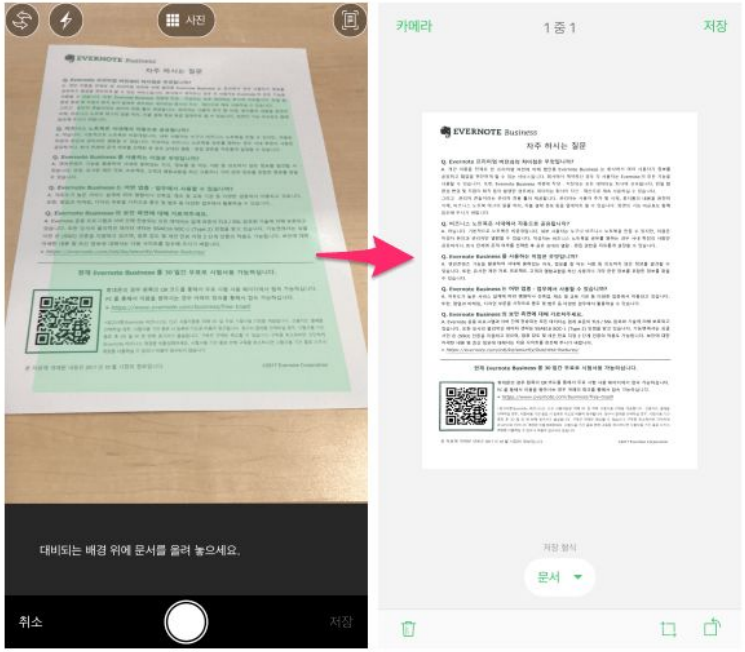
Image Warping Deep Learning

SNU AI, Yoonseo Kim

7, Sep. 2021

Applications



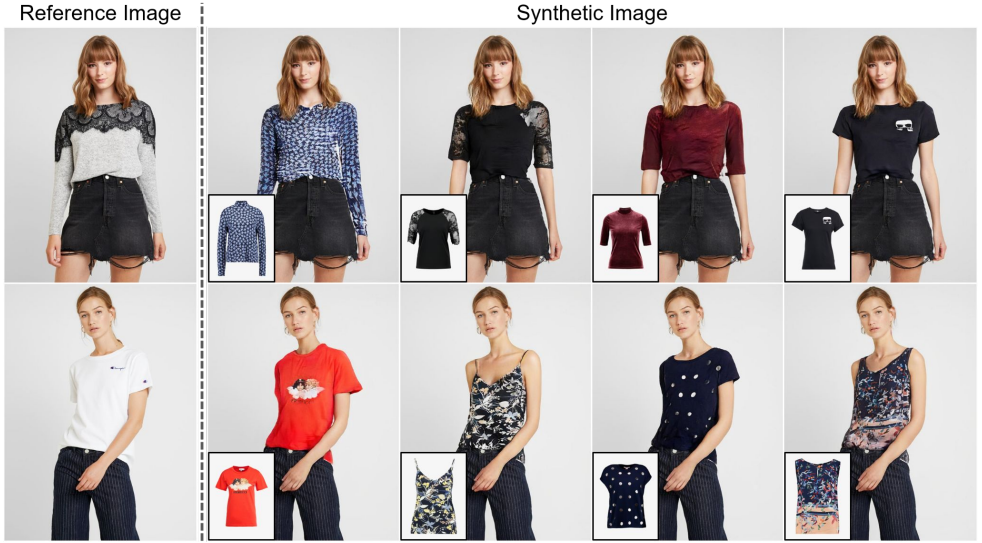
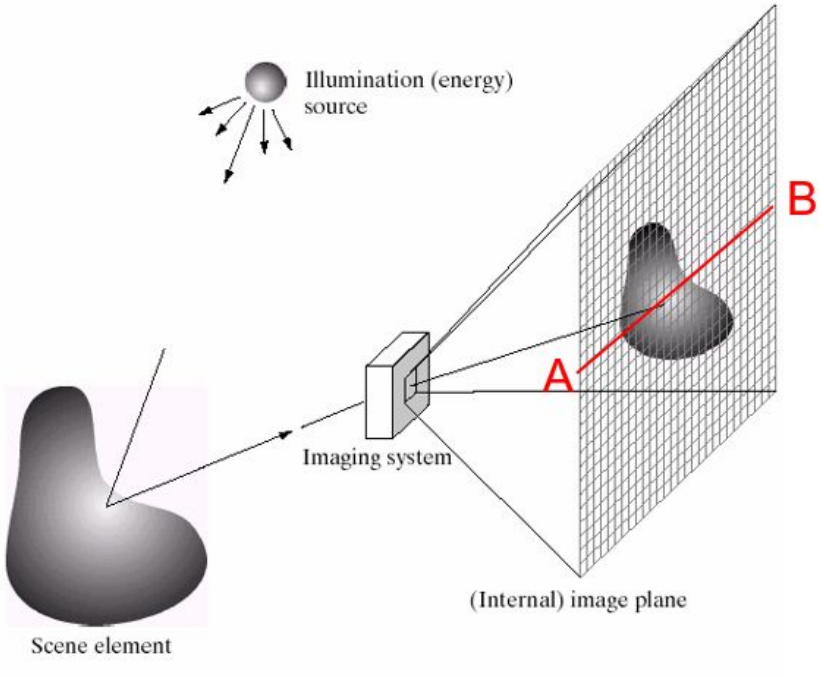
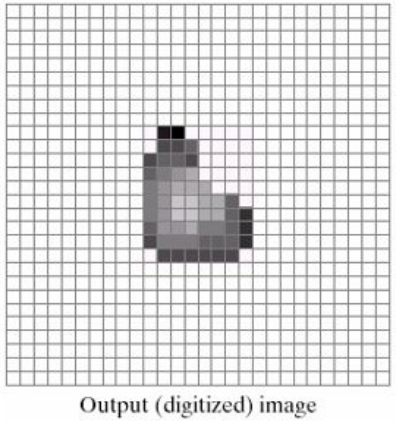


