpose/input/video",

  "output_image_path": "/mnt/storage1/Hyunjoon/openpose/output/image",
  "output_video_path": "/mnt/storage1/Hyunjoon/openpose/output/video"
}
```

PoseEstimation.py →

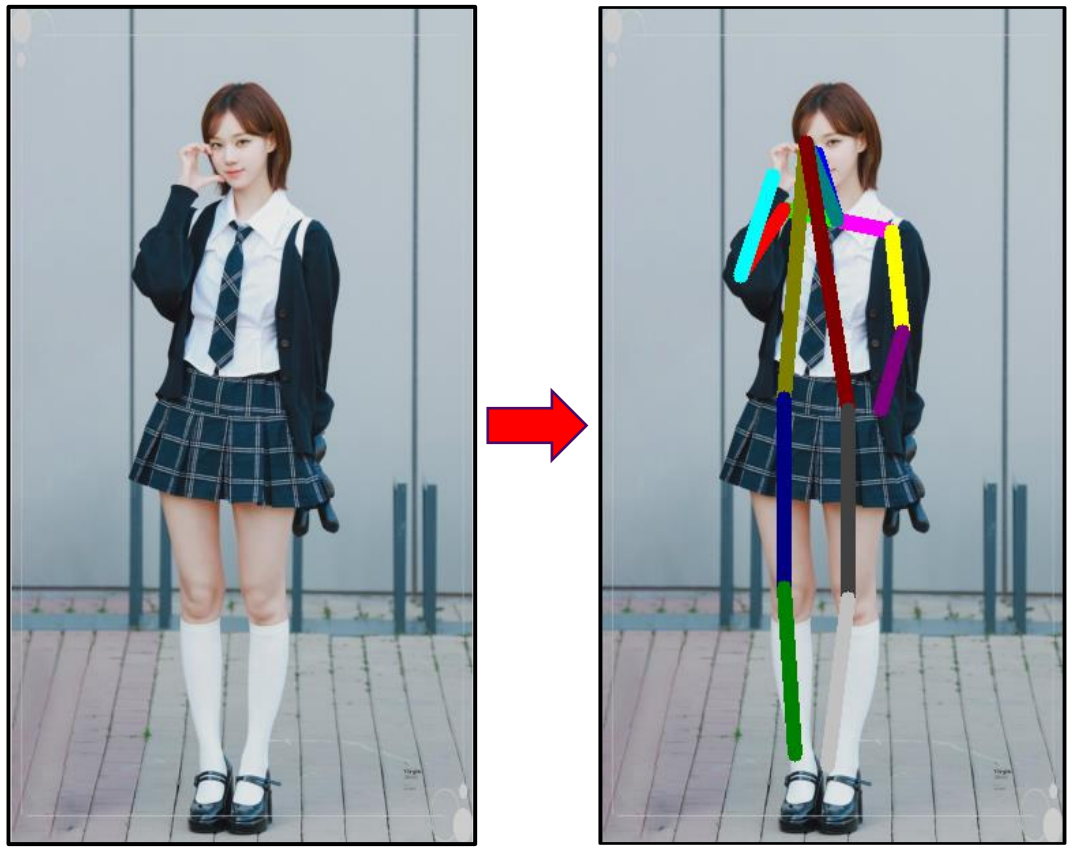


← pose.json

{.json}

```python
import os
import argparse
import cv2
import yaml

with open('pose.yaml', 'r') as file:
    config = yaml.safe_load(file)

BODY_PARTS = config['BODY_PARTS']
POSE_PAIRS = config['POSE_PAIRS']
colors = config['colors']
protoFile = config['protoFile']
weightsFile = config['weightsFile']

net = cv2.dnn.readNetFromCaffe(protoFile, weightsFile)

class PoseEstimation:
    def __init__(self):
        pass

    def process_image(self, image, output_path, image_name):
        imageHeight, imageWidth, _ = image.shape

        inpBlob = cv2.dnn.blobFromImage(image, 1.0/255, (imageWidth, imageHeight), (0, 0, 0), swapRB=False, crop=False)

        net.setInput(inpBlob)
        output = net.forward()

        H = output.shape[2]
        W = output.shape[3]
        print(f"Image: {image_name}, ID: {len(output[0])}, H: {H}, W: {W}")

        points = []
        for i in range(len(BODY_PARTS)):
            probMap = output[0, i, :, :]
            minVal, prob, minLoc, point = cv2.minMaxLoc(probMap)

            x = (imageWidth * point[0]) / W
            y = (imageHeight * point[1]) / H

            if prob > 0.1 :
                points.append((int(x), int(y)))
            else :
                points.append(None)

        imgcpy = image

        for idx, pair in enumerate(POSE_PAIRS):
            partA = pair[0]             # Head
            partA = BODY_PARTS[partA]   # 0
            partB = pair[1]             # Neck
            partB = BODY_PARTS[partB]   # 1

            if points[partA] and points[partB]:
                color = colors[idx % len(colors)]
                cv2.line(imgcpy, points[partA], points[partB], color, 2)

        cv2.imwrite(output_path, imgcpy)

def main():
    parser = argparse.ArgumentParser(description="Pose Estimation using OpenPose model")
    parser.add_argument('--source', type=str, required=True, help="Name of the input file (image or video)")
    parser.add_argument('--output', type=str, required=True, help="Name of the output file (image or video)")

    args = parser.parse_args()

    input_folder = "/mnt/storage1/Hyunjoon/openpose/input"
    output_folder = "/mnt/storage1/Hyunjoon/openpose/output"

    source_path = os.path.join(input_folder, args.source)
    output_path = os.path.join(output_folder, args.output)

    PE = PoseEstimation()

    if os.path.isfile(source_path):
        filename = os.path.basename(source_path)
        image = cv2.imread(source_path)
        PE.process_image(image, output_path, filename)
    else:
        print(f"Source path {source_path} does not exist or is not a file.")

