2020.11.28

Unsupervised Depth Estimation, 3D Face Rotation and Replacement

Joel Ruben Antony Moniz1* , Christopher Beckham2,3* ,
Simon Rajotte2,3 , Sina Honari2 , Christopher Pal2,3,4

20151739 공대현

❖ 요약

- 깊이 정보 없이 (Unsupervised), 3D 시점 변환을 해주는 방식
- 3D affine 변환 행렬 Parameter를 Back Propagation을 이용하여 학습시키는 방식
- Face Pose Frontalization이나, 원하는 포즈로 바꿔주는 3D transformation matrix predict
- Face Rotation과 2D image부터 Face Warping 하는 새로운 비지도 학습 기술

❖ 이 논문의 Contribution

1. 얼굴 특징 점들의 깊이와 3D affine Transformation Parameter를 동시에 추출해주는 NN 사용

2. 추론된 깊이와 Closed form solution을 이용하여, Source와 Target keypoint의 pseudoinverse Transformation을 return함

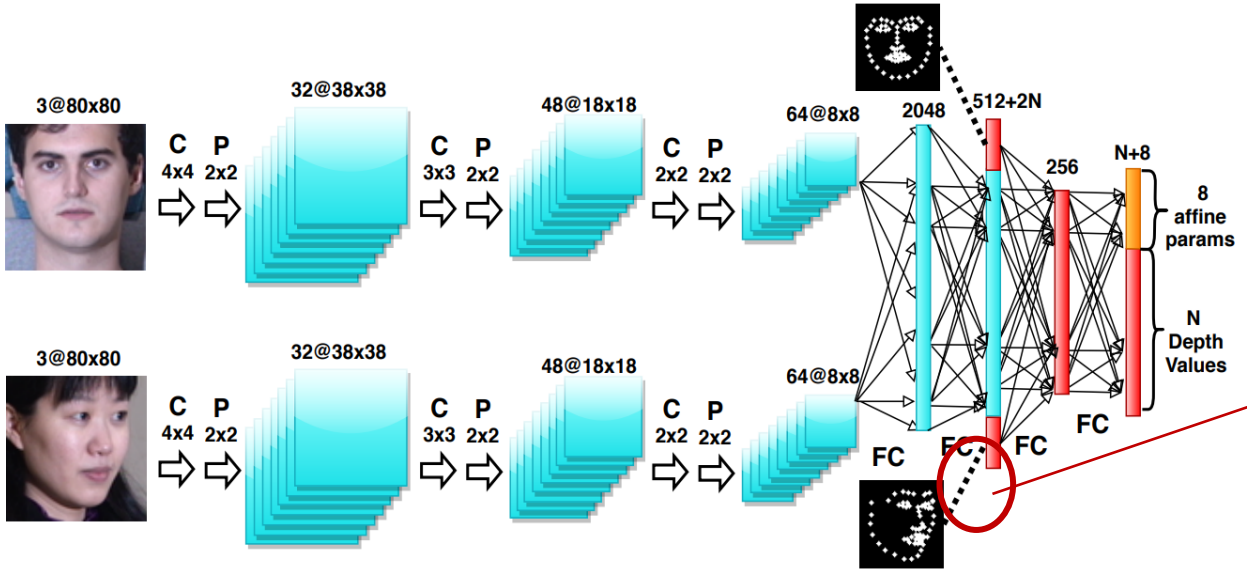
3. Face rotation을 위한 Ground-Truth of 3D images, Depth 없이 학습이 가능한 최초의 NN

1. DepthNets

2가지의 Architecture Scenario가 있음.

Scenario (A) : Siamese('샴 쌍둥이 할 때 샴) Architecture
- RGB source, target 이미지로부터 특징점들을 추출

Scenario (B) : Source, Target Image 의 특징점들을 이용하는 FCN Architecture



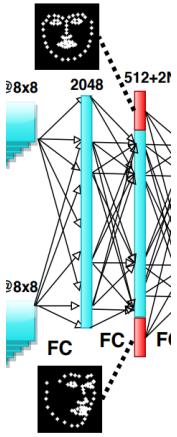
-> 파란색 과정은 (A)에서만 사용됨 , 빨간 부분은 (A),(B) 모두에서 사용됨

Keypoint 값을 FC units으로 Concatenating 한 것.

이 논문에서는 Face data를 사용했지만, 비슷한 조건의 다른 Object에 쓸 수 있다.