Image Formation




Output (digitized) image

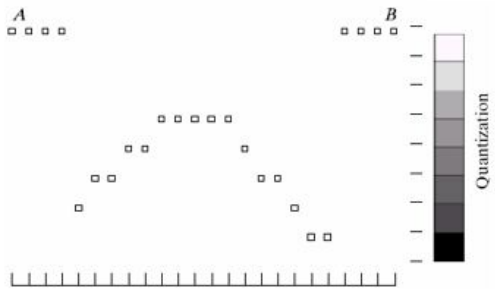


Image Formation

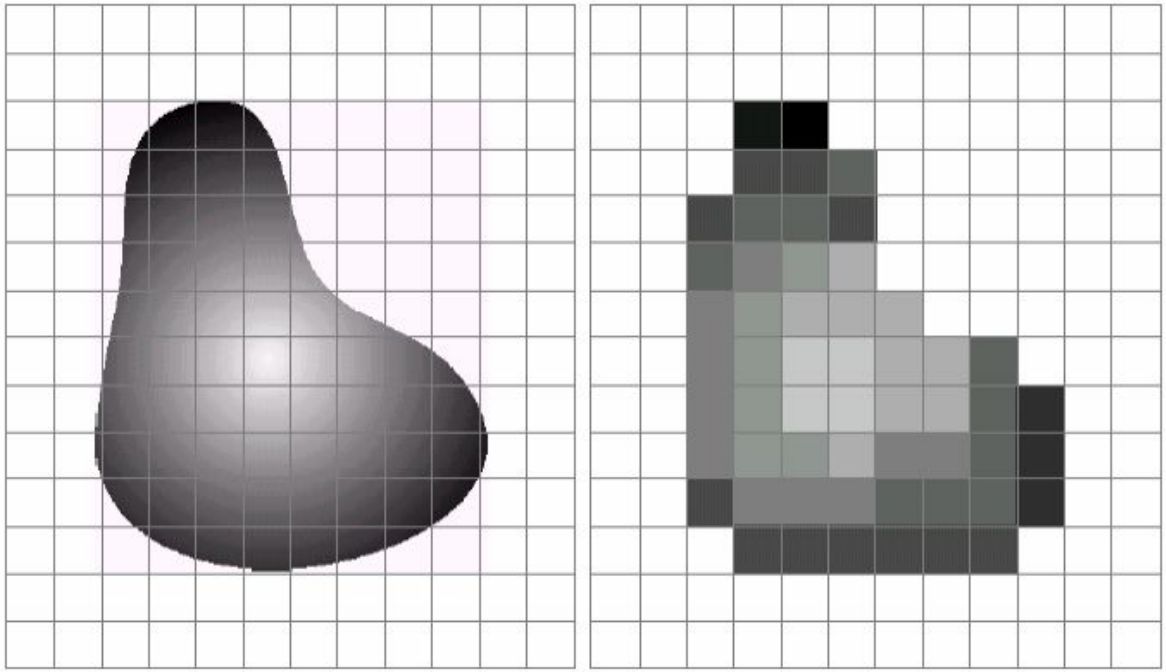


Image as a Function

$$f : R^2 \rightarrow R$$

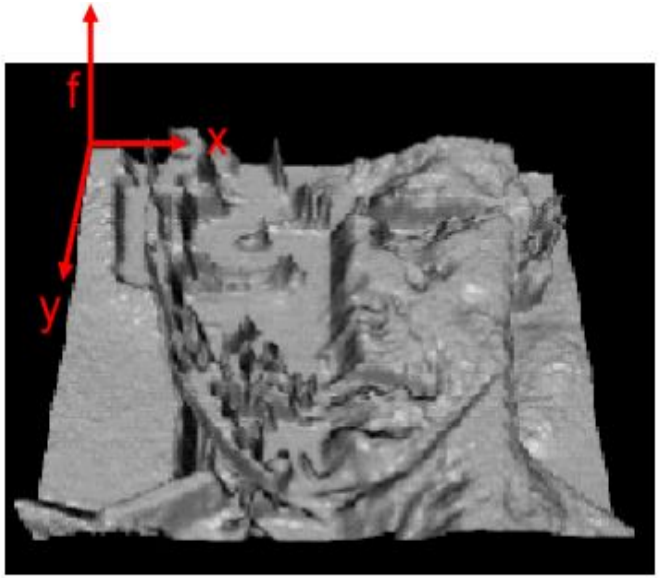




Image Filtering & Image Warping

Image Filtering : 치역 (Range)

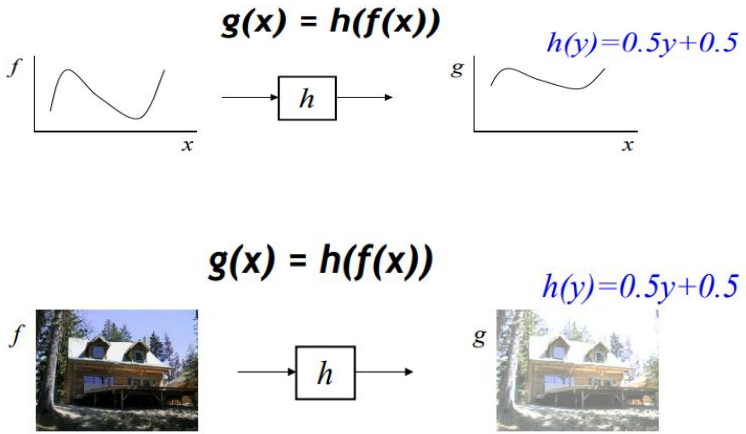


Image Warping : 정의역 (Domain)

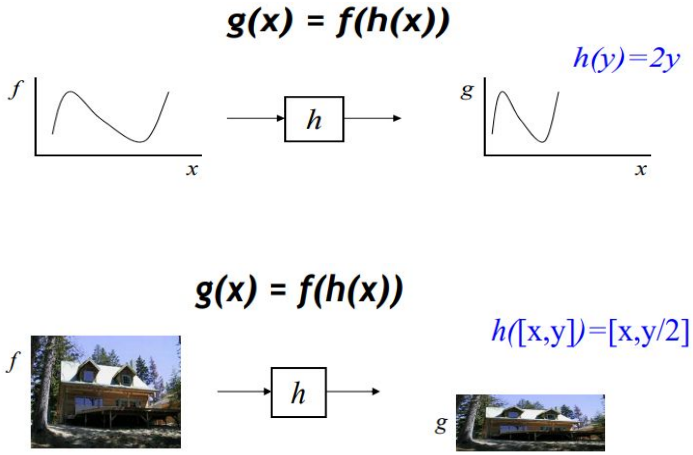


Image Warping is Transformation

- Image Warping = (x, y) → (x', y')
- Transformation = 두 벡터 공간 사이의 Mapping

- $T: \mathbf{x} \in \mathbb{R}^n \mapsto \mathbf{y} \in \mathbb{R}^m$: Mapping n-dim vector to m-dim vector
- Example:

$$T: \mathbf{x} \in \mathbb{R}^3 \mapsto \mathbf{y} \in \mathbb{R}^2 \quad \mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \in \mathbb{R}^3 \quad \mapsto \quad \mathbf{y} = T(\mathbf{x}) = \begin{bmatrix} 4 \\ 5 \end{bmatrix} \in \mathbb{R}^2$$

Global Warping (Parametric)