if __name__ == "__main__":
    main()
```

Implementation

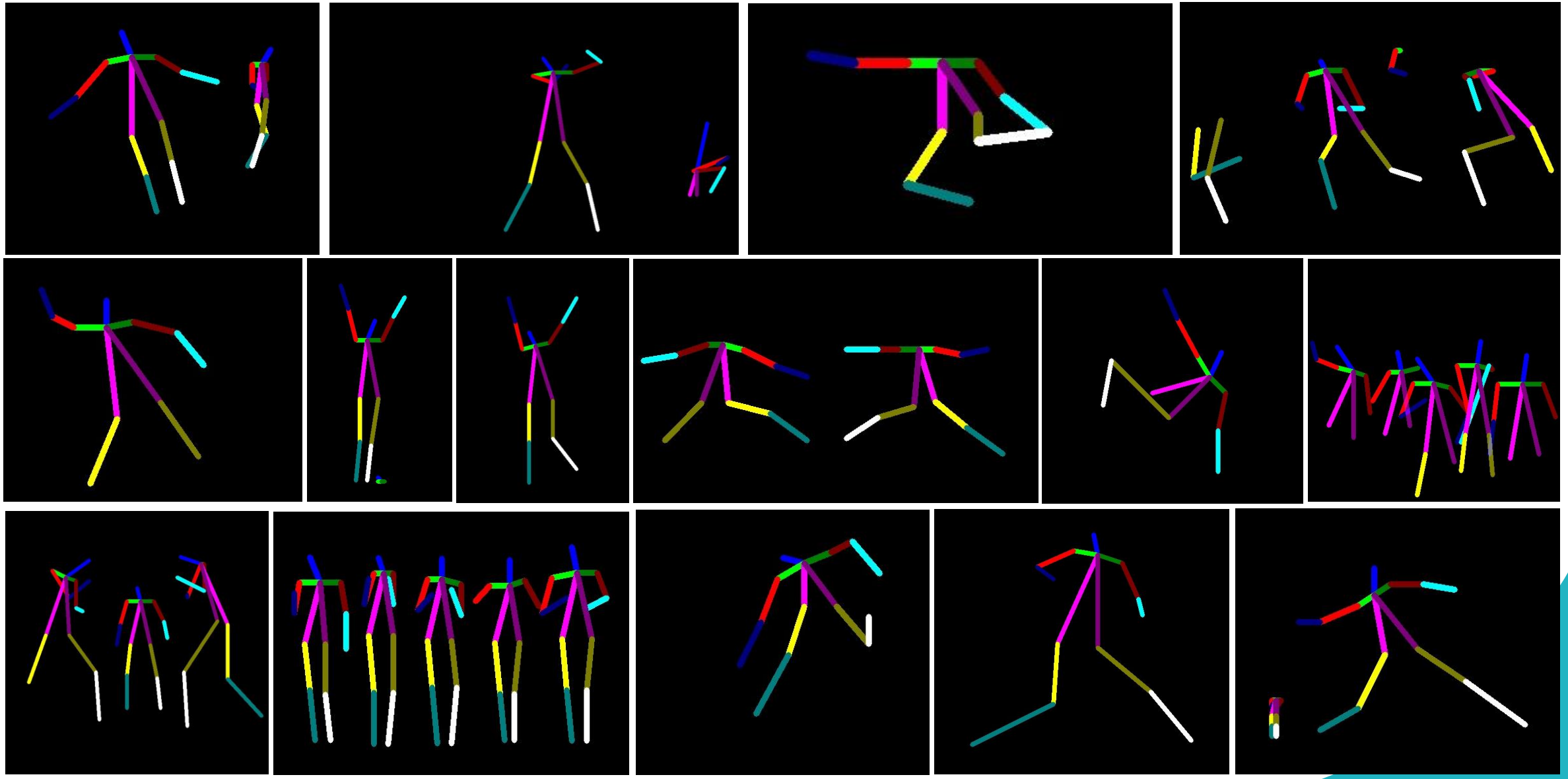


Only Pose Estimation

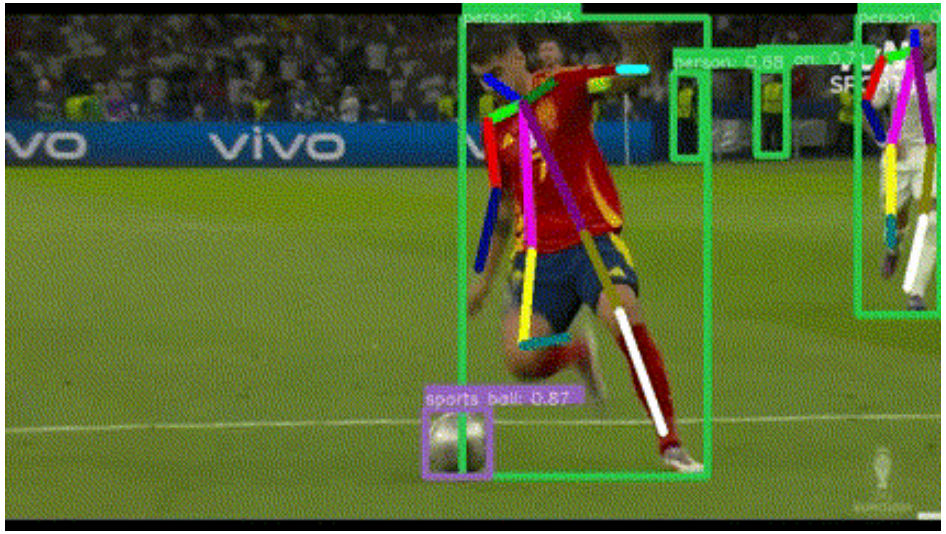
+ Detection

Implementation (Image)

Implementation (Image)



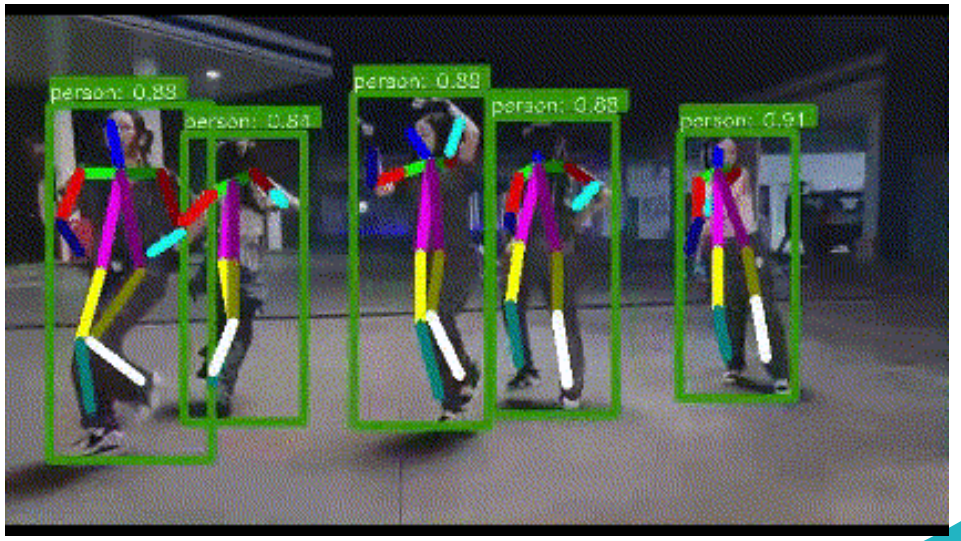
Implementation (Video)



EURO 2024 Final - England vs Spain

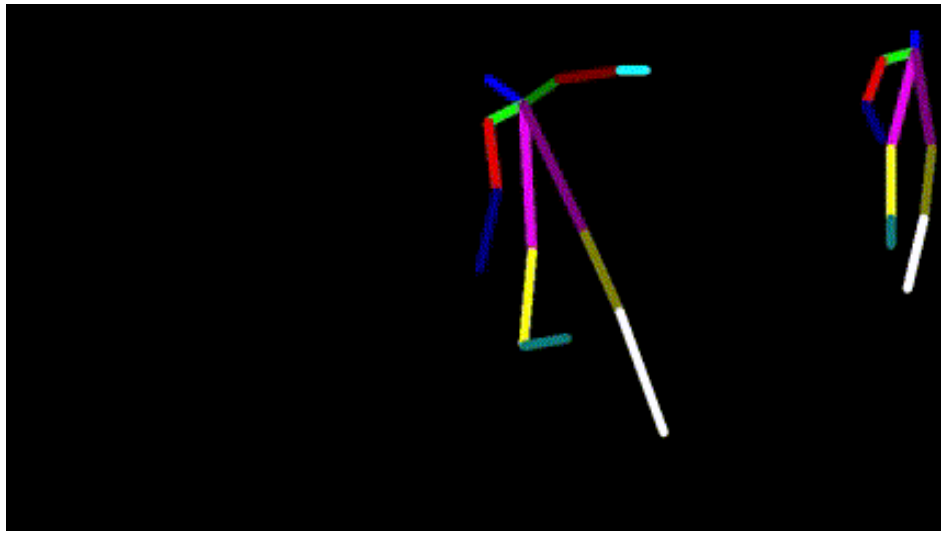
2024 Paris Olympic Final – Pole Vault

BTS - Dynamite

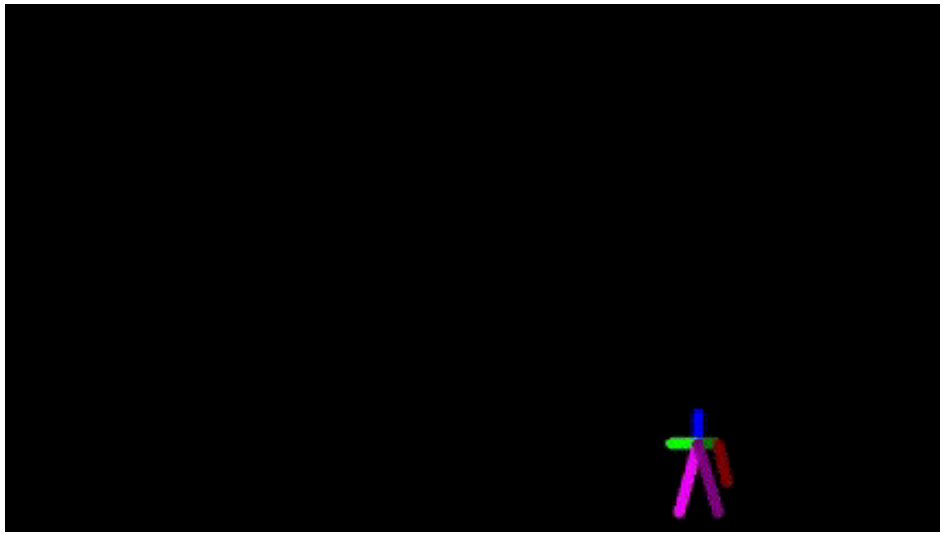


NewJeans - ETA

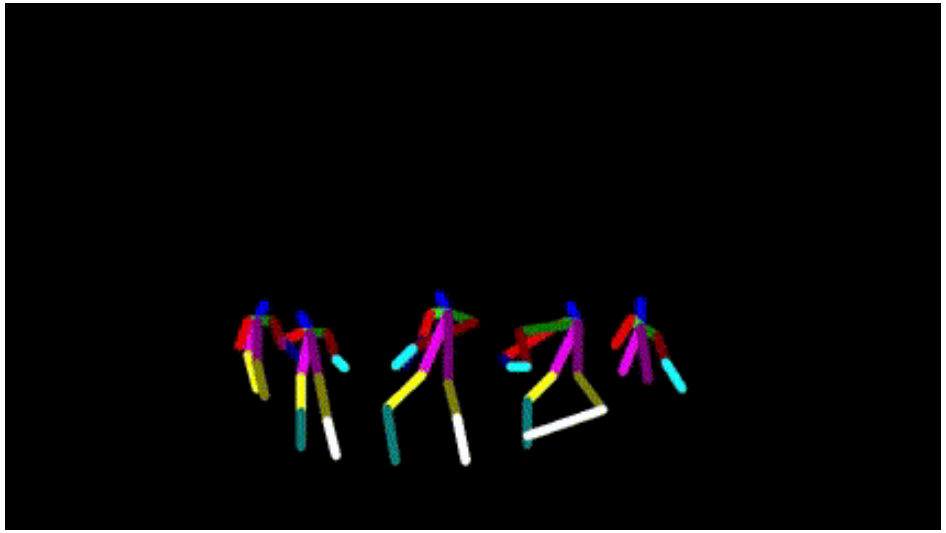
Implementation (Video)



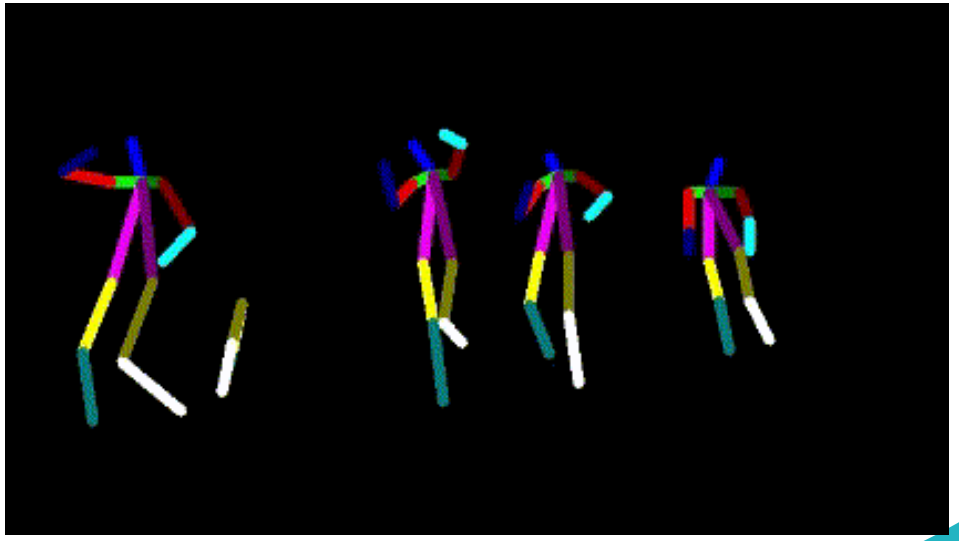
EURO 2024 Final - England vs Spain



2024 Paris Olympic Final – Pole Vault



BTS - Dynamite

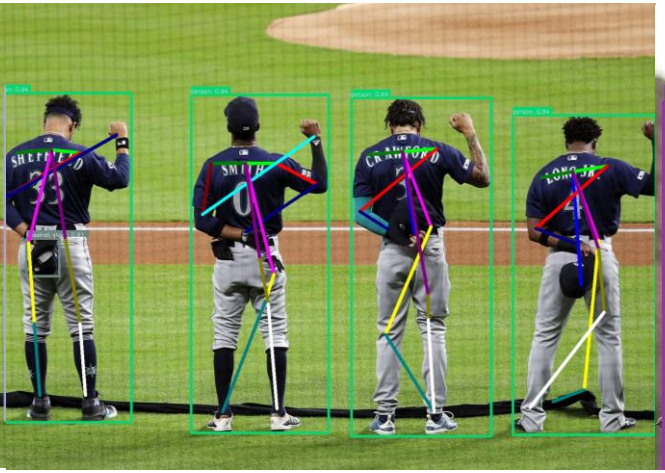
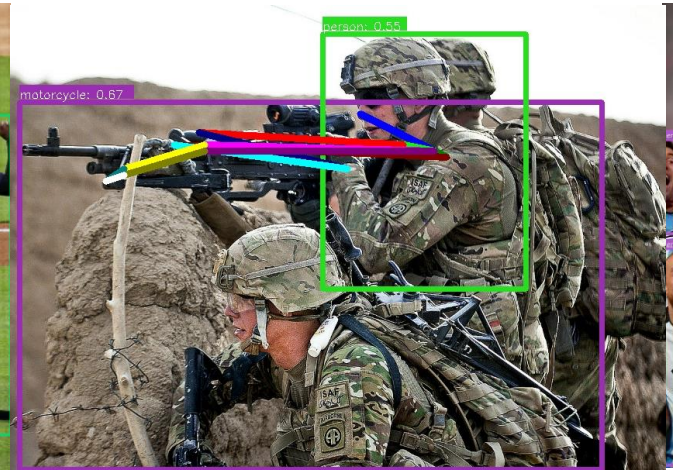


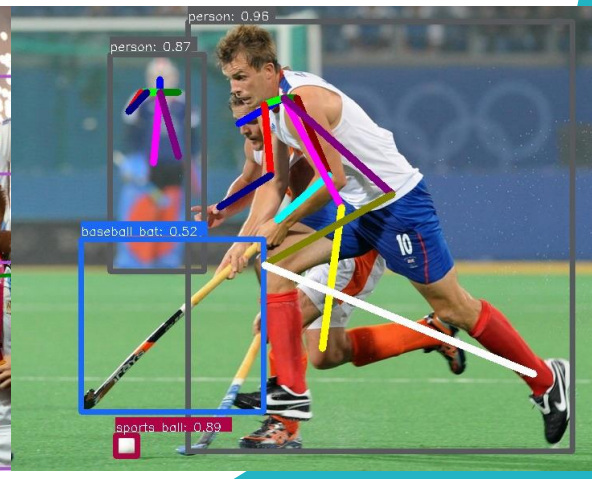
NewJeans - ETA

Conclusion

Limitations

- Degradation of performance when person in the image looks behind or too many people
- Weakness in occlusion, overlap, twist …
- If it detects incorrectly, estimation also works incorrectly
- Have difficulty in estimating small objects
- It can’t distinguish well when same labeled objects are nearby

Conclusion

Enable machines to visually understand and interpret humans and their interactions

- PAF refinement is critical and sufficient for high accuracy, removing the body part confidence map refinement while increasing the network depth
- First combined body and foot keypoint detector
- Prove that combining both detection approaches not only reduces the inference time but also maintains their individual accuracy
- OpenPose, the first open-source library for real-time body, foot, hand and facial keypoint detection

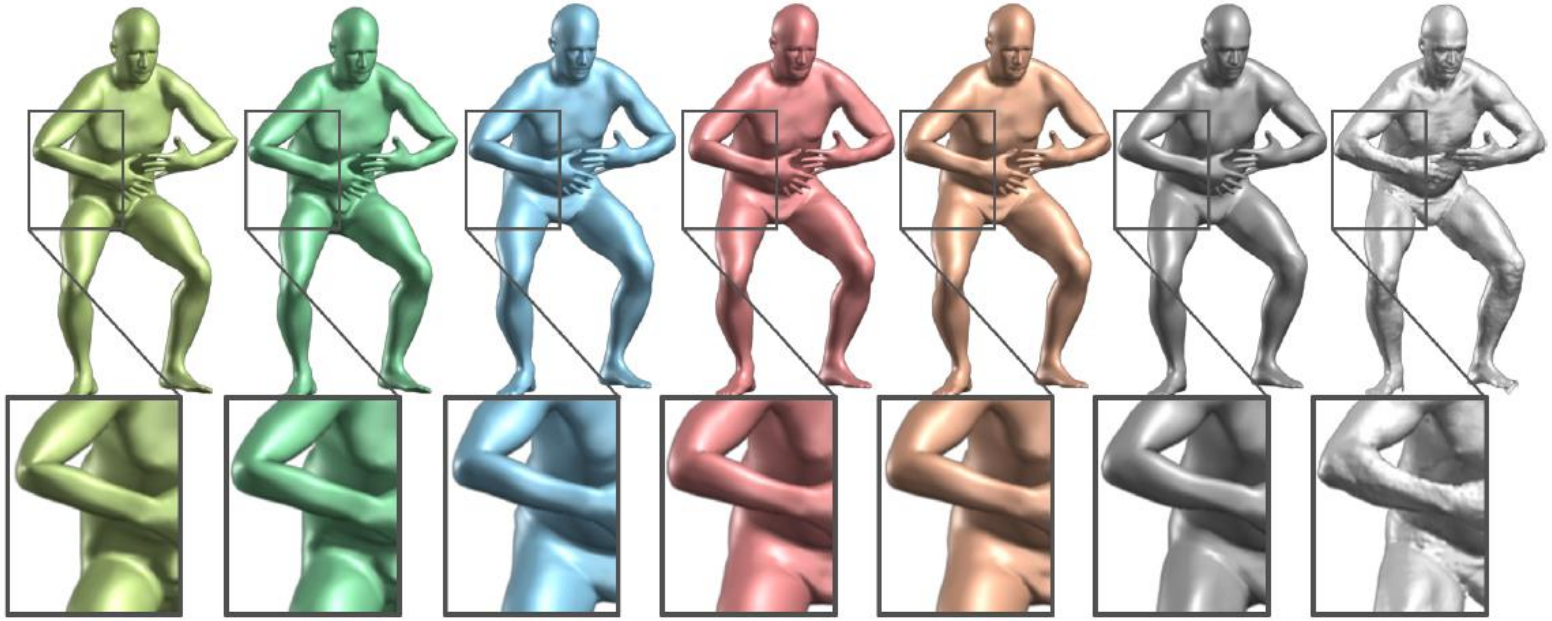
Can utilize in human analysis, human re-identification, retargeting, HCI …

SMPL

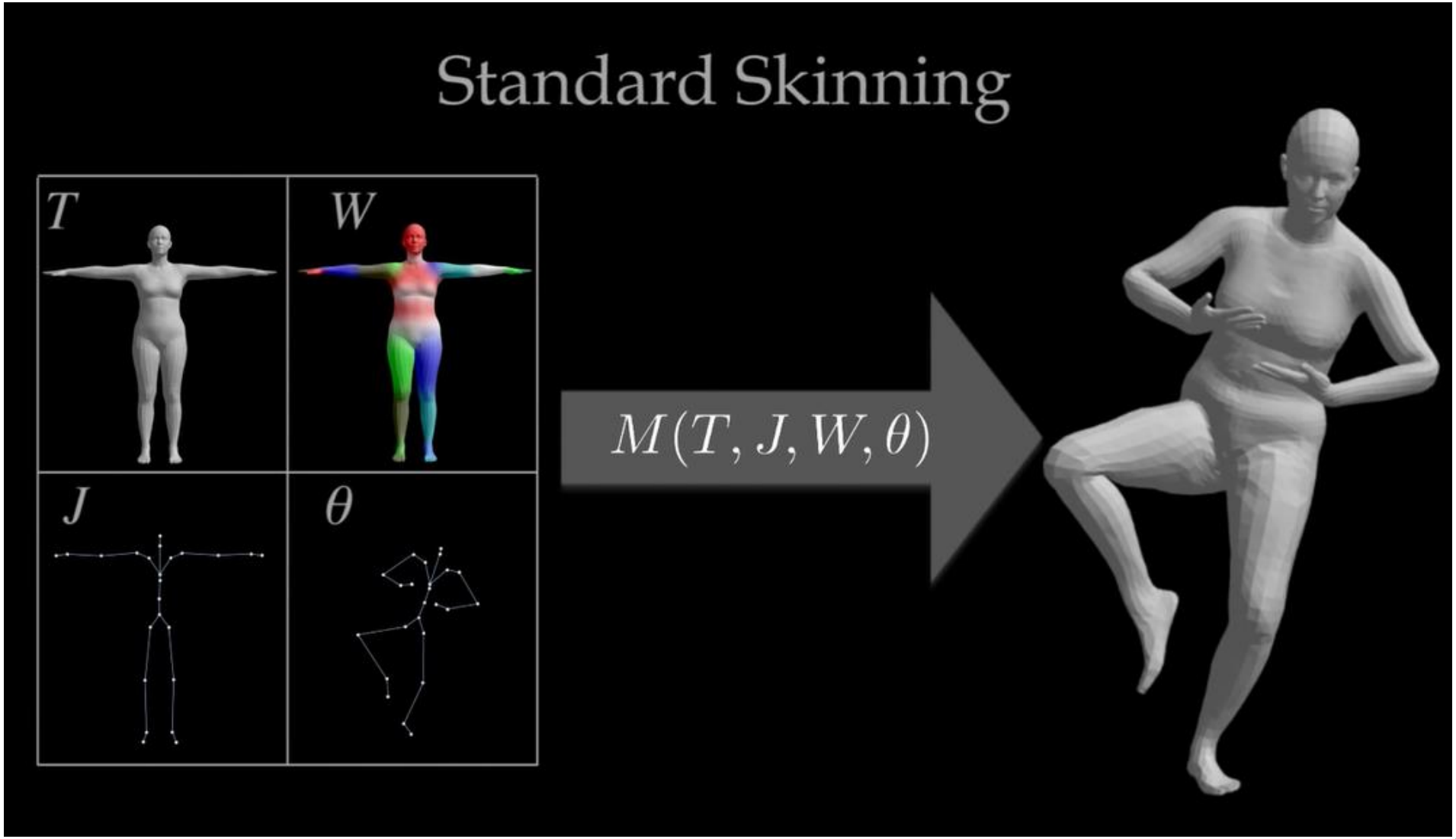
