3D Hand Pose Estimation

JGR-P2O: Joint Graph Reasoning based Pixel-to-Offset Prediction Network

송 재 훈

Vision & Display Systems Lab.

Dept. of Electronic Engineering, Sogang University

Outline

- 3D Hand Pose Estimation from single Depth image
- Applications
- Challenges
- Depth Camera
- Public Datasets
- Generate Input (Crop hand image)
- JGR-P2O : Joint Graph Reasoning based Pixel-to-Offset Prediction Network for 3D Hand Pose Estimation from a Single Depth Image

  - Related Work

  - Methods

  - Experiments

Paper Information

- JGR-P2O: Joint Graph Reasoning based Pixel-to-Offset Prediction Network for 3D Hand Pose Estimation from a Single Depth Image
- Authors : Linpu Fang[1], Xingyan Liu[1], Li Liu[2], Hang Xu[3], and Wen-xiong Kang[1]

  [1] South China University of Technology, China

  - ECCV 2020 accepted

서강대학교 SOGANG UNIVERSITY

VDS LAB

3D Hand Pose Estimation

- Goal : Localize hand keypoints (joints) from a ***single*** depth map

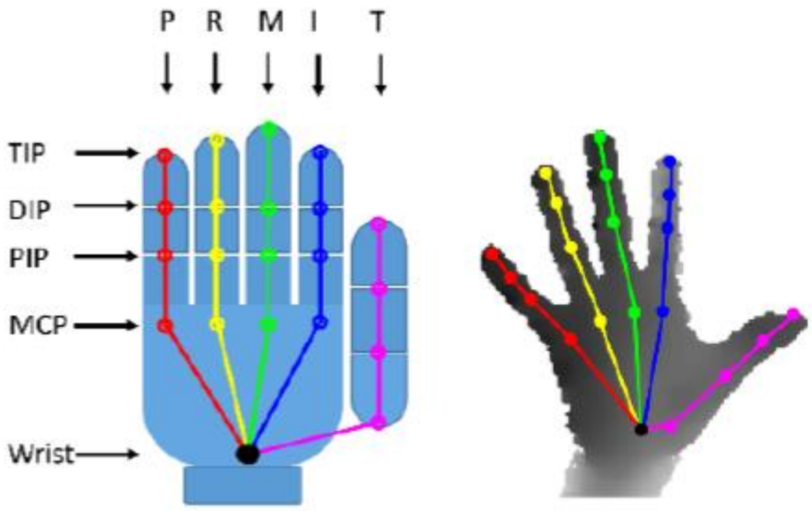


Fig. 3D hand model: 21 keypoints (joints)

3D Hand Pose Estimation

- Gesture recognition aims at classifying a set of discrete hand poses,
  : sometimes related to hand pose estimation
- Hand segmentation locates the hand more precisely by a binary mask
  : an enhanced version of hand detection

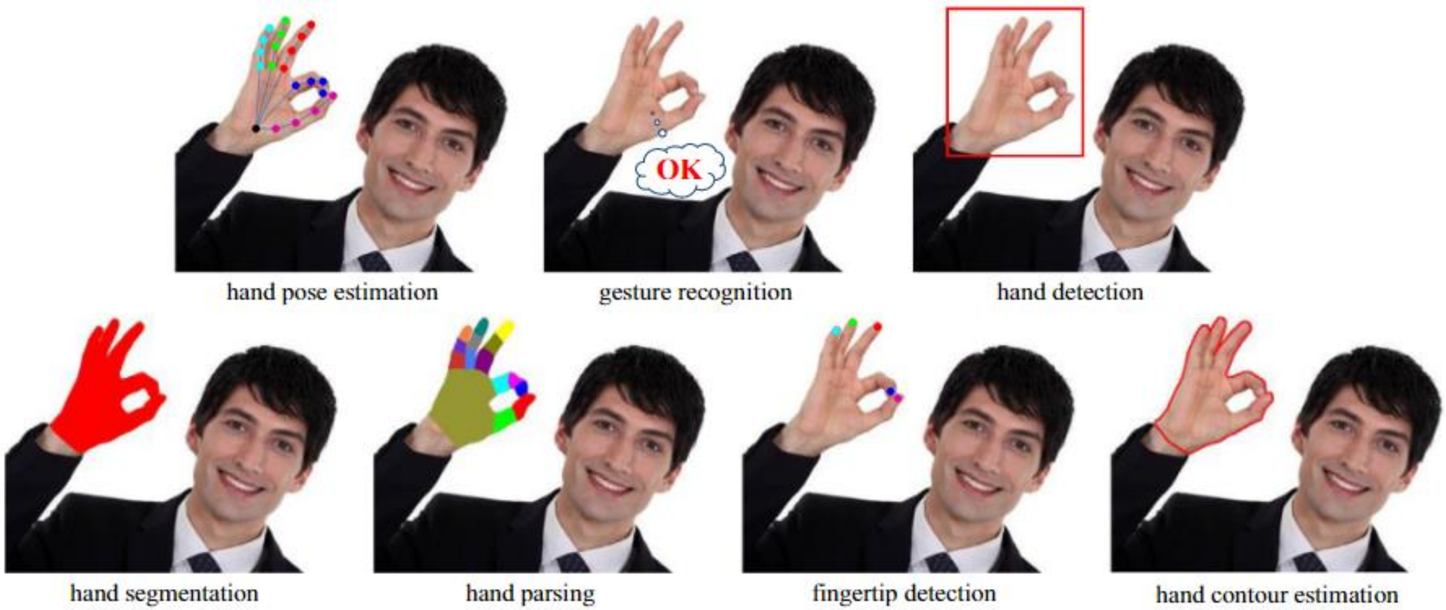
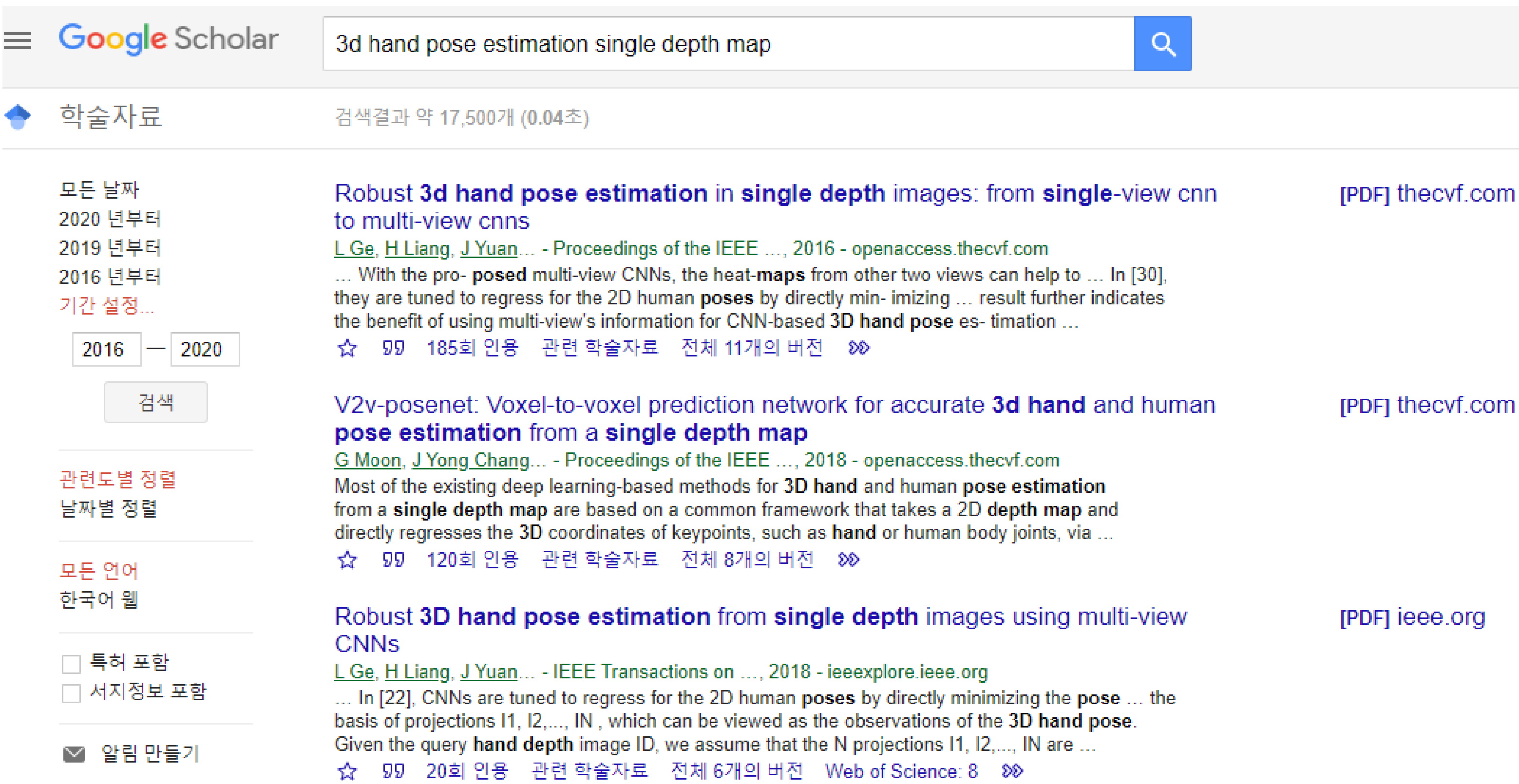