translation



rotation



aspect



affine

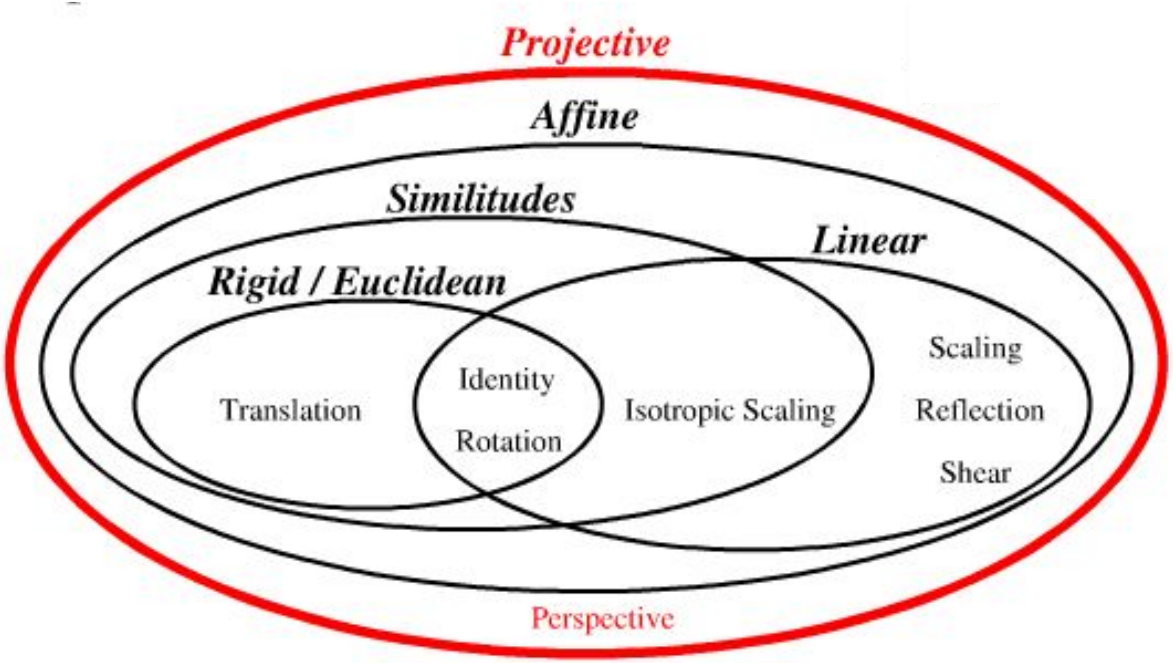


perspective

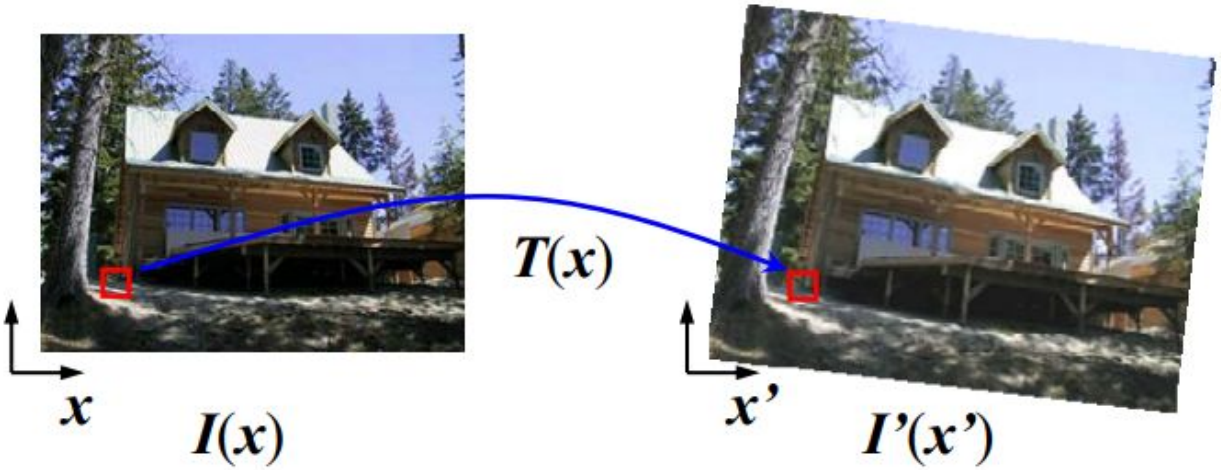


cylindrical

Transformations for parametric warping

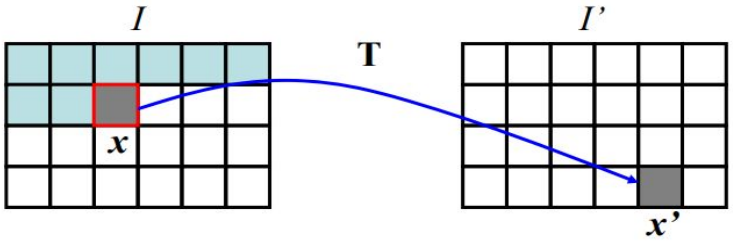


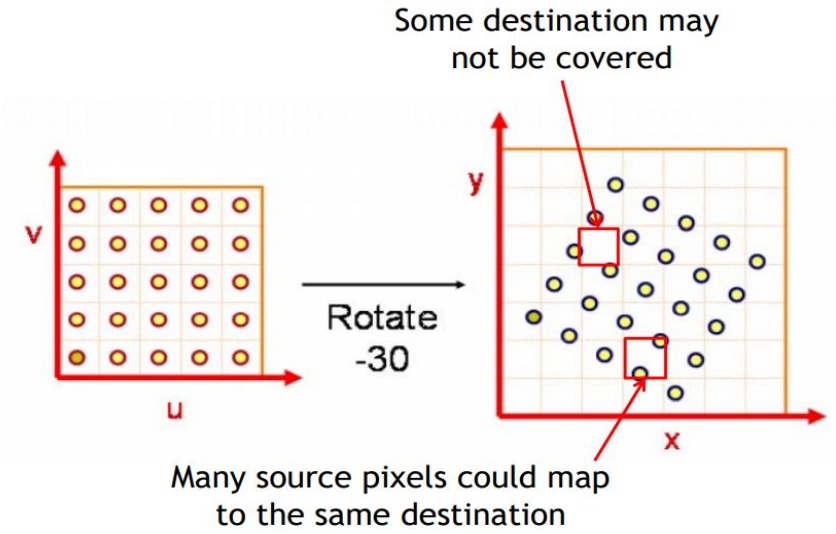
와핑된 이미지를 얻는 방법



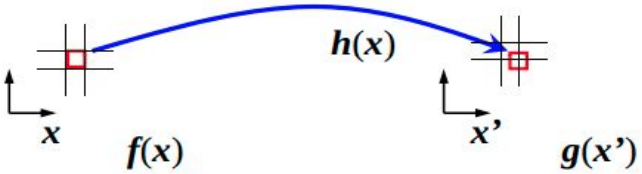
Forward Warping

```
fwarp(I, I', T)
{
  for (y=0; y<I.height; y++)
    for (x=0; x<I.width; x++) {
      (x',y')=T(x,y);
      I'(x',y')=I(x,y);
    }
}
```

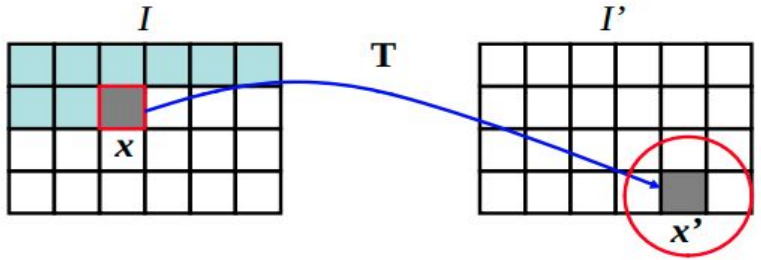




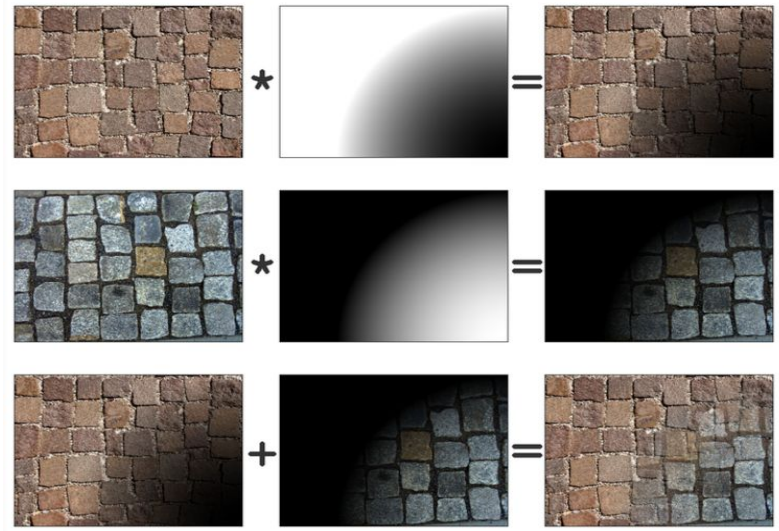
Forward Warping

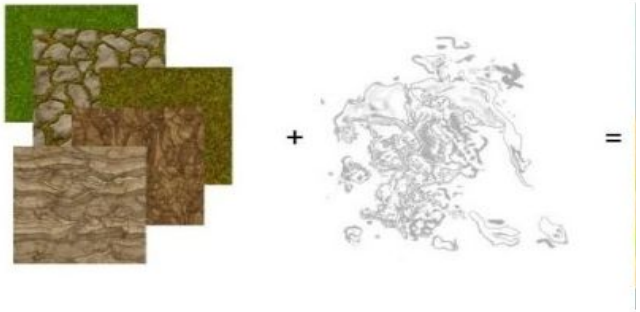



```
fwarp(I, I', T)
{
  for (y=0; y<I.height; y++)
    for (x=0; x<I.width; x++) {
      (x',y')=T(x,y);
      Splatting(I',x',y',I(x,y),kernel);
    }
}
```



(Texture) Splatting



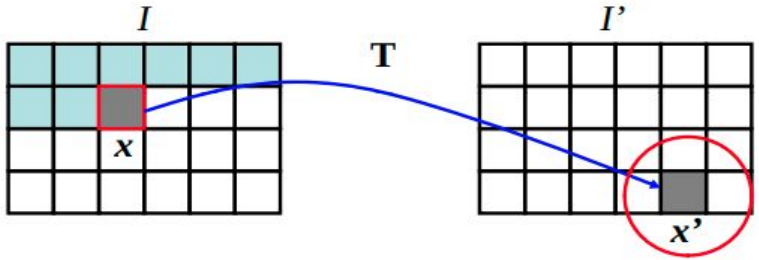


텍스처들 (스플랫)　　알파 맵



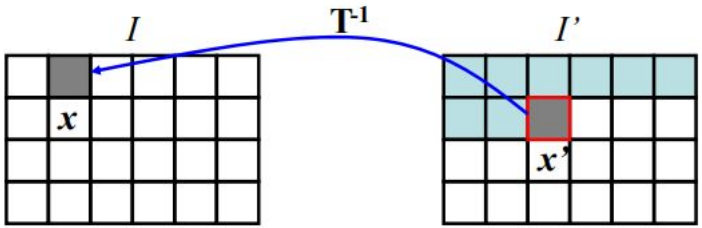
(Texture) Splatting

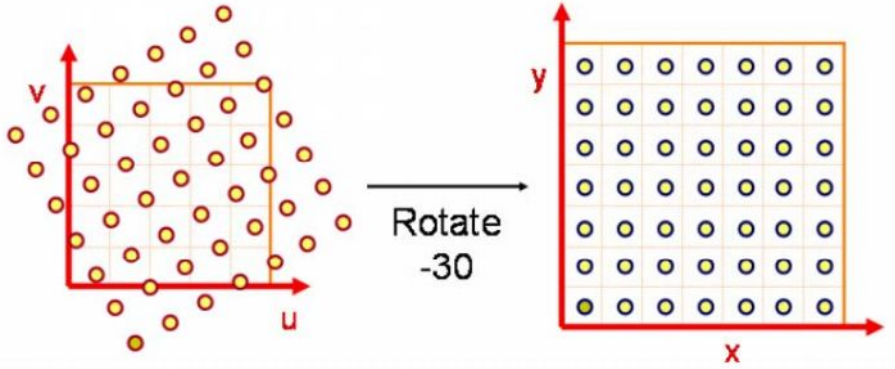
```
fwarp(I, I', T)
{
  for (y=0; y<I.height; y++)
    for (x=0; x<I.width; x++) {
      (x',y')=T(x,y);
      Splatting(I',x',y',I(x,y),kernel);
    }
}
```