: A Skinned Multi-Person Linear Model

ACM 2016

Introduction

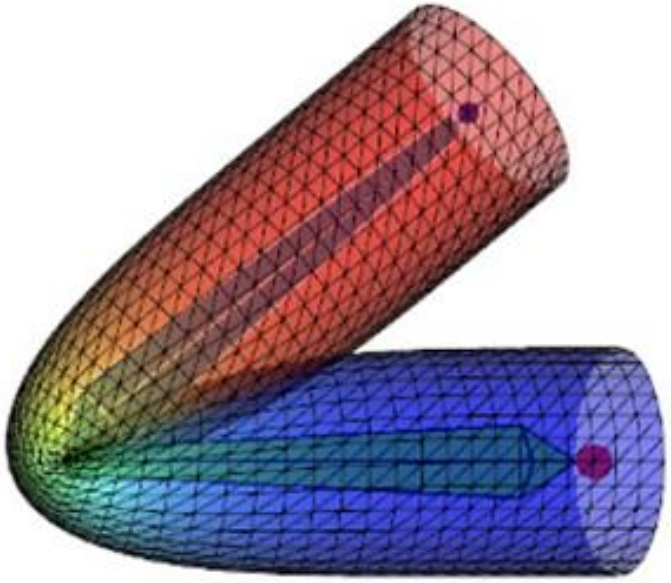


Introduction

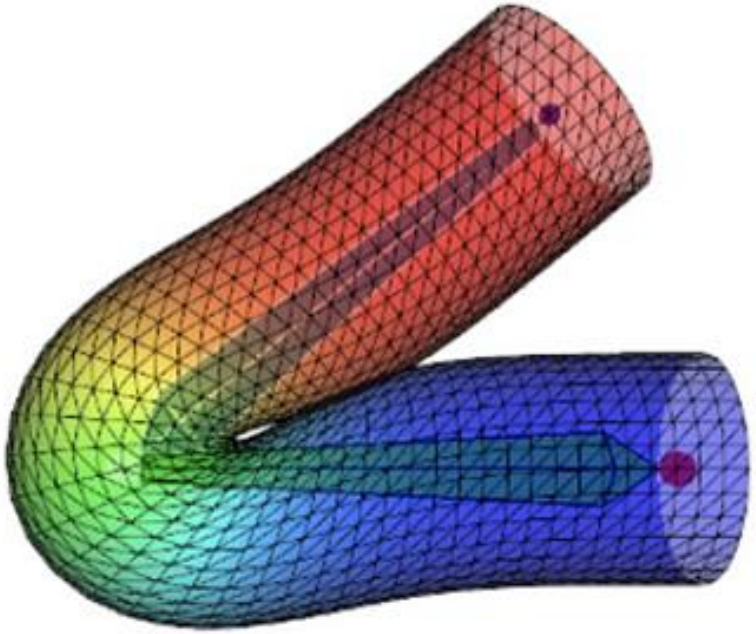


Introduction

LBS (Linear Blend Skinning) & DQS (Dual Quaternion Skinning)



Candy wrapper effect



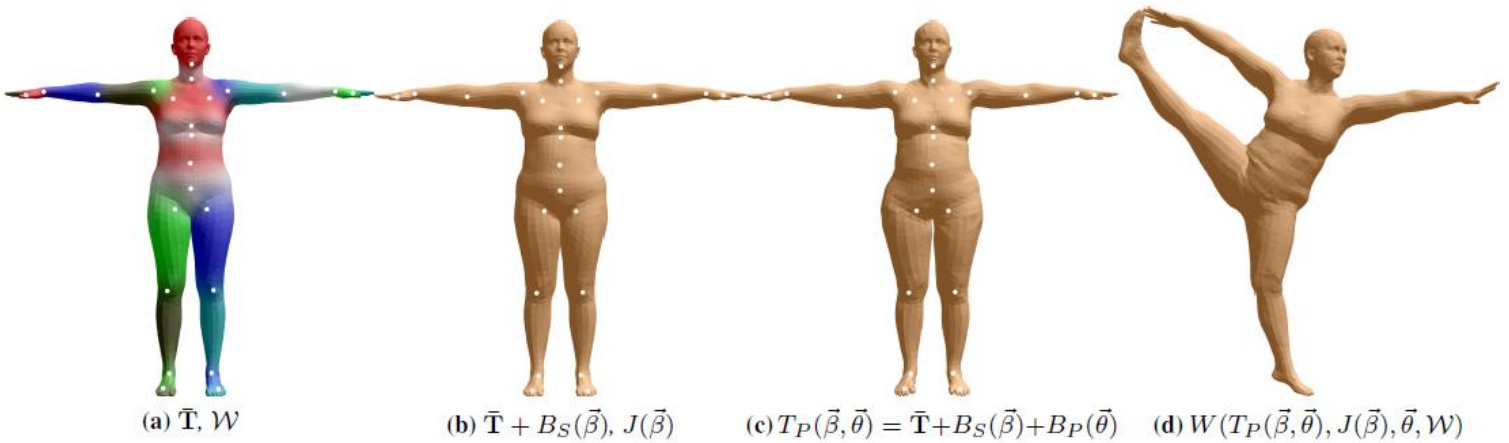
Unnatural protrusion on the knees and elbows when they bend

Model how vertices are related to an underlying skeleton structure

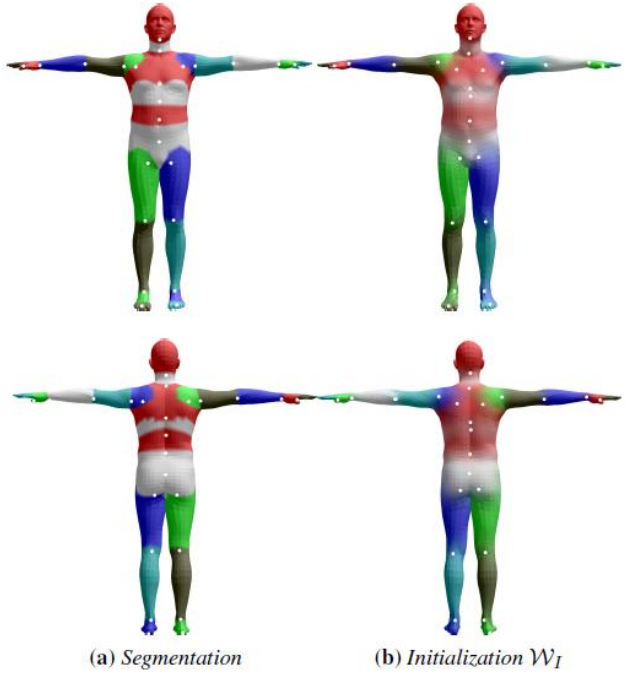
Model Formulation

$N = 6{,}890$ vertices, $K = 23$ joints

The model is defined by a mean template shape represented by a vector of N concatenated vertices $\overline{\mathrm{T}} \in \mathbb{R}^{3\mathrm{n}}$ in the zero pose $\overrightarrow{\theta^*}$



(a) $\bar{\mathbf{T}}$, $\mathcal{W}$ (b) $\bar{\mathbf{T}} + B_S(\vec{\beta})$, $J(\vec{\beta})$ (c) $T_P(\vec{\beta},\vec{\theta}) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta})$ (d) $W(T_P(\vec{\beta},\vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W})$



(a) *Segmentation* (b) *Initialization* $\mathcal{W}_I$

Figure 6: Initialization of joints and blend weights. *Discrete part segmentation in (a) is diffused to obtain initial blend weights, $\mathcal{W}_I$, in (b). Initial joint centers are shown as white dots.*

- $\mathcal{W} \in \mathbb{R}^{\mathrm{N}\times\mathrm{K}}$: a set of blend weights
- $\mathrm{B_S}(\hat{\beta})$: $\mathbb{R}^{|\hat{\beta}|} \rightarrow \mathbb{R}^{3N}$: a blend shape function
- $\mathcal{J}(\hat{\beta})$: $\mathbb{R}^{|\hat{\beta}|} \rightarrow \mathbb{R}^{3K}$: a function to predict K joint locations
- $\mathrm{B_P}(\hat{\theta})$: $\mathbb{R}^{|\hat{\theta}|} \rightarrow \mathbb{R}^{3N}$: a pose-dependent blend shape function
- $W(\cdot)$: a standard blend skinning function
  → applied to rotate the vertices around the estimated joint centers with smoothing defined by the blend weights