Fig. Comparison of hand pose estimation and its similar 6 fields

3D Hand Pose Estimation

- Still hot topic : more than 17,000 publications over last 5 years



Applications

- Crucial Technique for HCI (Human Computer Interaction) and VR & AR

VR (Oculus Rift)

AR (MS HoloLens)

Leap Motion Control

Touchless Control

Challenges

- Diverse geometric (shape) variations
- Viewpoints
- Severe self occlusions between different fingers
- Self similar parts
- Noise (poor quality of depth image)

Depth Cameras

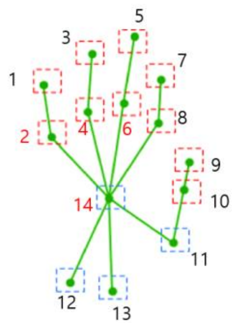
- Sensing principle : time of flight (ToF), structured light, or other stereo vision technologies.
- performance is influenced by environmental factors and application scenarios
  - The selection mainly depends on the nature of the problem

Camera	Model	Release date	Discontinued	Depth technology	Range	Max depth Fps	
Microsoft Kinect	1st generation	2010	Yes	Structured light	0.5–4.5 m	30	
	2nd generation	2014	Yes	ToF	0.5–4.5 m	30	
ASUS Xtion	PRO LIVE	2012	Yes	Structured light	0.8–3.5 m	60	
	2	2017	Yes	Structured light	0.8–3.5m	30	
Leap Motion (updated on December 20, 2018)		2013	No	Dual IR stereo vision	0.03–0.6 m	200	Short-range applications
Intel RealSense	F200	2014	Yes	Structured light	0.2–1.2 m	60	
	R200	2015	No	Structured light	0.5–3.5 m	60	
	LR200	2016	Yes	Structured light	0.5–3.5 m	60	Mid and long-range applications
	SR300	2016	No	Structured light	0.3–2 m	30	
	ZR300	2017	Yes	Structured light	0.5–3.5 m	60	
	D415	2018	No	Structured light	0.16–10 m	90	
	D435	2018	No	Structured light	0.11–10 m	90	
SoftKinetic	DS311	2011	Yes	ToF	0.15–4.5 m	60	
	DS325	2012	Yes	ToF	0.15–1 m	60	
	DS525	2013	Yes	ToF	0.15–1 m	60	
	DS536A	2015	Yes	ToF	0.1–5 m	60	
	DS541A	2016	Yes	ToF	0.1–5m	60	
Creative Interactive Gesture		2012	Yes	ToF	0.15–1 m	60	
Structure Sensor (updated on July 24, 2018)		2013	No	Structured light	0.4–3.5 m	60	Mobile applications