Inverse Warping

```
iwarp(I, I', T)
{
  for (y=0; y<I'.height; y++)
    for (x=0; x<I'.width; x++) {
      (x,y)=T^-1(x',y');
      I'(x',y')=I(x,y);
    }
}
```

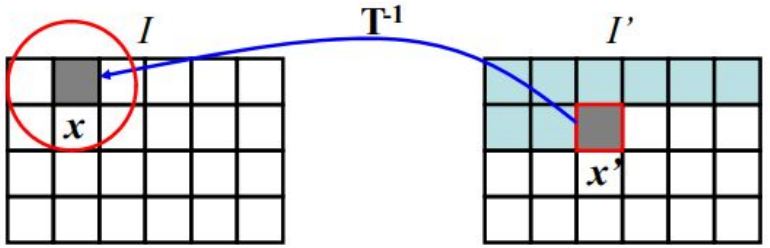




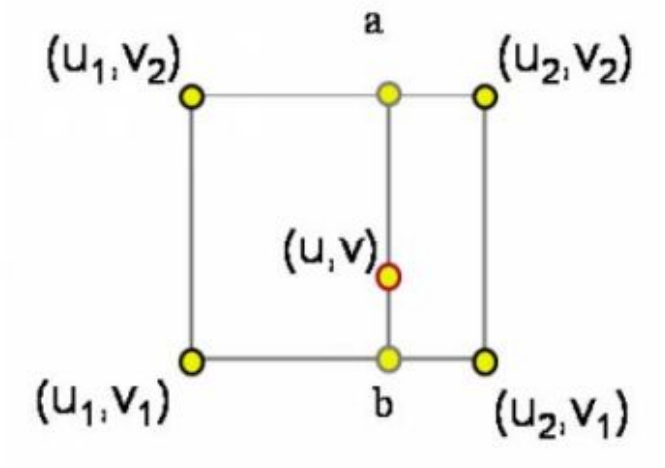
Inverse Warping

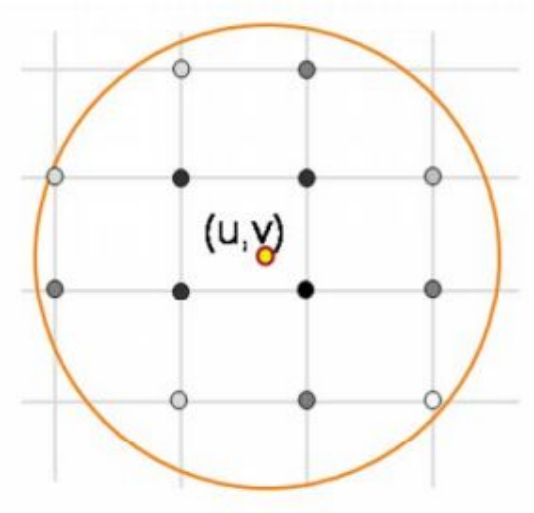



```
iwarp(I, I', T)
{
  for (y=0; y<I'.height; y++)
    for (x=0; x<I'.width; x++) {
      (x,y)=T^-1(x',y');
      I'(x',y')=Reconstruct(I,x,y,kernel);
    }
}
```



Reconstruction of Inverse Warping



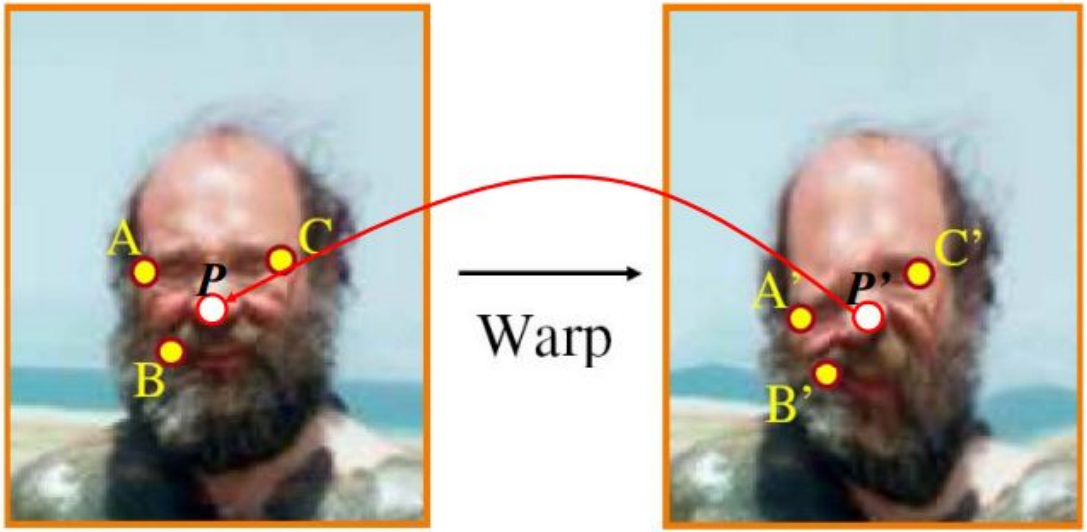


Local Warping (Non-parametric)

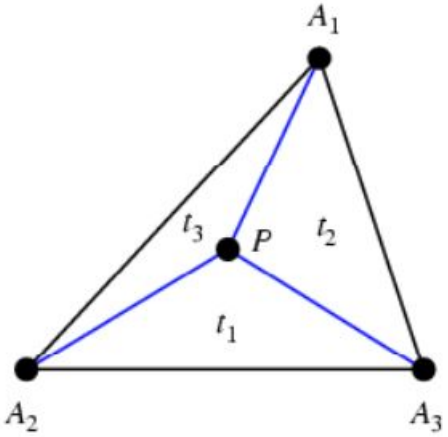
$$P = w_A A + w_B B + w_C C$$

$$P' = w_A A' + w_B B' + w_C C'$$

Barycentric coordinate



Local Warping (Non-parametric)