$\Rightarrow M(\vec{\beta},\vec{\theta},\Phi)$: $\mathbb{R}^{|\vec{\theta}|\times|\vec{\beta}|} \rightarrow \mathbb{R}^{3N}$
: maps shape and pose parameters to vertices

Blend Skinning

$\vec{\theta} = [\vec{\omega}_0^T, \dots, \vec{\omega}_K^T]^T$ defined by $|\vec{\theta}| = 3 \times 23 + 3 = 72$ parameters

Axis angle for every joint j is transformed to a rotation matrix using the ***Rodrigues formula***
: $\exp(\vec{\omega}_j) = \mathcal{L} + \hat{\vec{\omega}}_j \sin(\|\vec{\omega}_j\|) + \hat{\vec{\omega}}_j^{\,2} \cos(\|\vec{\omega}_j\|)$

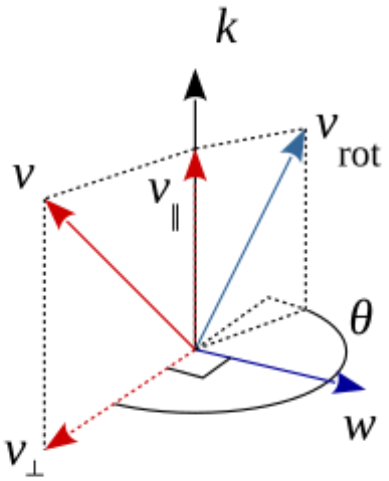
[Rodrigues formula]

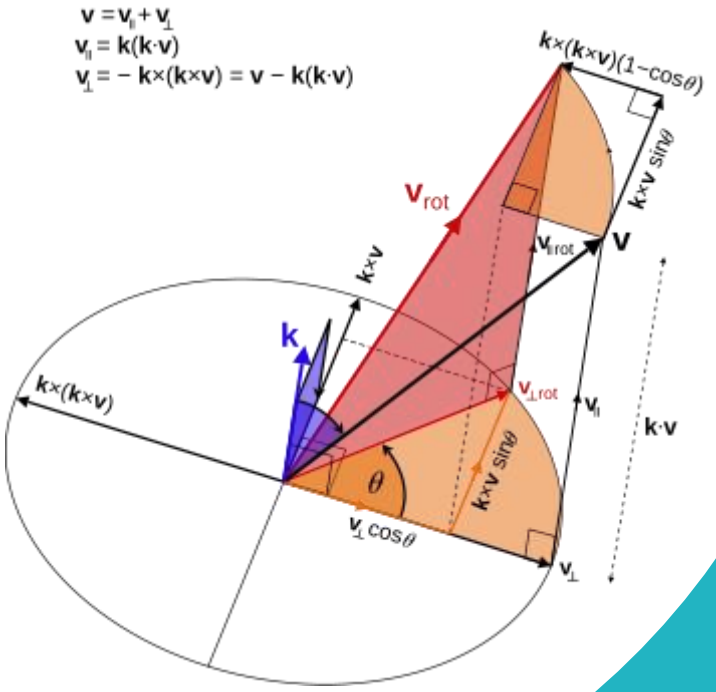
$$v_{rot} = v + (1 - cos\theta)k \times (k \times v) + \sin\theta \cdot k \times v$$

$$K = \begin{bmatrix} (k \times v)_x \\ (k \times v)_y \\ (k \times v)_z \end{bmatrix} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}$$

$$v_{rot} = Rv, \quad R = I + (sin\theta)K + (1 - cos\theta)K^2$$

$$\therefore v_{rot} = v + (sin\theta)Kv + (1 - cos\theta)K^2v$$





Blend Skinning

$W(\bar{T}, J, \vec{\theta}, \mathcal{W})$: $\mathbb{R}^{3N \times 3K \times |\vec{\theta}| \times |\mathcal{W}|} \rightarrow \mathbb{R}^{3N}$ = standard blend skinning function

$$\bar{t}_i' = \sum_{k=1}^{K} w_{k,i} G_k'(\vec{\theta}, J) \bar{t}_i$$

$$G_k'(\vec{\theta}, J) = G_k(\vec{\theta}, J) G_k(\vec{\theta}^*, J)^{-1}$$

$$G_k(\vec{\theta}, J) = \prod_{j \in A(k)} \begin{bmatrix} \exp(\vec{w_j}) & j_j \\ \vec{0} & 1 \end{bmatrix}$$

"To maintain **compatibility**, we keep the basic skinning function and instead modify the template in an additive way and learn a function to predict joint locations"

- $M(\vec{\beta}, \vec{\theta}; \Phi)$

$\Rightarrow M(\vec{\beta}, \vec{\theta}) = W(T_P(\vec{\beta}, \vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W})$