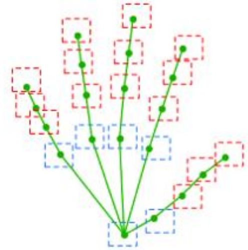
Table. Popular commercial depth cameras

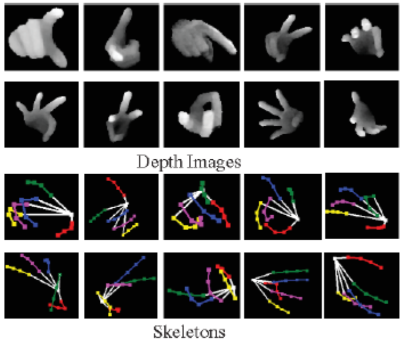
Hand Pose Public Datasets

- NYU : 14 joints
  - 72K training and 8.2K testing depth images

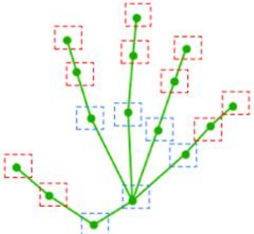


- MSRA : 21 joints
  - 76K depth images from 9 subjects with 17 gestures
  - Leave-one-subject-out cross-validation





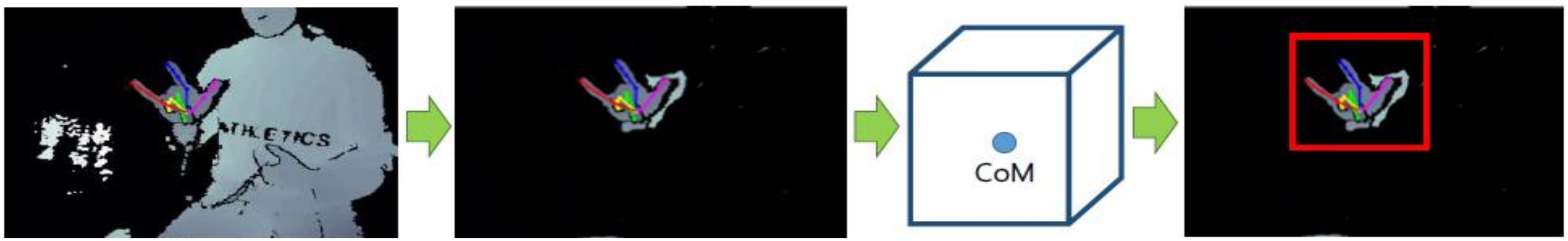
- ICVL : 16 joints
  - 330K training and 1.6K testing depth images
  - 10 different subjects



[1] Oberweger,, et al. "Deepprior ++: Improving fast and accurate 3d hand pose estimation." ICCV 2017

Generating Input

- Crop the hand image



Depth map from a dataset

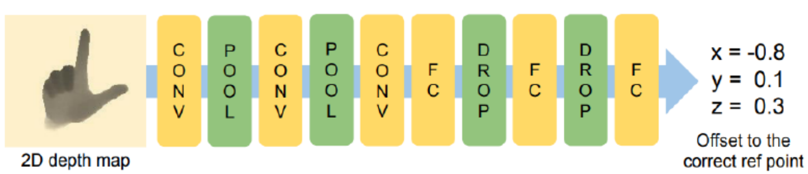
Depth thresholding and calculate center-of-mass(CoM)

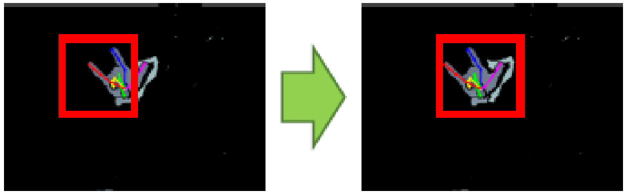
Draw a fixed-size cubic box around the CoM

Project the cubic box on the 2D image and crop the hand region

- Simple depth thresholding can exclude some parts of hand or human body

  → Refine the estimated CoM using a simple network [1]





Effect of the CoM refinement

Introduction

- Key Ideas

1) modeling the dependencies among joints and relations between the pixels and joints

→ help to learn more abundant contextual information and better local feature representation

2) unifying the dense pixel-wise offset predictions and direct joint regression

- Proposed method

1) Graph convolutional network(GCN) based joint graph reasoning module

- modeling the complex dependencies among joints
- augment the representation capability of each pixel.

2) Pixel-to-offset prediction module

- estimate all pixels' offsets to joints and calculate the joints' positions by weighted average over all pixels' predictions

→ discarding the complex post processing operations

Related Work

- Regression-based Methods
  - Directly regressing 3D hand pose parameters (3D coordinates / joint angles)

    \- global regression within the fully-connected layers incurs highly non-linear mapping

    → reduce the estimation accuracy

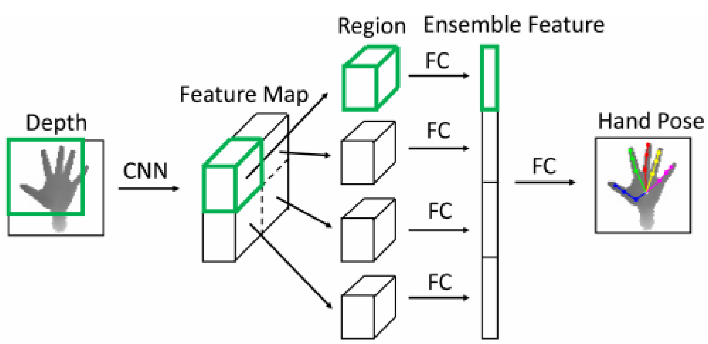


Fig. Region Ensemble Network (REN) [1]

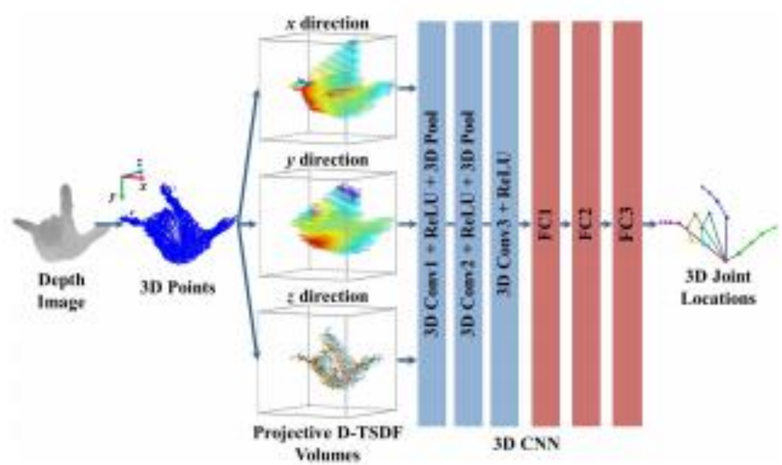


Fig. 3D CNNs [2]

[1] Wan, C., et al. "Dense 3d regression for hand pose estimation." CVPR 2018.
[2] Moon, G, et al. "V2V-PoseNet: Voxel-to-Voxel Prediction Network for Accurate 3D Hand and Human Pose Estimation From a Single Depth Map." CVPR 2018.