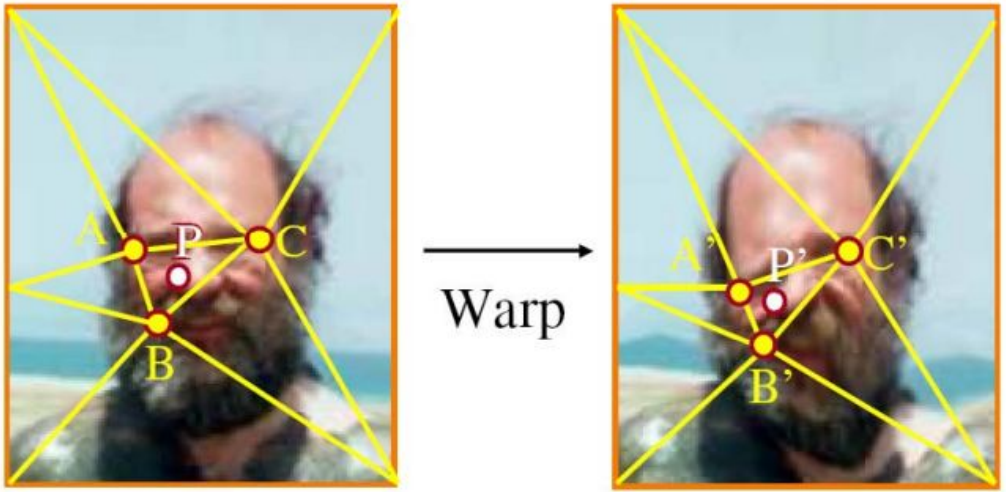


$$P = t_1 A_1 + t_2 A_2 + t_3 A_3$$
$$t_1 + t_2 + t_3 = 1$$

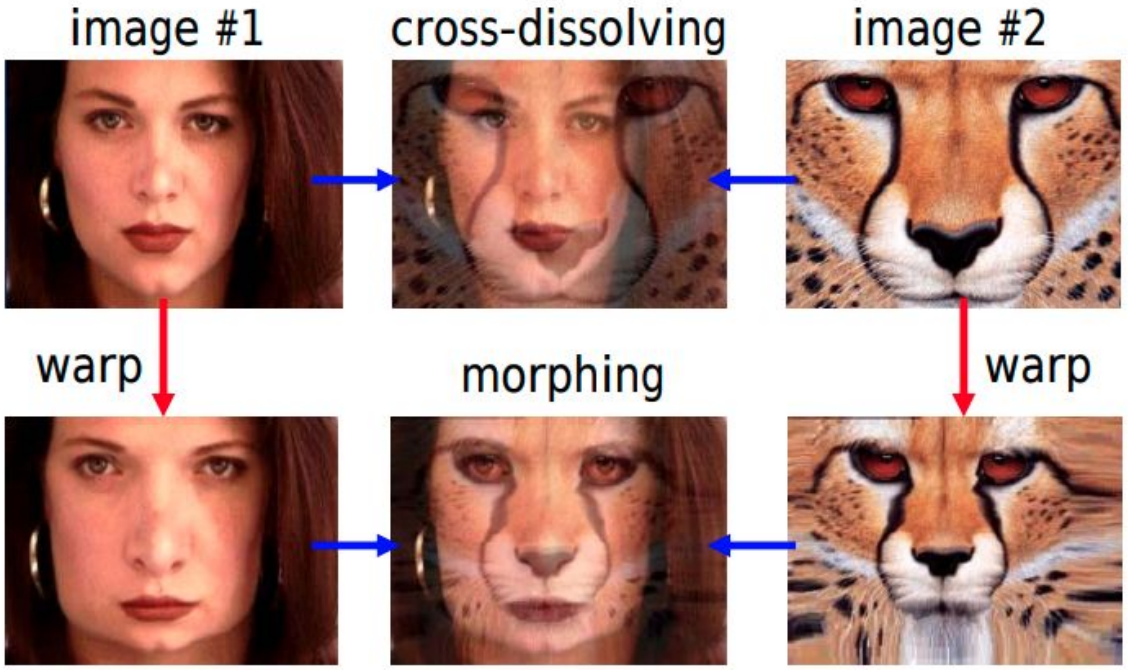
$$P = w_A A + w_B B + w_C C$$

$$P' = w_A A' + w_B B' + w_C C'$$

Barycentric coordinate

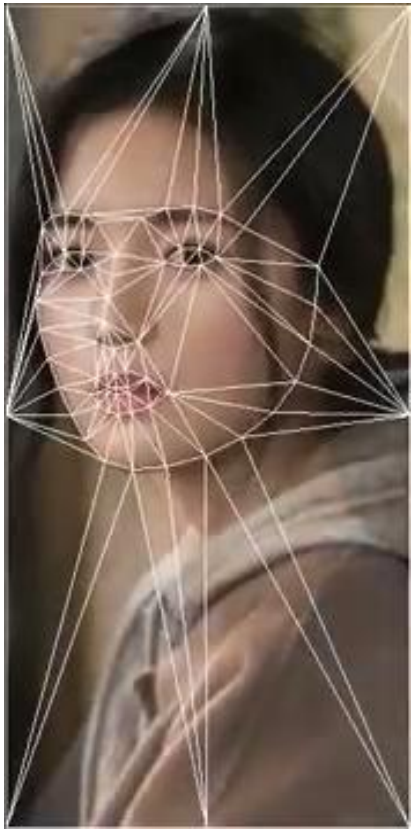


Morphing



Face Morphing





Cross Dissolve (Photometric)

$$M_t(x,y) = (1-t) * I_0(x,y) + t * I_1(x,y)$$







Ghosting Effect (t=0.5)

Cross Dissolve with Geometric transformation

- Compute the intermediate shape by linear interpolation of each feature pair

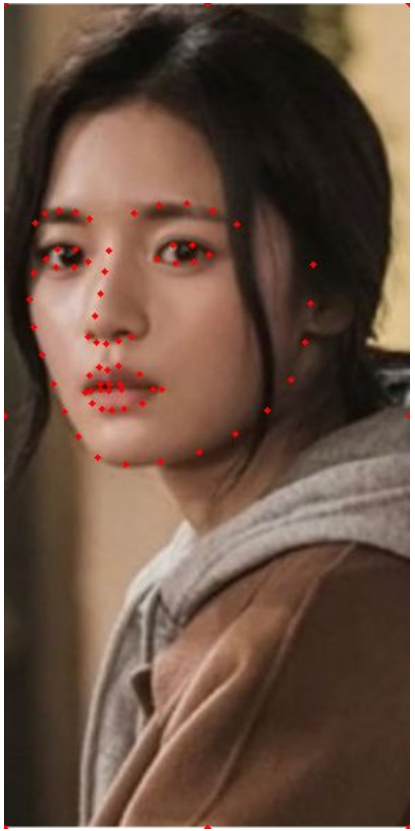
$$x_t = (1 - t) * x_i + t * x_j$$
$$y_t = (1 - t) * y_i + t * y_j$$

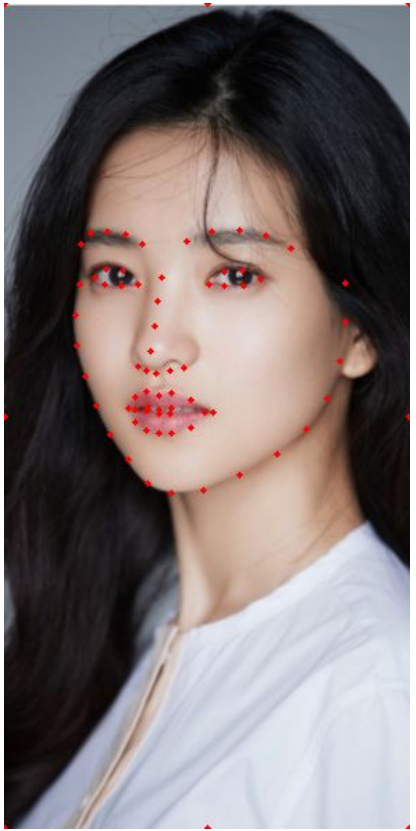
- Cross dissolve the color by interpolating two images

$$M(x_t, y_t) = (1 - t) * I_0(x_i, y_i) + t * I_1(x_j, y_j)$$

Finding Keypoints

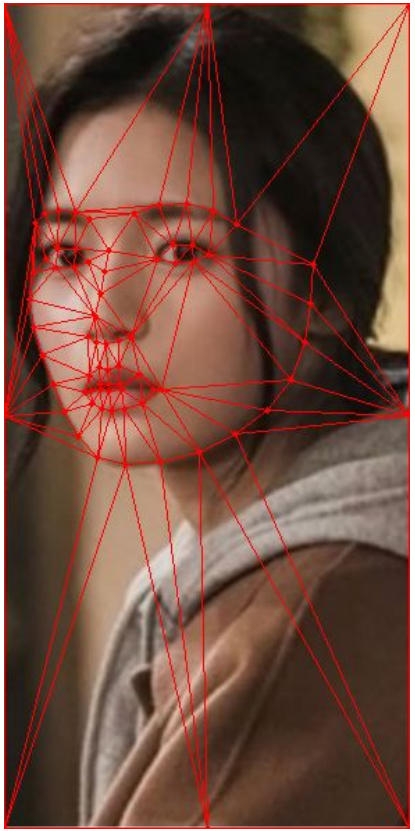
Detecting facial landmarks : 76 (68 + 8) points

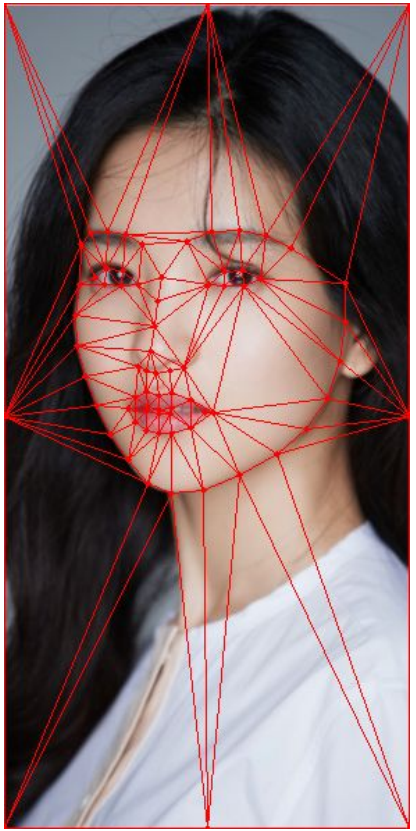




Warping Triangular Mesh

Finding triangle-to-triangle correspondences





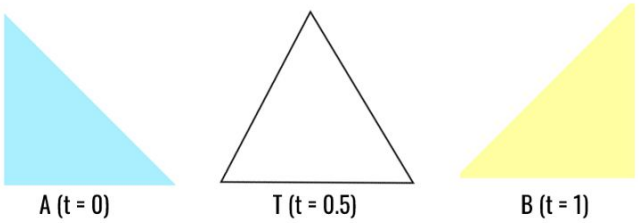
Feasibility of warping triangular mesh

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

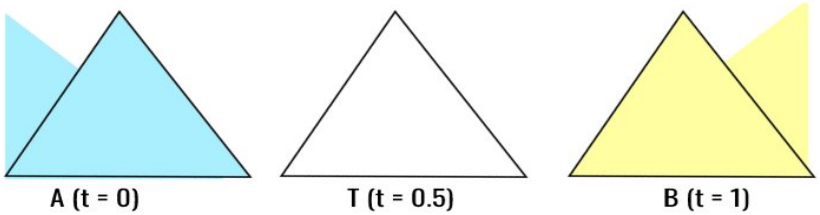
2D Affine Transformation with homogeneous coordinates