$(\ T_P(\vec{\beta}, \vec{\theta}) = \bar{T} + B_S(\vec{\beta}) + B_P(\vec{\theta})\ )$

$$\bar{t}_i' = \sum_{k=1}^{K} w_{k,i} G_k'\left(\vec{\theta}, J(\vec{\beta})\right)\left(\bar{t}_i + b_{S,i}(\vec{\beta}) + b_{P,i}(\vec{\theta})\right)$$

Blend Skinning

Shape blend shapes

$$B_S(\vec{\beta};\mathcal{S}) = \sum_{n=1}^{|\vec{\beta}|} \beta_n S_n$$

Pose blend shapes

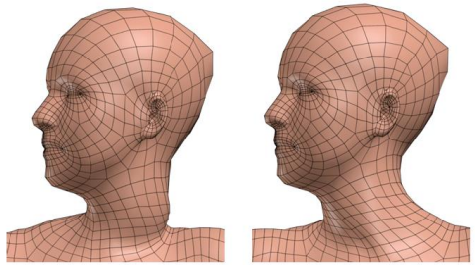
$$B_P(\vec{\theta};\mathcal{P}) = \sum_{n=1}^{9K} \left(R_n(\vec{\theta}) - R_n(\vec{\theta}^*)\right) P_n$$

Joint location

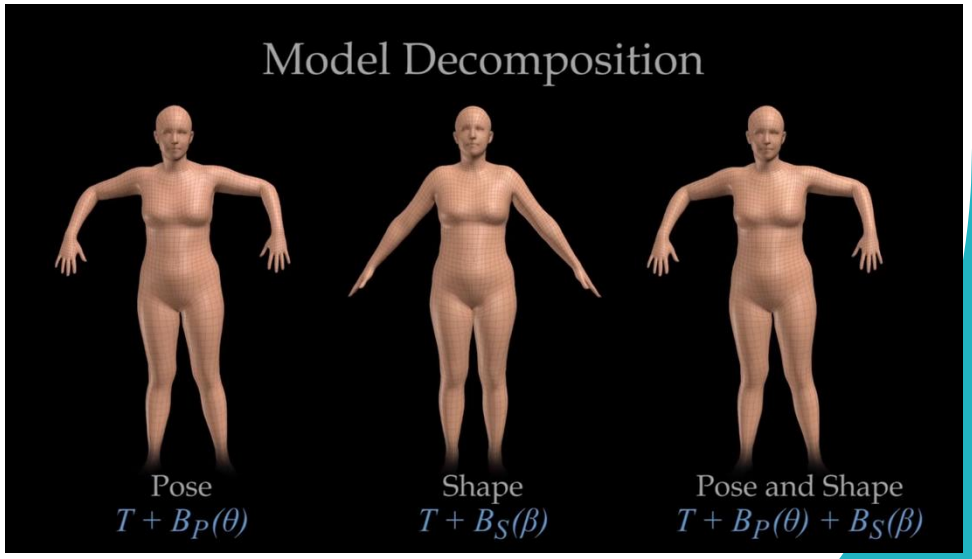
$$J(\vec{\beta};\mathcal{J},\bar{T},\mathcal{S}) = \mathcal{J}\left(\bar{T} + B_S(\vec{\beta};\mathcal{S})\right)$$

SMPL Model

SMPL Parameters : $\Phi = \{\bar{T}, \mathcal{W}, \mathcal{S}, \mathcal{J}, \mathcal{P}\}$



(a) *Euler-angles* (b) *SMPL parameterization*



$$M(\vec{\beta},\vec{\theta};\Phi) = W(T_P(\vec{\beta},\vec{\theta};\bar{T},\mathcal{S},\mathcal{P}), J(\vec{\beta};\mathcal{J},\bar{T},\mathcal{S}), \vec{\theta}, \mathcal{W})$$

$$t_i' = \sum_{k=1}^{K} w_{k,i} G_k'\left(\vec{\theta}, J(\vec{\beta};\mathcal{J},\bar{T},\mathcal{S})\right) t_{P,i}(\vec{\beta},\vec{\theta};\bar{T},\mathcal{S},\mathcal{P})$$

where $t_{P,i}(\vec{\beta},\vec{\theta};\bar{T},\mathcal{S},\mathcal{P}) = \bar{t}_i + \sum_{m=1}^{|\vec{\beta}|} \beta_m s_{m,i} + \sum_{n=1}^{9K} \left(R_n(\vec{\theta}) - R_n(\vec{\theta}^*)\right) p_{n,i}$

Training

Pose Parameter Training

$$E_D\left(\hat{T}^P, \hat{J}^P, \mathcal{W}, \mathcal{P}, \theta\right) = \sum_{j=1}^{P_{reg}} \left\| V_j^P - W\left(\hat{T}_{s(j)}^P + B_P\left(\vec{\theta}_j; \mathcal{P}\right), \hat{J}_{s(j)}^P, \vec{\theta}_j, \mathcal{W}\right)\right\|^2$$

$$E_Y\left(\hat{J}^{\mathrm{P}}, \hat{T}^P\right) = \sum_{i=1}^{P_{subj}} \lambda_U \left\| \hat{J}_i^P - U\left(\hat{J}_i^P\right)\right\|^2 + \left\| \hat{T}_i^P - U\left(\hat{T}_i^P\right)\right\|^2$$