Related Work

- Detection-based Methods
  - Dense local prediction manner via setting a heat map for each joint
    - Recent works directly detect 3D joints from 3D heat maps (3D CNN)

      → poor trade-off (high accuracy, computationally inefficient)

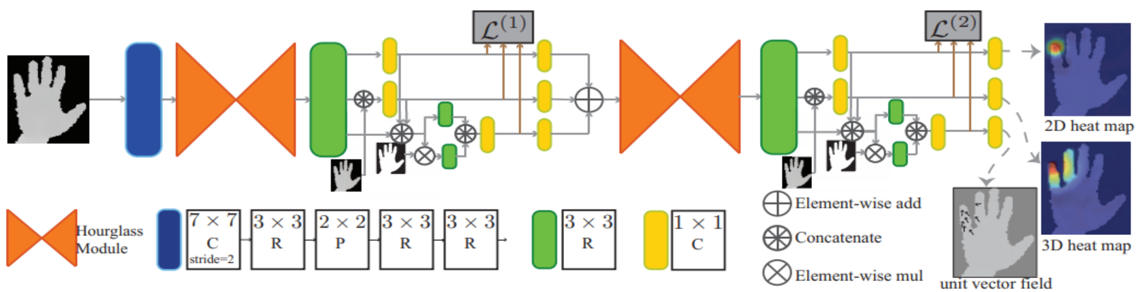


Fig. Dense 3D regression [1]

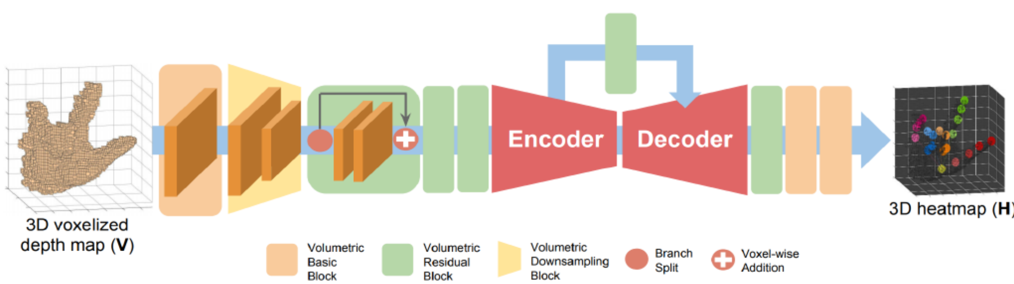


Fig. V2V-PoseNet [2]

[1] Chen, X.., et al. "Pose guided structured region ensemble network for cascaded hand pose estimation." Neurocomputing 2019.
[2] Madadi, M., et al. "End-to-end global to local cnn learning for hand pose recovery in depth data." arXiv 2017.

Related Work

- Hierarchical Methods
  - Divide the hand joints into different subsets / network branches to extract local pose features
  - All the local pose features are combined for forming the global representation

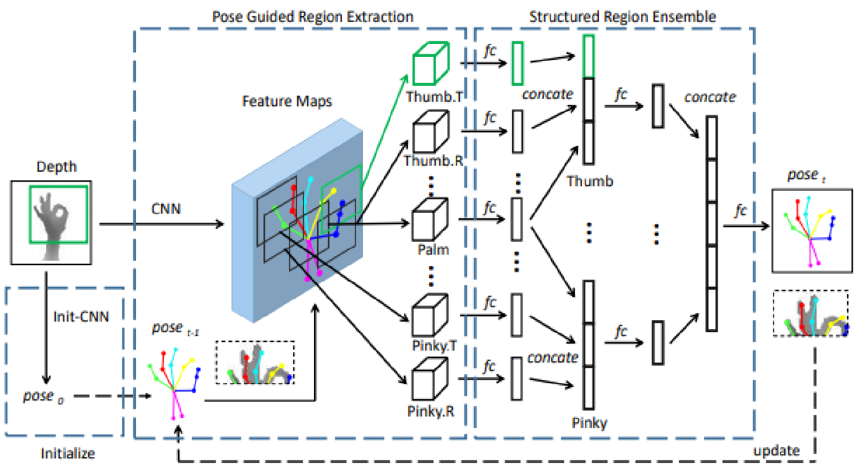


Fig. Pose-REN [1]

11 branches (2*5 fingers + palm)

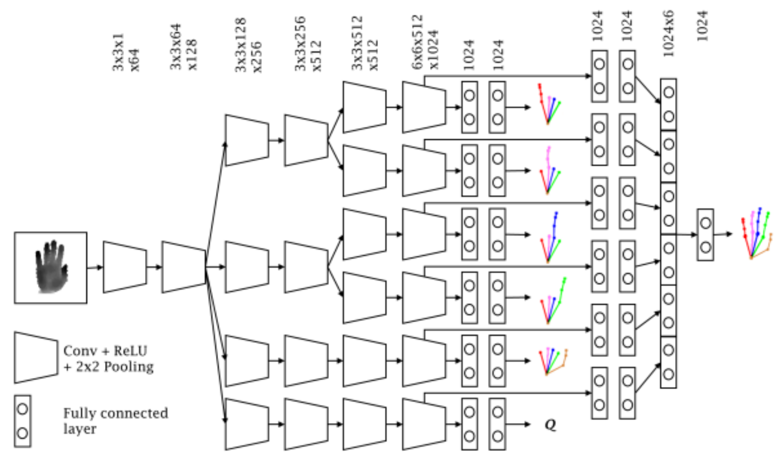


Fig. Local CNN [2]

6 branches (5 fingers + palm)