Summary of morphing procedure

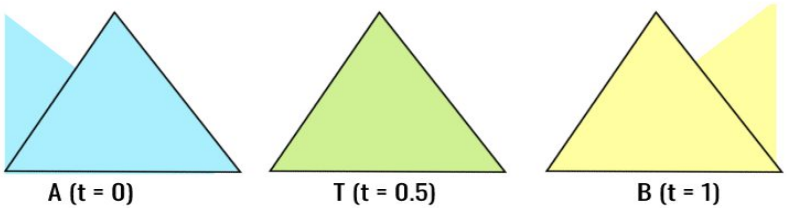
1. Linear Interpolation 을 통해 중간 Frame 의 삼각형 모양 계산 (T)



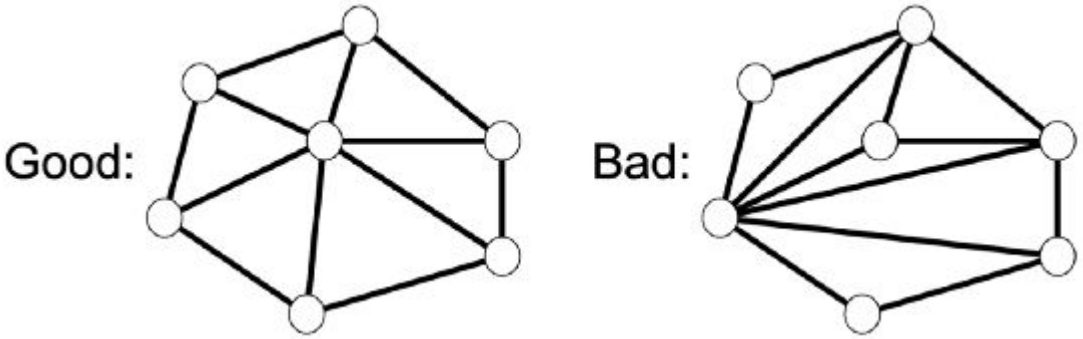
2. 각 이미지의 삼각형을 (A, B) Affine Transformation
   → 중간 삼각형 (T) 모양으로 Warp



3. Warped 된 두 삼각형을 Cross Dissolve 하여 중간 삼각형 (T) 의 값을 채운다.

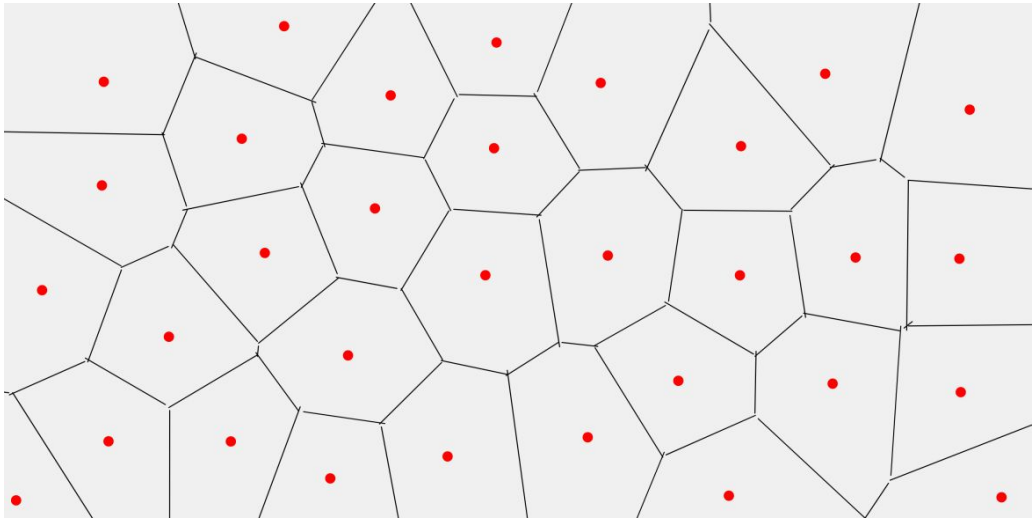


Delaunay Triangulation



최소각을 최대로 만드는 삼각 분할 기법 : **Delaunay Triangulation**

Voronoi Diagram

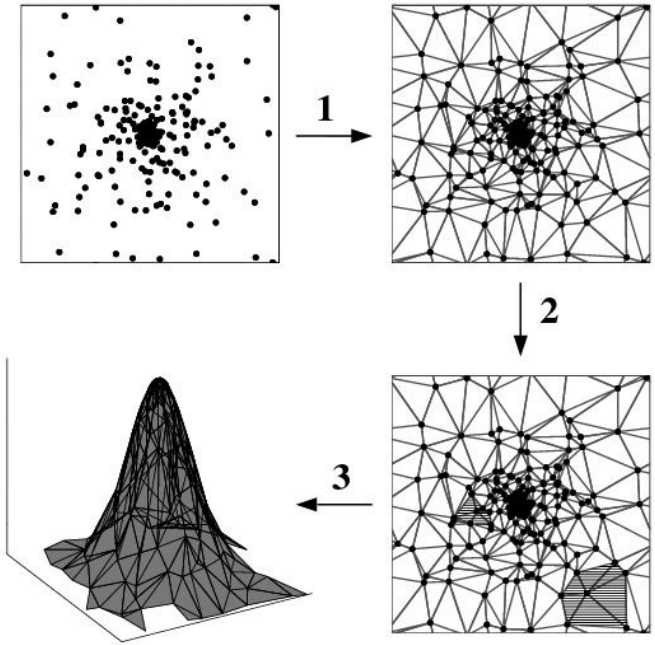


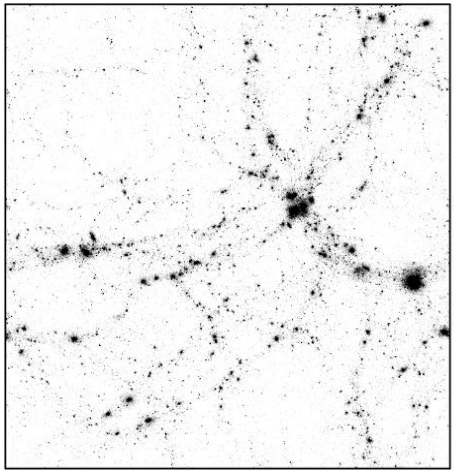
특정 점과의 거리가 가장 가까운 점들의 집합으로의 평면 분할기법 : **Voronoi Diagram**

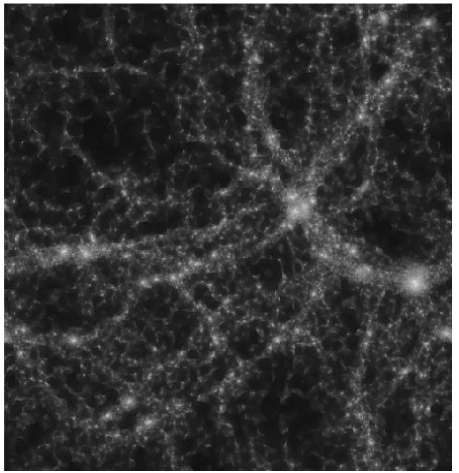
Voronoi Diagram



Use-case of Delaunay Triangulation

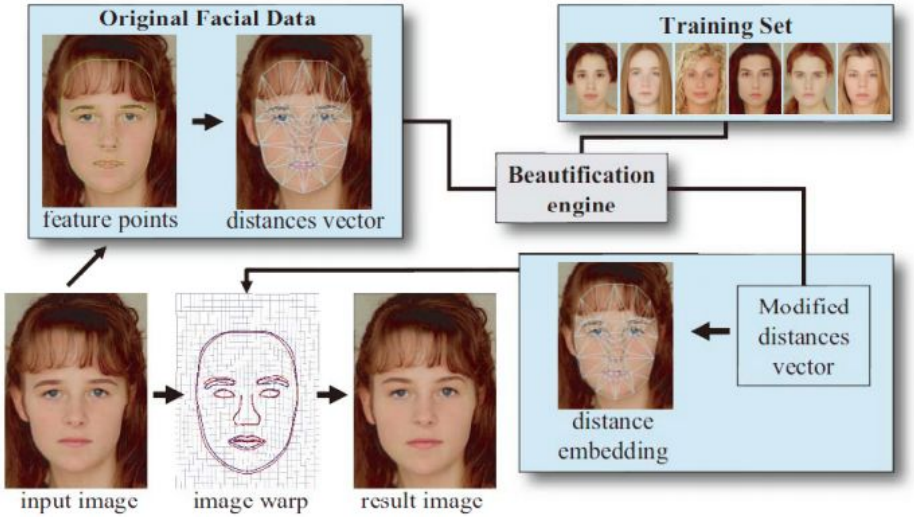






Delaunay Tessellation Field Estimation (DTFE)