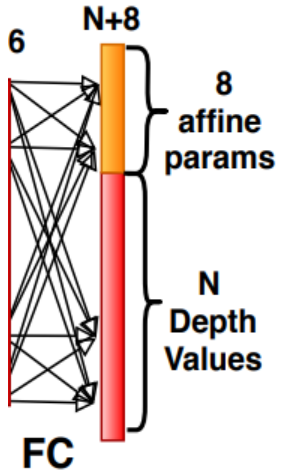
1. DepthNets

Input



Geometry,
Source와 Target의
포즈(2D 특징점들)

Output



Depth와
Viewpoint geometry

2. Sequence of Steps

1) Keypoint extraction

RCN(Recombinator Network) 으로 Keypoint 추출

2) Keypoint processing

N(keypoints)+8(transformation matrix parameters)를 Dense layer 생성

3) Geometric Affine Transformation Normalizer

예측된 affine Transformation matrix로 source에서 target으로 변환

$$\begin{bmatrix} x_n^i \\ y_n^i \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_3 & t_x \\ m_4 & m_5 & m_6 & t_y \end{bmatrix} \begin{bmatrix} x_s^i \\ y_s^i \\ z_p^i(\boldsymbol{I_s}, \boldsymbol{I_t}) \\ 1 \end{bmatrix}$$

x_n, y_n: affine transformation matrix에 의해 normalized된 keypoint

x_t, y_t: Ground Truth keypoint

z_p: input keypoints로부터 예측된 depth값, source와 target keypoints로부터 생성되므로 $z_p(I_s, I_t)$

$F(I_s, I_t)$ 는 2x4 affine transformation matrix를 뜻함.

$$\mathcal{L} = \sum_{i=1}^{K} \left\| \mathbf{x}_t^i - \mathbf{F}(\boldsymbol{I_s}, \boldsymbol{I_t}) \, [x_s^i \;\; y_s^i \;\; \mathbf{z_p^i}(\boldsymbol{I_s}, \boldsymbol{I_t})]^T \right\|^2$$

2. Sequence of Steps

4) Image Warper

최종적인 2d 이미지를 출력 할 때는 depth proxy를 projection하는데 그냥 z를 drop시킨다.

-> Source Image, keypoints, affine matrix를 "*OpenGL*"에 input으로 넣어 target pose에 맞는 warp된 이미지를 얻는다.
단, OpenGL은 image를 warp시키는 도구일 뿐, 학습시키는 과정에 포함되지 않는다.

2.2 Estimationg Viewpoint Geometry

Test 할 때 Estimate 방법

Training 할 때는 2.1처럼 model이 depth와 3d affine transformation parameters를 output하지만, Test 할 때는 Predicted된 Face끼리의 affine matrix를 사용하지 않고, depth만 사용한다.

3D points 와 그에 대응하는 target geometry의 2D points가 주어진다면 **linear least squares estimation** 으로 matrix parameters를 구할 수 있다.

$$\begin{bmatrix} x_s^1 & y_s^1 & z_s^1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & x_s^1 & y_s^1 & z_s^1 & 0 & 1 \\ x_s^2 & y_s^2 & z_s^2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & x_s^2 & y_s^2 & z_s^2 & 0 & 1 \\ & & & & \vdots & & & \\ x_s^K & y_s^K & z_s^K & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & x_s^K & y_s^K & z_s^K & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ m_5 \\ m_6 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x_t^1 \\ y_t^1 \\ x_t^2 \\ y_t^2 \\ \vdots \\ x_t^K \\ y_t^K \end{bmatrix} \longrightarrow \boldsymbol{m} = [\boldsymbol{A}^T \boldsymbol{A}]^{-1} \boldsymbol{A}^T \boldsymbol{x}_t,$$

Psuedo-inverse 식

linear least squares estimation

2.3 Viewpoint와 Depth의 접합

핵심은 loss function으로 구한 근본적인 matrix 대신에 closed form solution으로도 matrix를 구할 수 있다는 것:

→ $\boldsymbol{m} = [\boldsymbol{A}^T\boldsymbol{A}]^{-1}\boldsymbol{A}^T\boldsymbol{x}_t,$ *(Eq.3)*

$$\sum_{i=1}^{K} \left\| \underbrace{\begin{bmatrix} x_t^i \\ y_t^i \end{bmatrix}}_{\boldsymbol{x}_t^i} - \underbrace{\begin{bmatrix} m_1 & m_2 & m_3 & t_x \\ m_4 & m_5 & m_6 & t_y \end{bmatrix}}_{\boldsymbol{m}} \underbrace{\begin{bmatrix} x_s^i \\ y_s^i \\ z_p^i(\boldsymbol{I_s}, \boldsymbol{I_t}) \\ 1 \end{bmatrix}}_{\boldsymbol{x}_s^i} \right\|^2 = \sum_{i=1}^{K} \left\| \boldsymbol{x}_t^i - \text{reshape}\left[[\boldsymbol{A}^T\boldsymbol{A}]^{-1}\boldsymbol{A}^T\boldsymbol{x}_t\right]\boldsymbol{x}_s^i \right\|^2$$

여기서 Model이 train과 test time동안 depth 값을 출력
Affine transformation matrix는 Eq.3으로 대체.
Section 2.1,2.2와 가장 큰 차이는 m이 더 이상 DepthNet에 의해 predicted 되지 않음.