$$E_J\left(\hat{T}^P, \hat{J}^P\right) = \sum_{i=1}^{P_{subj}} \left\| \mathcal{J}_I \hat{T}_i^P - \hat{J}_i^P \right\|^2$$

$$E_P(\mathcal{P}) = \|\mathcal{P}\|_F^2 = (trace[\mathcal{P}^T\mathcal{P}])^{\frac{1}{2}}$$

$$E_W(\mathcal{W}) = \|\mathcal{W} - \mathcal{W}_I\|_F^2$$

$$\boldsymbol{E_*\left(\hat{T}^P, \hat{J}^P, \theta, \mathcal{W}, \mathcal{P}\right)} = \boldsymbol{E_D + \lambda_Y E_Y + \lambda_J E_J + \lambda_P E_P + E_W}$$

Training

Joint Regression

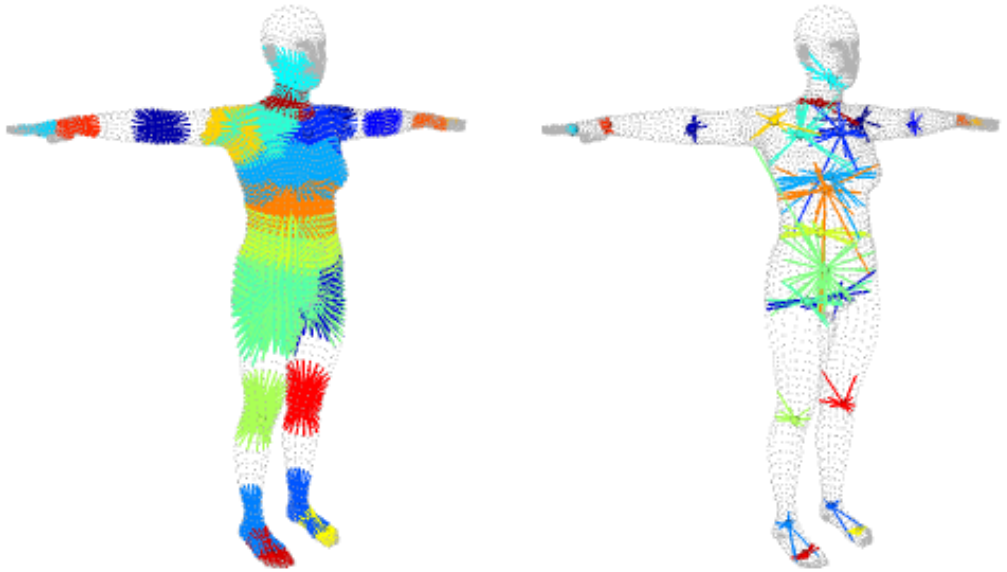
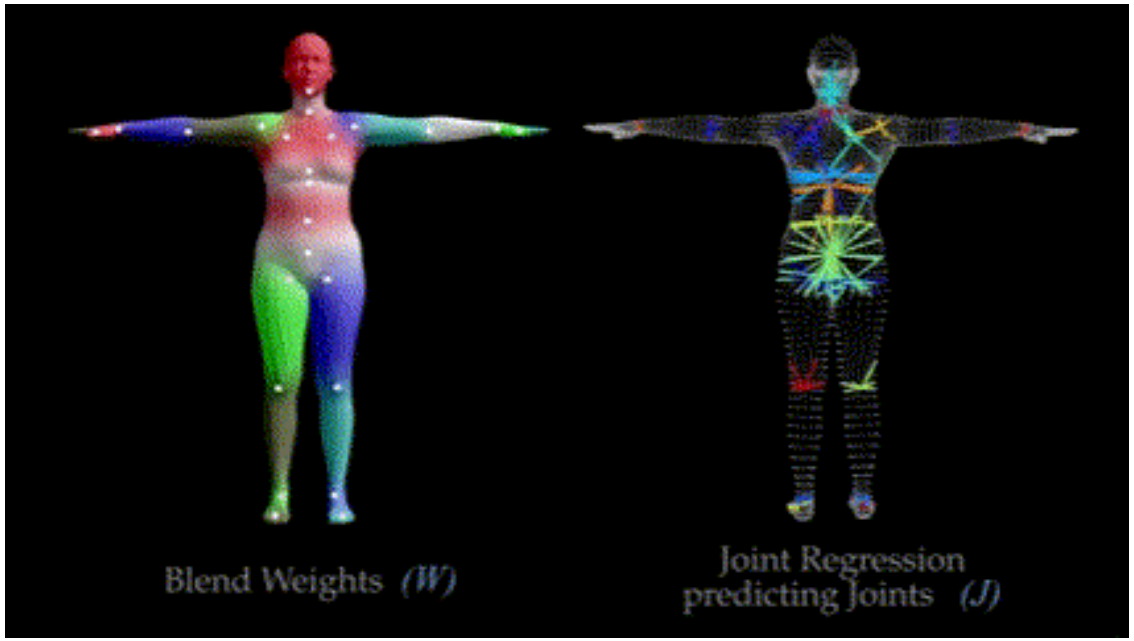


Figure 7: Joint regression. *(left) Initialization. Joint locations can be influenced by locations on the surface, indicated by the colored lines. We assume that these influences are somewhat local. (right) Optimized. After optimization we find a sparse set of vertices and associated weights influencing each joint.*



Optimizing E_* gives a template mesh and joint locations for subjects,
but we want to predict joint locations for new subjects with new body shapes

Training

Shape Parameter Training

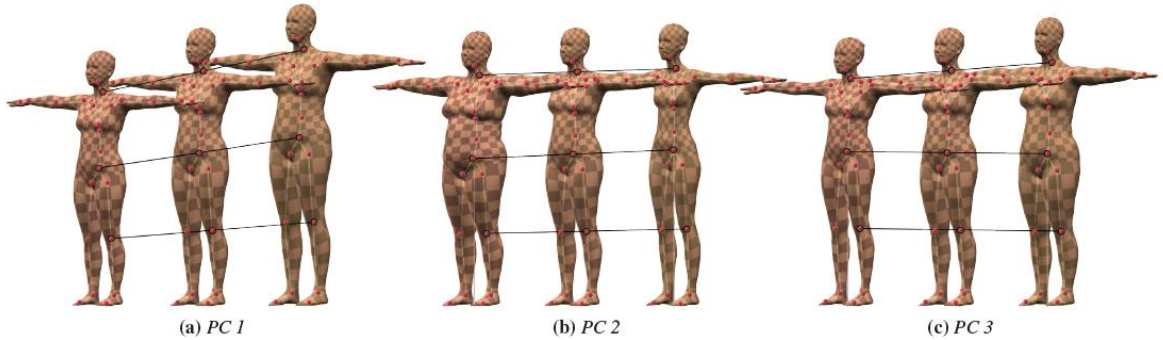


Figure 8: Shape blend shapes. *The first three shape principal components of body shape are shown. PC1 and PC2 vary from -2 to +2 standard deviations from left to right, while PC3 varies from -5 to +5 standard deviations to make the shape variation more visible. Joint locations (red dots) vary as a function of body shape and are predicted using the learned regressor, $\mathcal{J}$.*

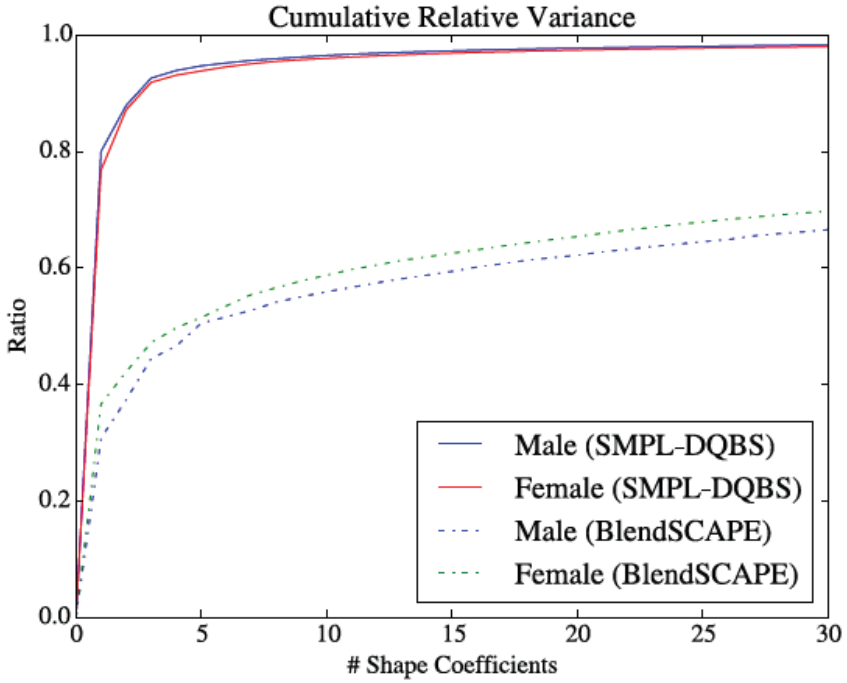
$W_e(\hat{T}_\mu^P, \hat{J}_\mu^P, \vec{\theta}, \mathcal{W}), V_{j,e}^S \in \mathbb{R}^3$

: edge of the model and edge of the registration

$$\vec{\theta}_j = \underset{\vec{\theta}}{argmin} \sum_e \left\| W_e(\hat{T}_\mu^P + B_P(\vec{\theta}; \mathcal{P}), \hat{J}_\mu^P, \vec{\theta}, \mathcal{W}) - V_{j,e}^S \right\|^2$$