Related Work

- GCN-based Works
  - Use GCN-based methods to augment the local feature representation for dense prediction

  1) Each symbolic node receives votes from all local features (local-to-semantic voting)

     : Gray arrows

  2) After graph reasoning, evolved features are mapped back to each location : Purple arrows

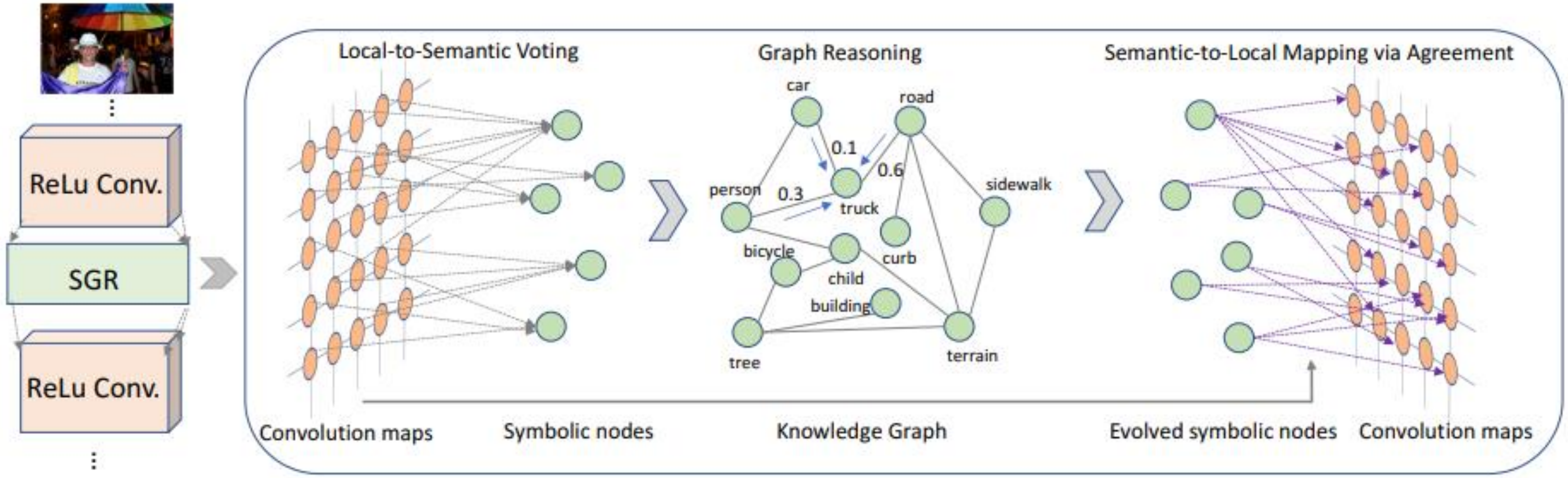


Fig. Overview of SGR layer [1]

Overview

- 3D HPE as dense pixel-to-offset predictions
  - input : depth image / output : joints' position (uv coordinates)
    1) Local feature extraction : use high-efficient hourglass network
       stack two hourglass to enhance learning power
    2) Joint Graph Reasoning module
    3) Pixel-to-offset module

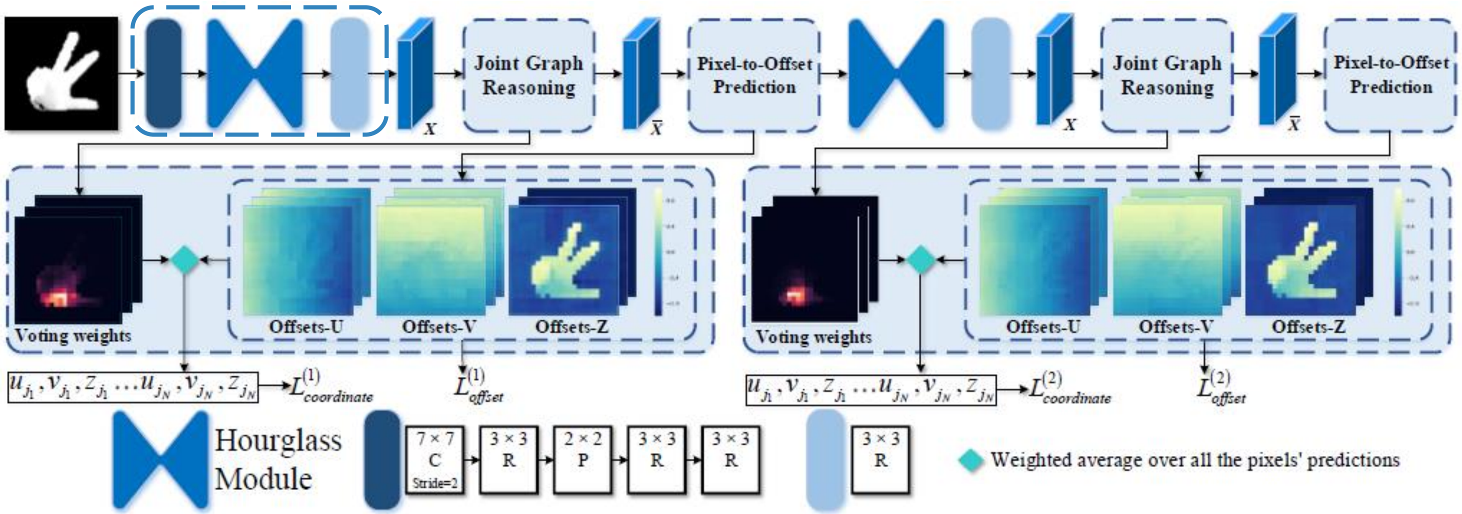
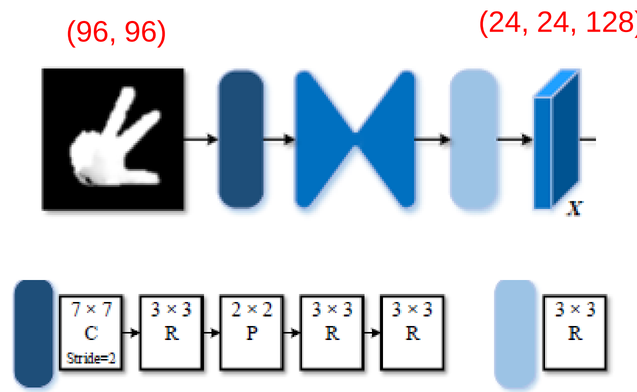


Fig. Overview of JGR-P2O

[1] Newell, A., et al. "Stacked Hourglass Networks for Human Pose Estimation." Springer 2016.