$z_s^i = \mathbf{z_p^i}(\{x_s^j, y_s^j, x_t^j, y_t^j\}_{j=1\ldots N})$ 이므로 analytical soultion애기 때문에 z_s를 backpropagation 할 수 있음.

→ Depth 값을 Ground Truth없이 비지도 학습으로 학습 가능!

2.4 Adversarial Image-to-Image Transformation

Face를 Target으로 변형해주기 때문에, Background가 불가피하게 missing됨

Face Swap 이나 Frontalization 후, Cycle-GAN을 사용.

Cycle-GAN은 두 도메인 사이의 이미지 변형에서 paired data가 필요하지 않음.

3. 실험 결과

Model	Color	MSE	MSE_norm
1) A simple 2D affine registration	grey	1.562	9.547
2) A 3D affine registration model using an average 3D face template	purple	0.724	7.486
3) A DepthNet that separately estimates depth and geometry	brown	0.568	6.292
4) The model above, but with a Siamese CNN image model	violet	0.539	6.115
5) Secondary least squares estimation for visual geometry using the depths from 3)	red	0.400	5.184
6) Secondary least squares estimation for visual geometry using the depths from 4)	green	0.399	5.175
7) Backpropagation through the pseudoinverse based solution for visual geometry	orange	0.357	4.932
8) The model above, but with a Siamese CNN image model	blue	**0.349**	**4.891**

Violence

Pseudo inverse 로 Backpropagation+image CNN 한 게 가장 error가 적었다. (Section 2.3 방식)

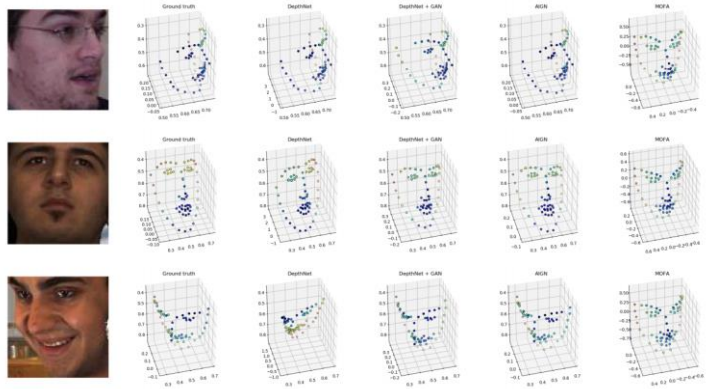
3.2 Unpaired Faces에 대한 다른 모델과의 비교

3DFAW dataset으로 다른 identity에 대해 테스트해 봄 (unpaired)
이 데이터셋은 3d keypoint에 대한 GT가 있음.

Model	Need Depth	Manual Init.	MSE ($\times 10^{-5}$)	Depth Correlation Matrix Trace (DepthCorr)		
				Left pose	Front pose	Right pose
GT Depth	Yes	-	8.86 ± 6.55	**66**	**66**	**66**
AIGN [24]	Yes	No	9.06 ± 6.61	44.08	50.81	49.04
MOFA [20]	No	Yes	8.75 ± 6.33	11.14	15.97	17.54
DepthNet (Ours)	No	No	$\mathbf{7.65 \pm 6.97}$	27.36	26.32	22.34
DepthNet + GAN (Ours)	Yes	No	8.74 ± 6.24	59.69	58.68	59.76

Table 3: Comparing MSE and DepthCorr for different models. A lower MSE indicates the model maps better to the target faces. A higher DepthCorr indicates more correlation between estimated and GT depths.

DepthNet 이 Label 없이도 MSE가 가장 낮은 것을 확인할 수 있음



→ DepthNet이 Depth estimation에 대한 정확도는 뒤쳐지지만,
최종 결과물(source to target)에 대한 정확도는 가장 높음

3.3 Face Rotation and Adversarial Repair

Depthnet+OpenGL로 frontalized 된 정면이미지를, 원래 이미지가 배경이 되도록 CycleGAN을 사용

비슷한 방법으로 Face swap을 할 수 있다.

4. DepthNet의 장점

어떤 포즈, 어떤 geometry더라도 mapping 가능

Depth 정보에 직간접적으로 의존하지 않는다.

Manual initialization이 필요없다.

5. Experimental Setup

RCN , DepthNet, CycleGAN 모델 따로 학습함
OpenGL은 test time때 rendering 할 때만 쓰임.

1. Face detector로 80x80으로 face crop
2. RCN 으로 68(N) keypoints 획득
3. DepthNet 최종 output = N + 8(matrix)

OpenGL을 사용할때는 source로 frontal face를 사용하는 게 가장 quality가 좋음

Cycle GAN 사용법

X -> Y 로 transformation 할 때,
X : DepthNet-Resulting Face
Y : Ground Truth Face which is frontal

Background Synthesis에선,
X: DepthNet-frontalized face 와 original image의 background

FaceSwap에선,
X: target image face위에 warp된 source를 포함하는 이미지

Dataset: VGG,CelebA dataset