Data-Driven Face Beautification (Siggraph 08)





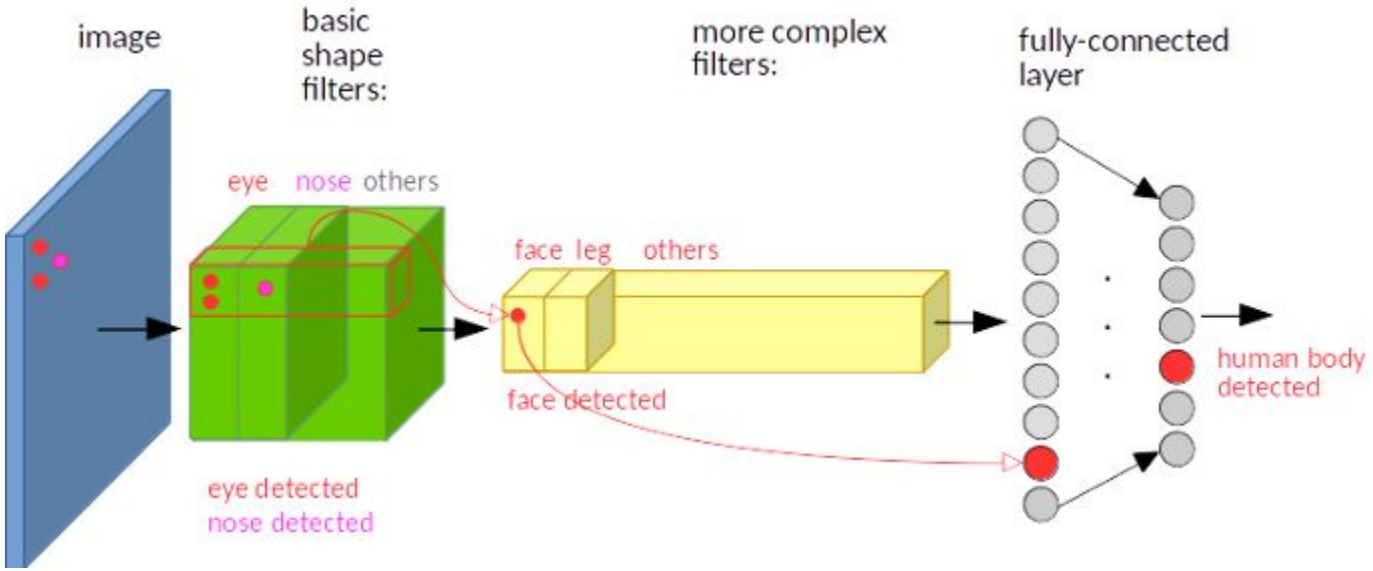
Bringing Portraits to life (Siggraph Asia 17)



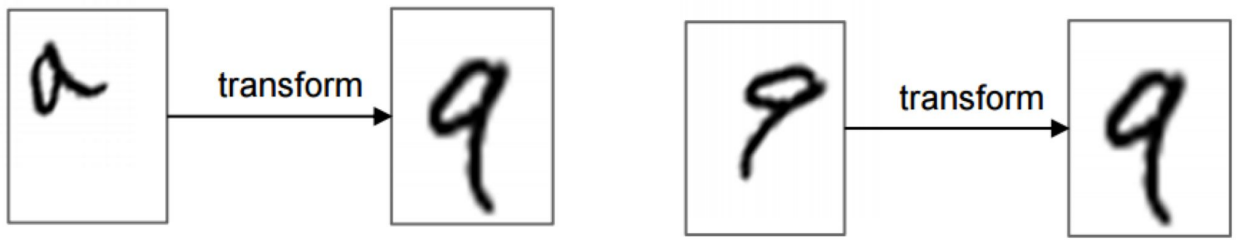
Image warping with Neural Networks

1. **Global Warping**
    - Spatial Transformer Networks (NeurIPS 15)
2. **Local Warping**
    - X2Face (ECCV 18)
    - Animating Arbitrary Objects via Deep Motion Transfer (CVPR 18 Oral)
    - First Order Motion Model (NeurIPS 19)
    - Motion Representation for Articulated Animations (CVPR 21)
    - One-shot Free-View Neural Talking-Head Synthesis (CVPR 21 Oral)

Spatial Transformer Networks (NeurIPS 15)

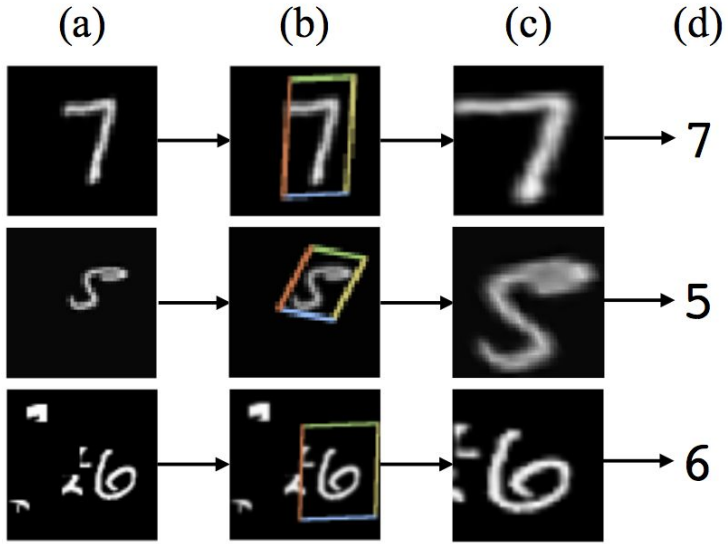


Spatial Transformer Networks (NeurIPS 15)



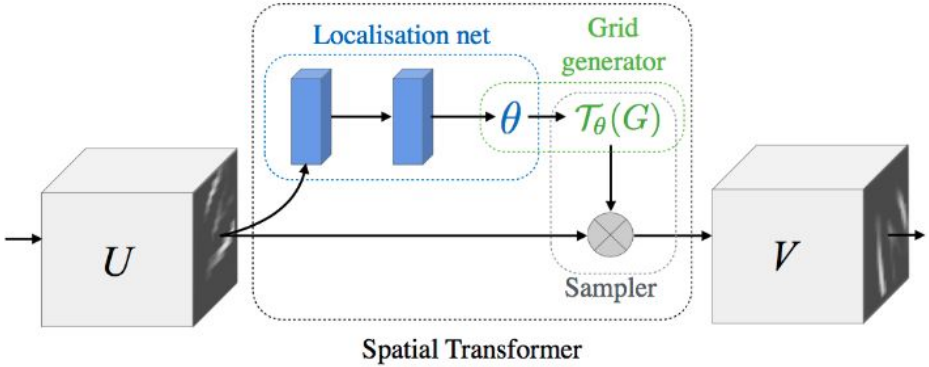
Spatial Transformer

Spatial Transformer Networks (NeurIPS 15)

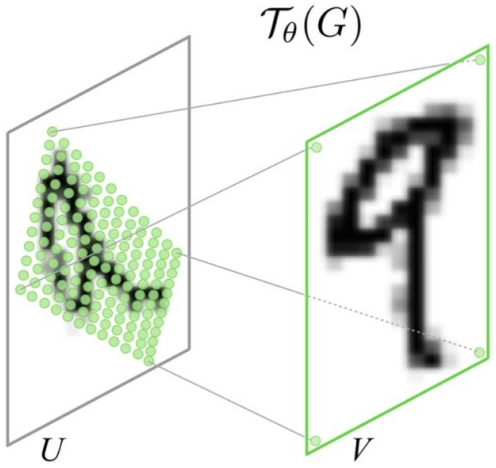


Visualization of Spatial Transformations

Spatial Transformer Networks (NeurIPS 15)