$$\hat{T}_j^S = \underset{\hat{T}}{argmin} \left\| W(\hat{T} + B_P(\vec{\theta}_j; \mathcal{P}), \mathcal{J}\hat{T}, \vec{\theta}_j, \mathcal{W}) - V_j^S \right\|^2$$

Run PCA on $\{\hat{T}_j^S\}_{j=1}^{S_{subj}}$ to obtain $\{\bar{T}, \mathcal{S}\}$



Quantitative Evaluation

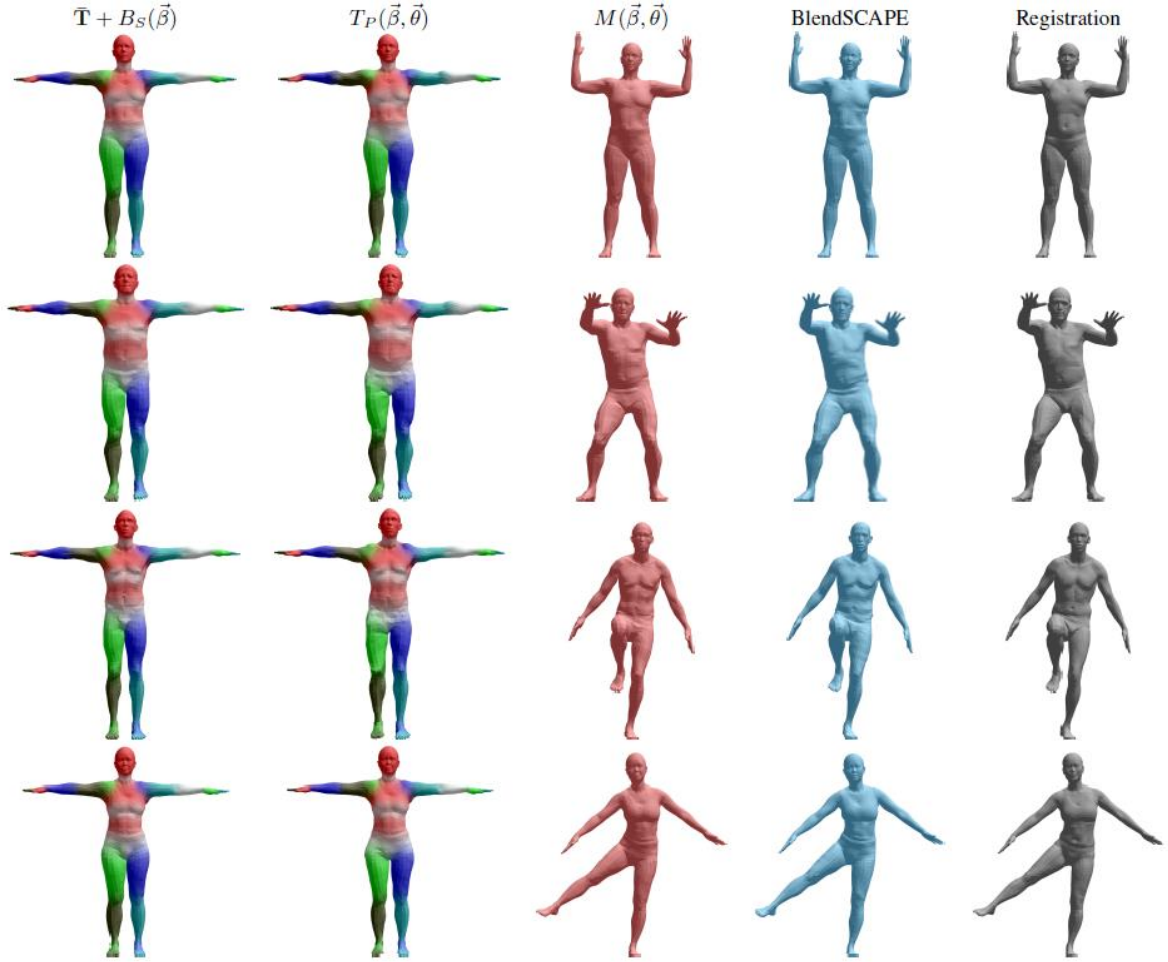


Figure 10: *Model fitting with intermediate stages. We fit both BlendSCAPE (blue) and SMPL-LBS, $M(\vec{\beta}, \vec{\theta})$, (red) to registered meshes by optimizing pose and shape. $\bar{\mathbf{T}} + B_S(\vec{\beta})$ shows the estimated body shape and $T_P(\vec{\beta}, \vec{\theta})$ shows the effects of pose-dependent blend shapes. Here we show SMPL-LBS, because T_P shows more variation due to pose than SMPL-DQBS.*

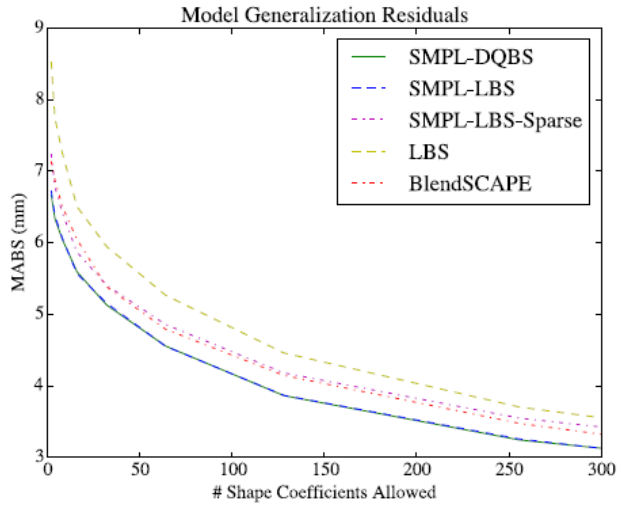


Figure 11: *Model generalization indicates how well we can fit an independent registration. Mean absolute vertex error versus the number of shape coefficients used.*

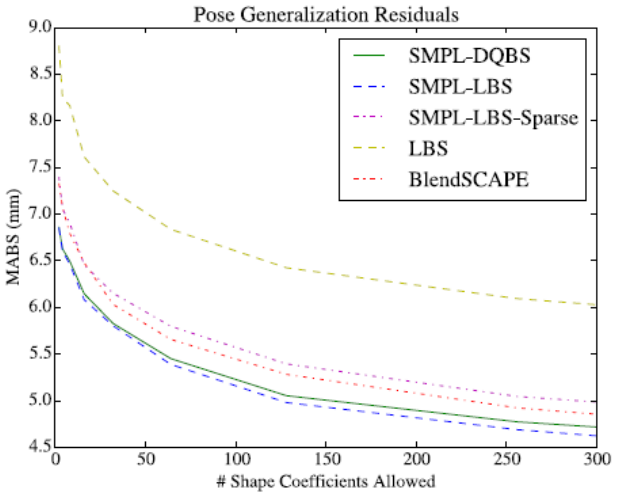


Figure 12: *Pose generalization error indicates how well a fitted shape generalizes to new poses.*

- SMPL is more accurate on average
- Pose blend shapes have much more to correct with LBS
- How well standard LBS fits the test data

Visual Evaluation

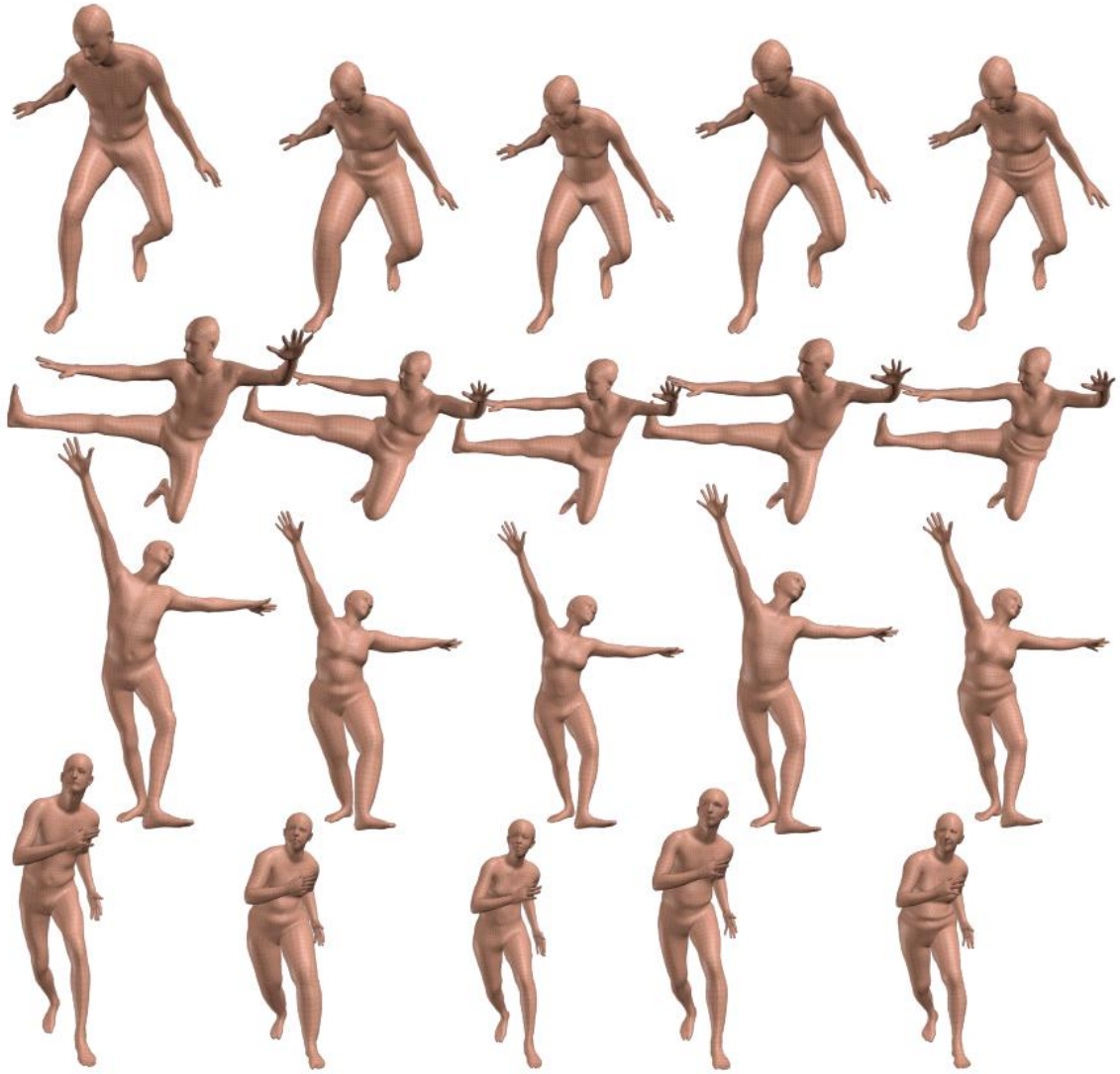


Figure 13: Animating SMPL. *Decomposition of SMPL parameters into pose and shape: Shape parameters, $\vec{\beta}$, vary across different subjects from left to right, while pose parameters, $\vec{\theta}$, vary from top to bottom for each subject.*

Decomposition of shape parameters $\vec{\beta}$ and pose parameters $\vec{\theta}$ in SMPL

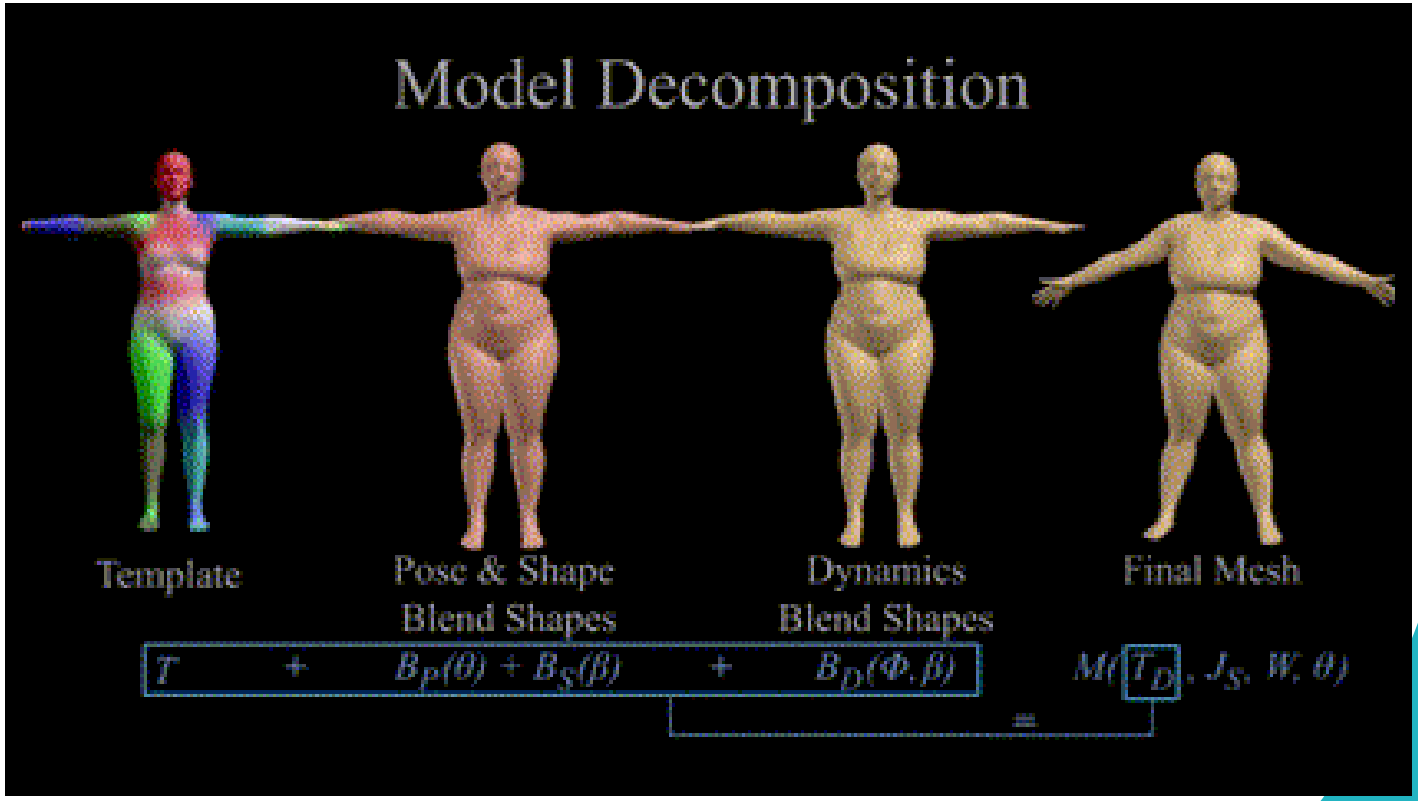
DMPL : Dynamic SMPL

SMPL models static soft-tissue deformations with pose

Fitting 4D registrations by optimizing only the pose of a SMPL model with a personalized template shape