JGR-P2O

- Local Feature Extraction
  1) Cropped input 96 x 96 → 24 x 24 (hourglass in/output)
  2) Hourglass module [1]
     - Encoder – decoder structure
     - Short-cut elementwise addition structure



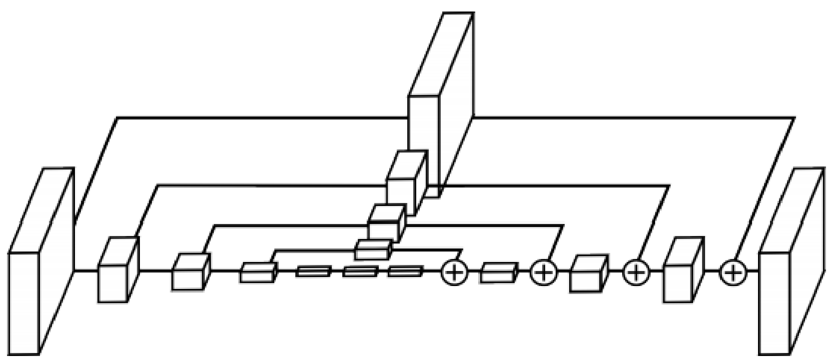


Fig. single hourglass module
Box is residual module

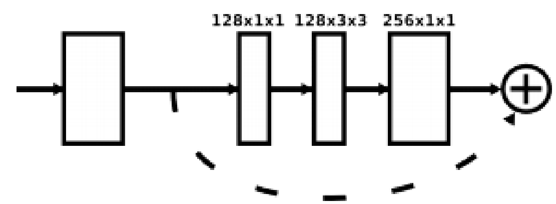


Fig. residual module

JGR-P2O

- GCN-based Joint Graph Reasoning Module

1) generate joints' features F by pixel-to-joint voting

- each joints is represented as the weighted average over all local features

➢ voting weights : $W = \Phi(\phi(X))$ $\phi(\cdot)$: 1x1 Conv. , Φ: spatial softmax normalization

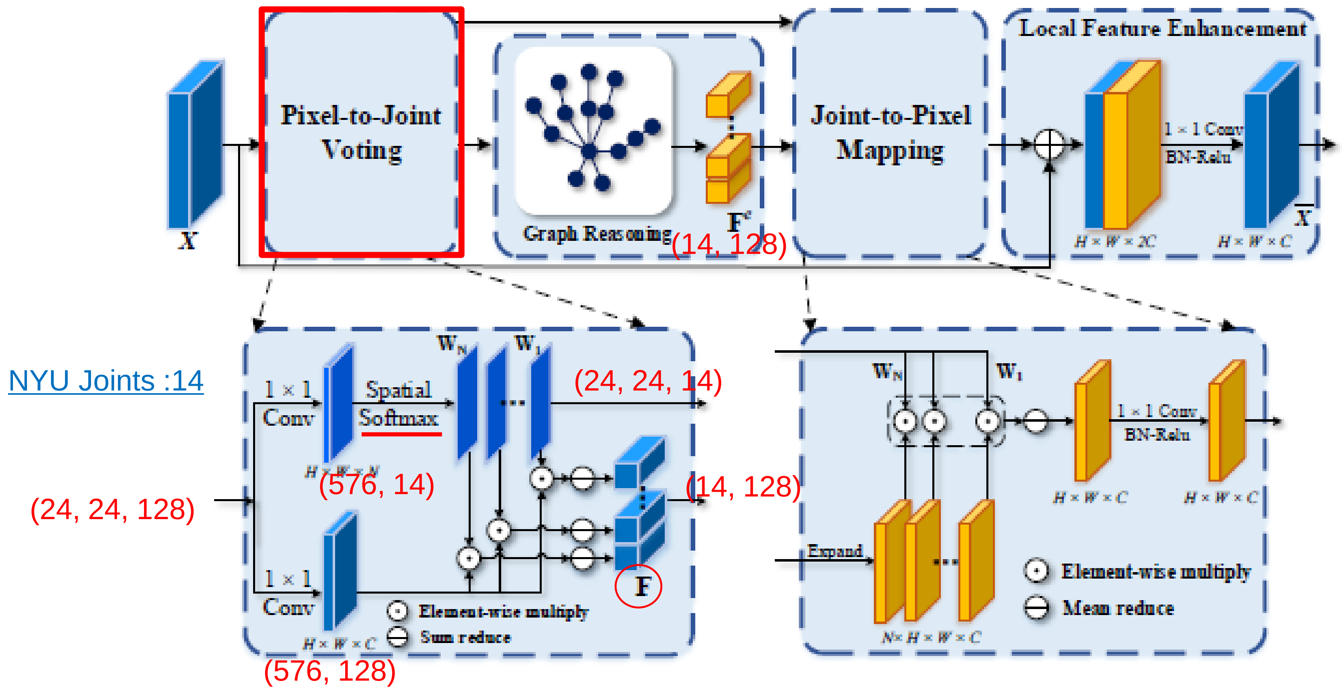


Fig. Flowchart of GCN-based joint graph reasoning module

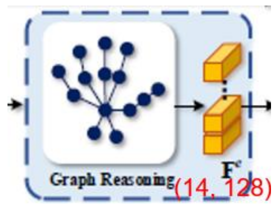
JGR-P2O

- GCN-based Joint Graph Reasoning Module

  2) define the connections between joints as a graph : model the dependencies among joints

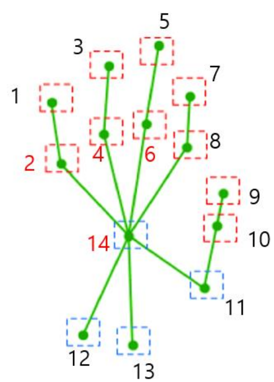
  map the joints' features to the corresponding graph nodes

  joint features are propagated within the graph by grape reasoning → enhanced joint feature F^e



where, I_N : identity matrix
$\tilde{D}$: diagonal node degree matrix of $(A + I_N)$

A (Adjacency Matrix) + I_N




```
args.dataset == 'NYU':
 A = np.array([[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
               [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
               [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
               [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
               [0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0],
               [0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1],
               [0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0],
               [0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1],
               [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0],
               [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0],
               [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1],
               [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1],
               [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1],
               [0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1]])
```

➢ Skeleton Graph Method [1]

$$\mathbf{A}^e = \tilde{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I}_N)\tilde{\mathbf{D}}^{-\frac{1}{2}}$$

A^e : Connection weight matrix

```
[[0.5,   0.408, 0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0],
 [0.408, 0.333, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.204],
          .
             .
                  .
                       .
```

$$\mathbf{F}^e = \underbrace{\sigma}_{\text{ReLU}}(\mathbf{A}^e \underbrace{\mathbf{F}\mathbf{W}^e}_{\text{FC layer}})$$

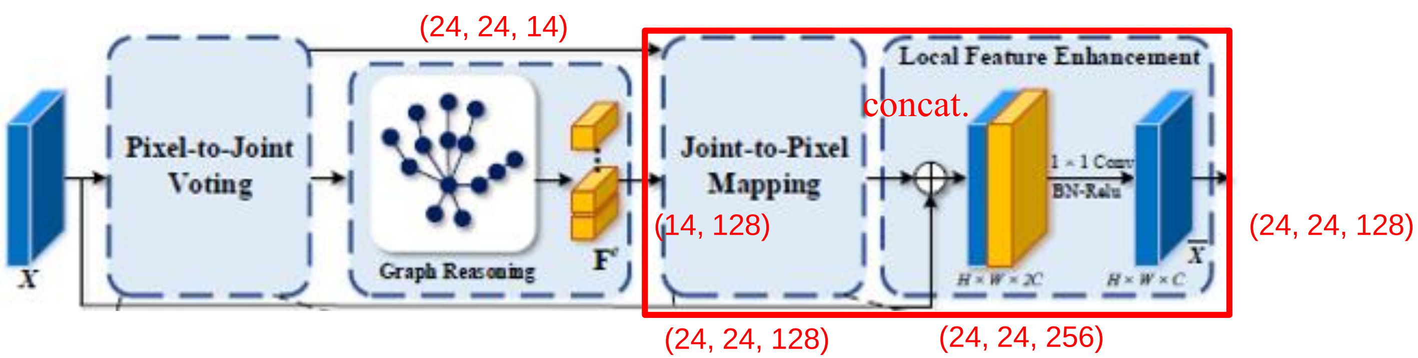
JGR-P2O

- GCN-based Joint Graph Reasoning Module

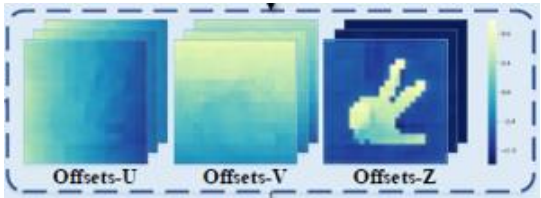
3) F^e is mapped back to local features by joint-to-pixel mapping (inverse of pixel-to-joint voting)

: generate the joint context representation for all pixels (24 x 24 x 128)

4) Original feature + joint context representation → obtain enhanced local feature



JGR-P2O



(24, 24, 14) (24, 24, 14) (24, 24, 14)

- Pixel-to-Offset Prediction Module
  - Depth image : 2D image plane cords + depth values (i.e. UVZ coords)
  - Joint's position is determined by pixel's UVZ coords and offset vector from pixel to a joint
  - Coords $(u_{j_k}, v_{j_k}, z_{j_k})$ of joint k is obtained by weighted average over all the pixel' prediction

$$\begin{cases} u_{j_k} = \sum_i w_{ki}\,(u_{p_i} + \Delta u_{ki}) \\ v_{j_k} = \sum_i w_{ki}\,(v_{p_i} + \Delta v_{ki}) \\ z_{j_k} = \sum_i w_{ki}\,(z_{p_i} + \Delta z_{ki}) \end{cases},$$

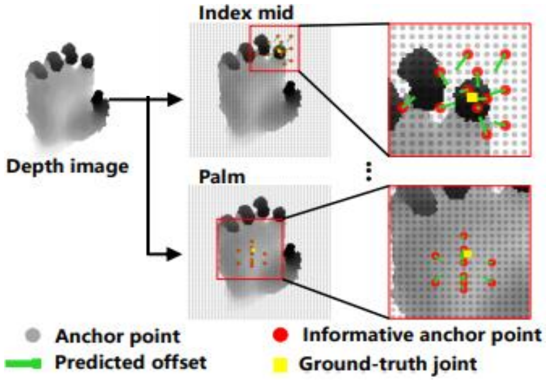
where, $(u_{p_i}, v_{p_i}, z_{p_i})$: UVZ coords of pixel p_i

$(\Delta u_{ki}, \Delta v_{ki}, \Delta z_{ki})$: predicted offset values from p_i to joint k

w_{ki} : normalized prediction weight of pixel p_i

  - implemented by 1x1 Conv. layer (input $\bar{X}$(24, 24, 128))
  - Simpler than A2J [1]

    : 2 branches (estimates UV and Z coords)



JGR-P2O

- Training Strategy

  1) Coordinate-wise regression loss : $L_{coordinate} = \sum_{k} \sum_{c} L_{\delta} \left(c_{j_k} - c^{*}_{j_k} \right),$

  2) Pixel-wise offset regression loss : $L_{offset} = \sum_{k} \sum_{i} \sum_{c} L_{\delta} \left(\Delta c_{ki} - \Delta c^{*}_{ki} \right),$

  - Huber(smooth *l*1) loss function : less sensitive to outlier (δ=0.01)

$$\text{smooth}_{\ell_1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 0.01, \\ 0.01(|x - 0.005|), & \text{otherwise,} \end{cases}$$

  3) Final loss (S=2, $\beta = 0.0001$) : $L = \sum_{s=1}^{S} L^{(s)}_{coordinate} + \beta L^{(s)}_{offset}.$