$$\vec{\phi}_t = \left[\dot{\vec{\theta}}_t, \ddot{\vec{\theta}}_t, v_t, a_t, \vec{\delta}_{t-1}, \vec{\delta}_{t-2}\right]$$

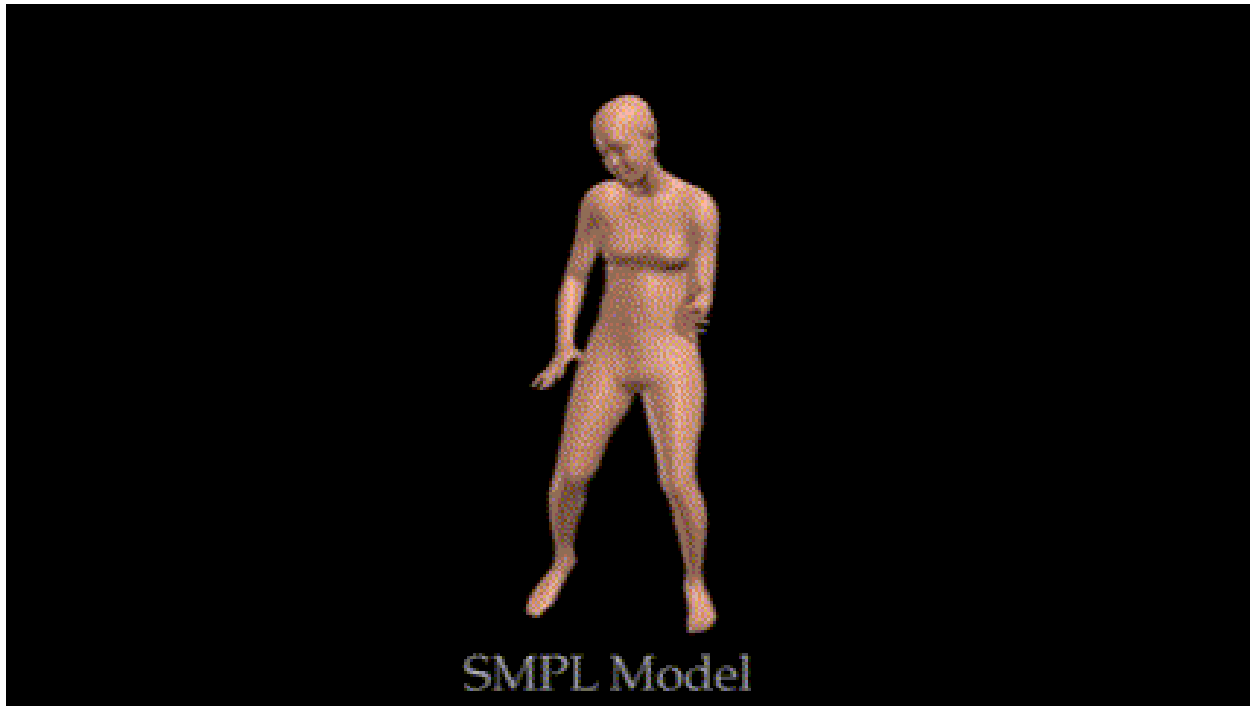
Dynamic blend function : $B_D(\vec{\theta}_t, \vec{\beta})$

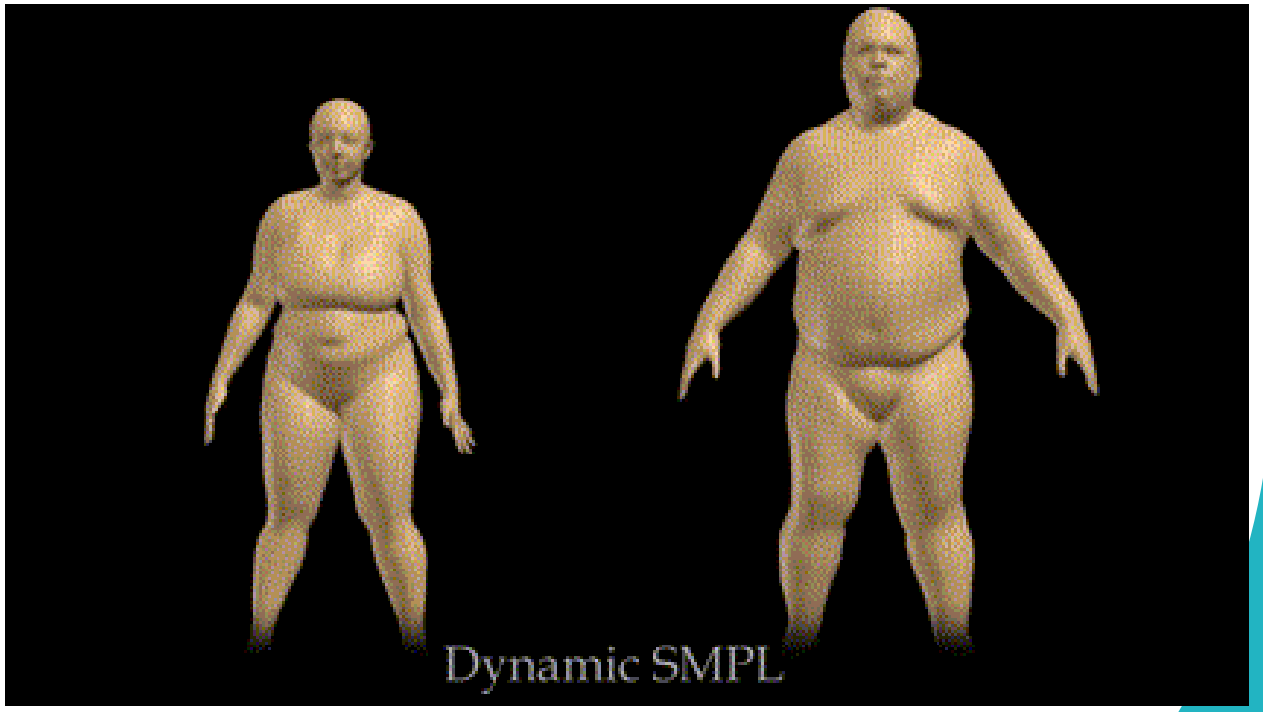


Shape in the zero pose : $T_D(\vec{\beta}, \vec{\theta}_t, \vec{\phi}_t) = \bar{T} + B_S(\vec{\beta}) + B_P(\vec{\theta}_t) + \boldsymbol{B_D(\vec{\phi}_t, \vec{\beta})}$

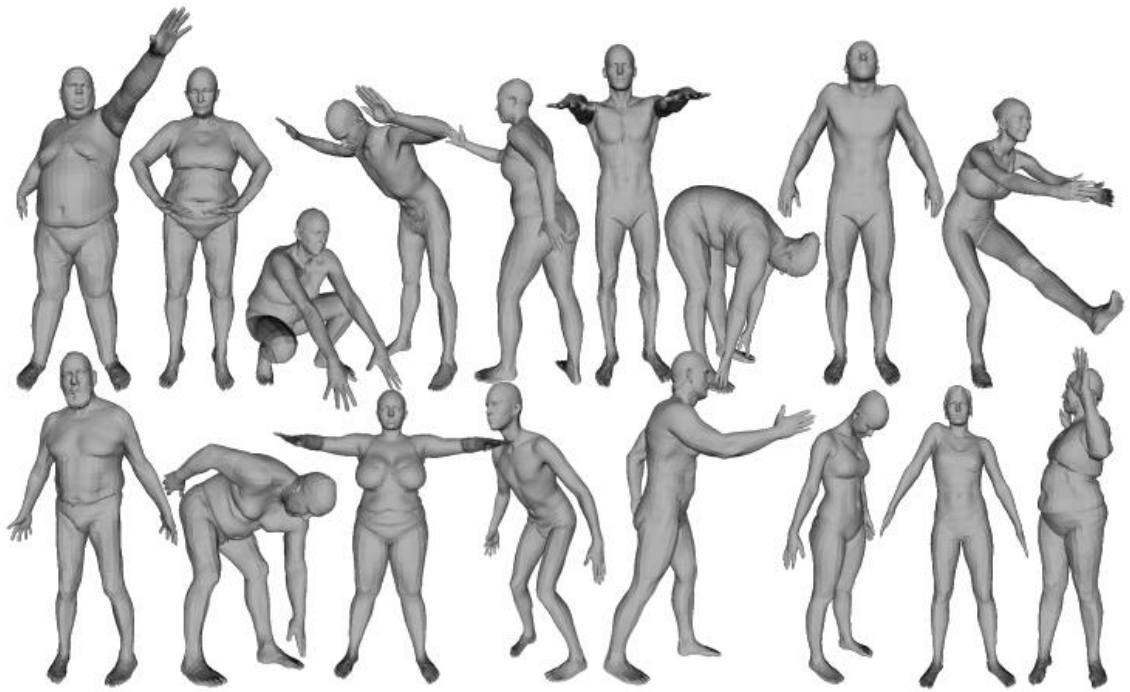
Predicting dynamic blendshapes : $B_D(\vec{\phi}_t, \vec{\beta}; \mathcal{D}) = \mu_D + \mathcal{D}\vec{\delta}_t = \mu_D + \mathcal{D}f(\vec{\phi}_t, \vec{\beta})$

SMPL vs DMPL

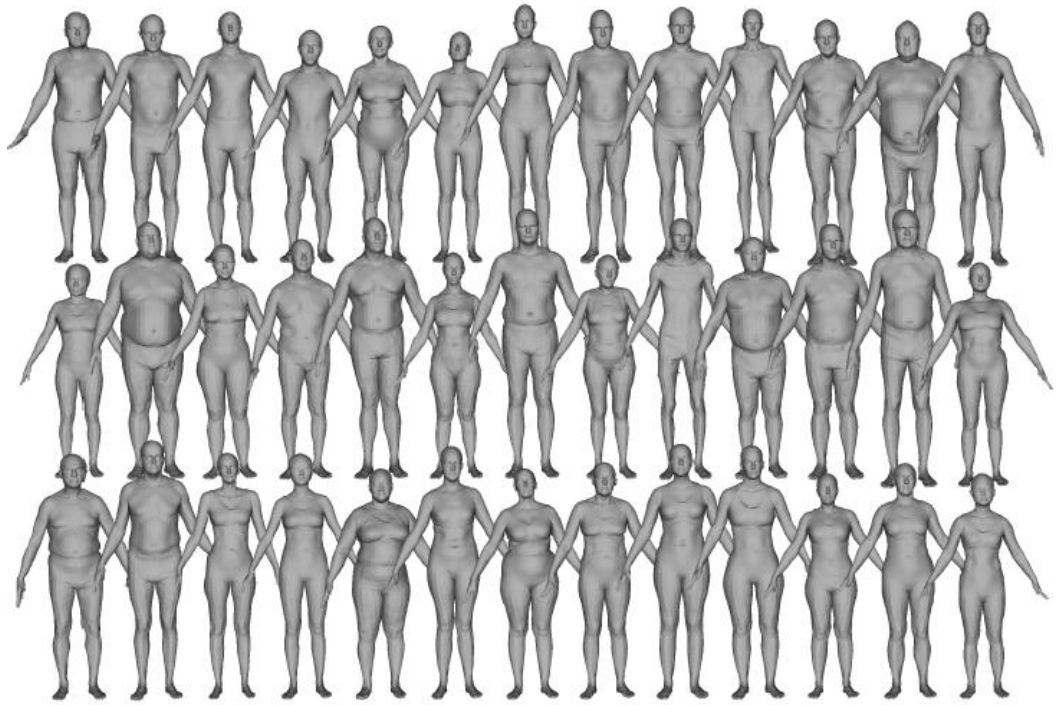




Dataset



Multipose dataset



Multishape dataset

Reference

- Alexander Toshev, Christian Szegedy.「**DeepPose: Human Pose Estimation via Deep Neural Networks**」**,** 2013
- Alejandro Newell, Kaiyu Yang, Jia Deng.「**Stacked Hourglass Networks for Human Pose Estimation**」, 2016
- Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, Yaser Sheikh.「**OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields**」, 2018
- Shih-En Wei, Varun Ramarkrishna, Takeo Kanade, Yaser Sheikh.「**Convolutional Pose Machines**」**,** 2016
- Gao Huang, Zhuang Liu, Laurens Van der Maaten.「**Densely Connected Convolution Layers**」**,** 2018
- Zhe Cao, Tomas Simon, Shih-En Wei, Yaser Sheikh.「**Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields**」**,** 2017
- https://learnopencv.com/object-keypoint-similarity/
- https://ctkim.tistory.com/entry/Human-Pose-Estimation
- https://github.com/CMU-Perceptual-Computing-Lab/openpose
- https://towardsdatascience.com/review-deeppose-cascade-of-cnn-human-pose-estimation
- https://velog.io/@hanlyang0522/OpenPose-Realtime-Multi-Person-2D-Pose-Estimation-using-Part-Affinity-Fields
- https://smpl.is.tue.mpg.de/

THANK YOU