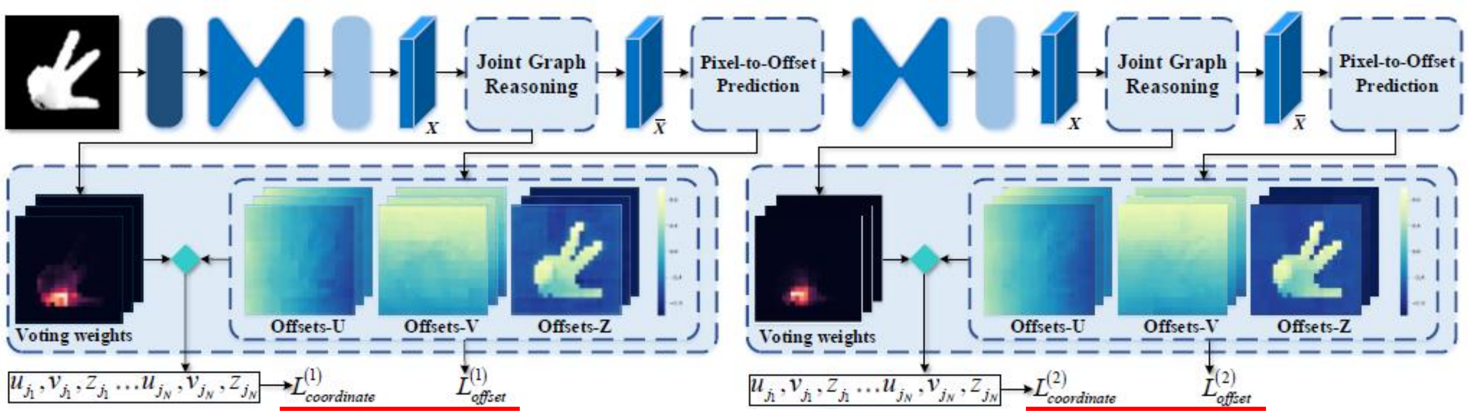
c_{jk} : predicted coords of joint k

c_{jk}^{*} : ground truth coords

L_{δ} : Huber loss function

Δc_{ki} offset value from pixel P_i

Δc^{*}_{ki} ground truth offset value



Experiments

- Two Metrics

  1) average 3D distance error (mm) between gt and predicted position

  2) percentage of succeeded frames whose errors for all joints are within a threshold

- Settings

  - Depth value are normalized to [-1, 1] for the cropped image
  - Adam optimizer
  - Batch size : 32
  - Online data augmentation : rotation [-180, 180]°, scaling [0.9, 1.1], translation [-10, 10]pixels
  - initial learning rate : 0.0001 (reduced by factor 0.96 every epoch)
  - train 58 epochs

Experiments

- Ablation Studies

1) Effectiveness of individual components

: Baseline (Backbone + P2O module)

- adding the pixel-wise offset loss and JGR module decreases the estimation error

Table. Effectiveness of individual components

Component			Mean error (mm)
P2O	Offset Loss	JGR	
✓			10.83
✓	✓		$10.54^{-0.29}$
✓	✓	✓	$\mathbf{8.29}^{-2.25}$

2) Number of hourglass modules

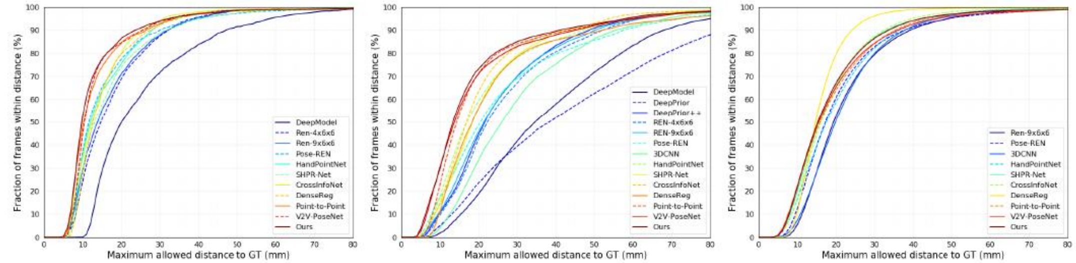
: Increasing the number of hourglass improves the estimation precision

but 3 hourglass can obtain negligible improvement → stack 2 hourglasses

Table. Comparison of different number of stacked hourglass module

#Hourglasses	Mean error (mm)	#Params
1	8.63	0.72M
2	8.29	1.37M
3	8.27	2.02M

Experiments



- Comparison with state-of-the-art
  - the proportions of success frames are highest
  - dense prediction-based methods are generally superior to direct regression-based methods
  - the lowest mean estimation errors
  - has the minimum model size
  - fast running speed : 111.2 fps (NVIDIA 1080Ti GPU)

Method	Mean error (mm)			Type	#Params	Speed (fps)
	ICVL	NYU	MSRA			
DeepModel[50]	11.56	17.04	-	DR	-	-
DeepPrior[26]	10.40	19.73	-	DR	-	-
DeepPrior++[25]	8.10	12.24	9.50	DR	-	30.0
REN-4x6x6[12]	7.63	13.39	-	DR	-	-
REN-9x6x6[12]	7.31	12.69	9.70	DR	-	-
Pose-REN[2]	6.79	11.81	8.65	DR	-	-
3DCNN[9]	-	14.1	9.60	DR	104.9M	215
HandPointNet[7]	6.94	10.54	8.50	DR	2.58M	48.0
SHPR-Net[3]	7.22	10.78	7.76	DR	-	-
CrossInfoNet[6]	6.73	10.08	7.86	DR	23.8M	124.5
DenseReg[43]	7.30	10.2	**7.20**	DP	5.8M	27.8
Point-to-Point[10]	6.30	9.10	7.70	DP	4.3M	41.8
V2V-PoseNet[22]	6.28	8.42	7.59	DP	457.5M	3.5
Point-to-Pose Voting[17]	-	8.99	-	DP	-	80.0
A2J[45]	6.46	8.61	-	DP	44.7M	105.1
JGR-P2O(Ours)	**6.02**	**8.29**	7.55	DP	1.4M	111.2

DR : Direct Regression-based
DP : Dense Prediction-based

Thank You