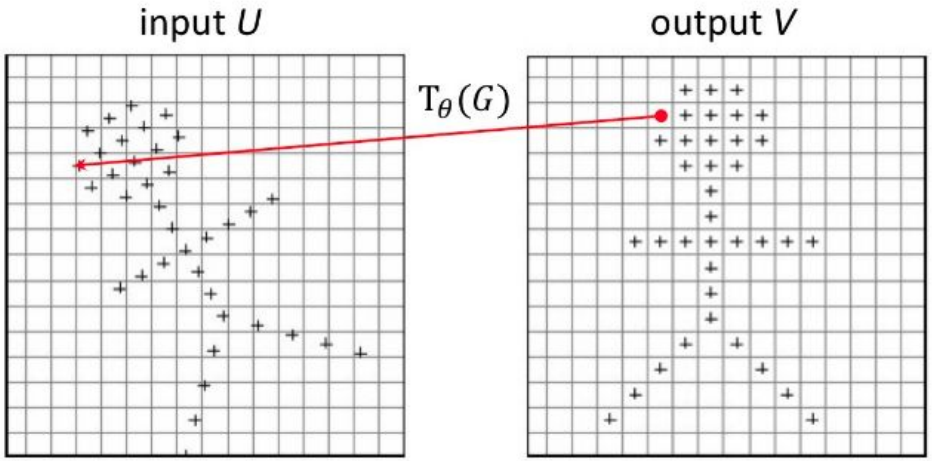


Spatial Transformer



$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \mathtt{A}_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$

Spatial Transformer Networks (NeurIPS 15)



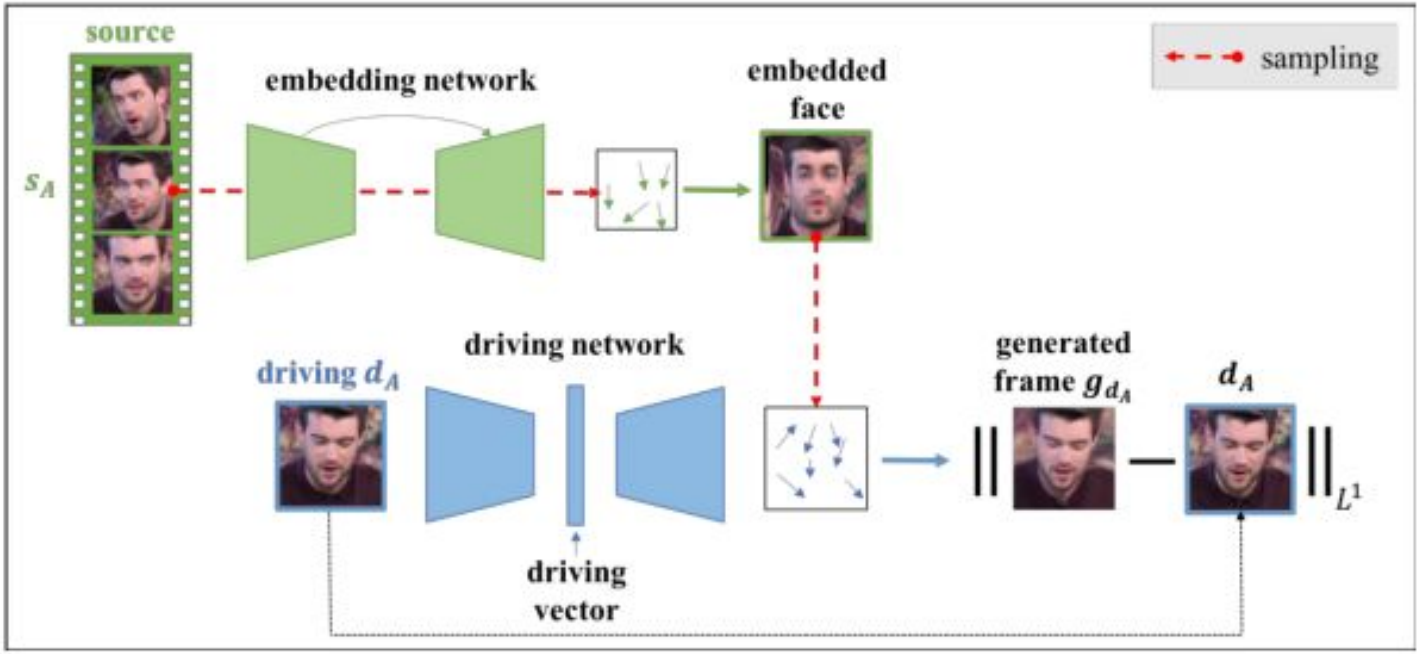
$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c k(x_i^s - m; \Phi_x) k(y_i^s - n; \Phi_y) \quad \forall i \in [1 \dots H'W'] \, \forall c \in [1 \dots C]$$

Interpolation

X2Face : A network for controlling face generation (ECCV 18)



X2Face : A network for controlling face generation (ECCV 18)

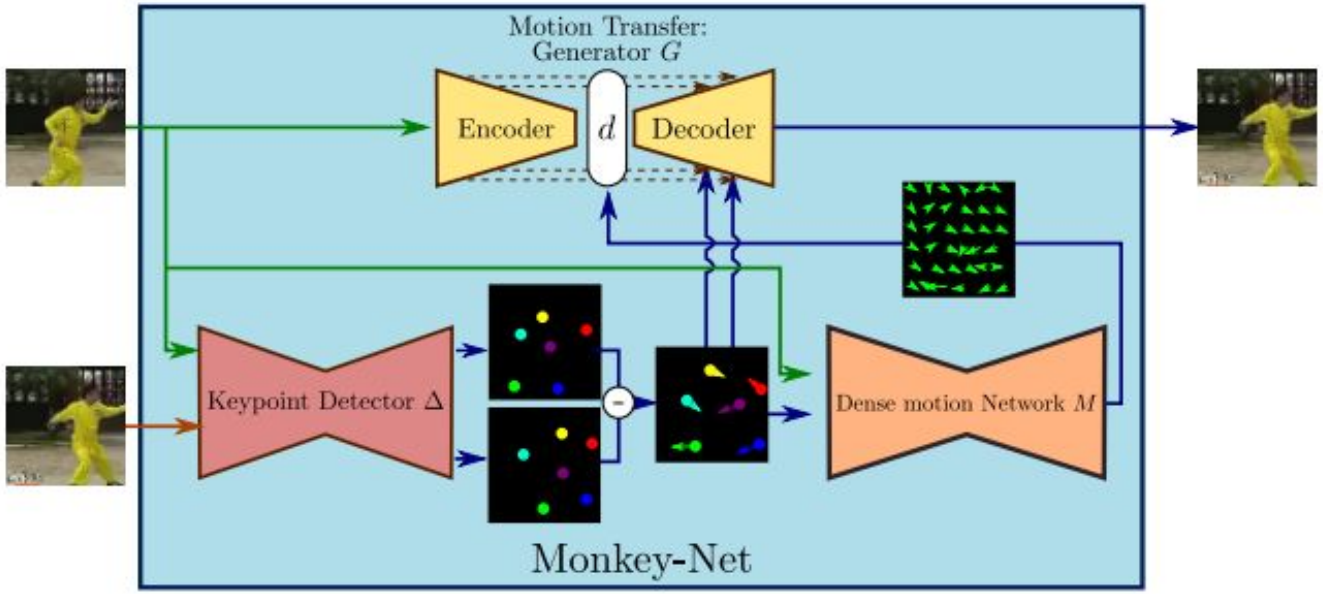


Animating Arbitrary Objects via Deep Motion Transfer (CVPR 18 Oral)





Animating Arbitrary Objects via Deep Motion Transfer (CVPR 18 Oral)

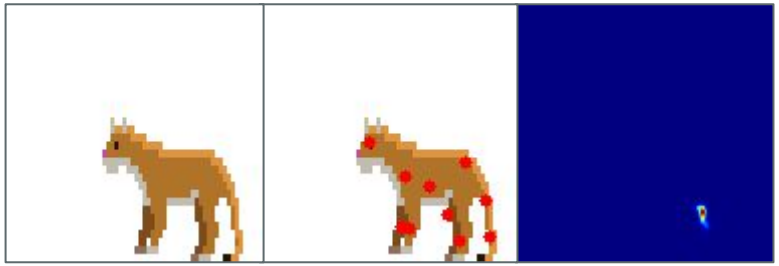


(b) Monkey-Net Architecture for Self-Learned Animation

Animating Arbitrary Objects via Deep Motion Transfer (CVPR 18 Oral)

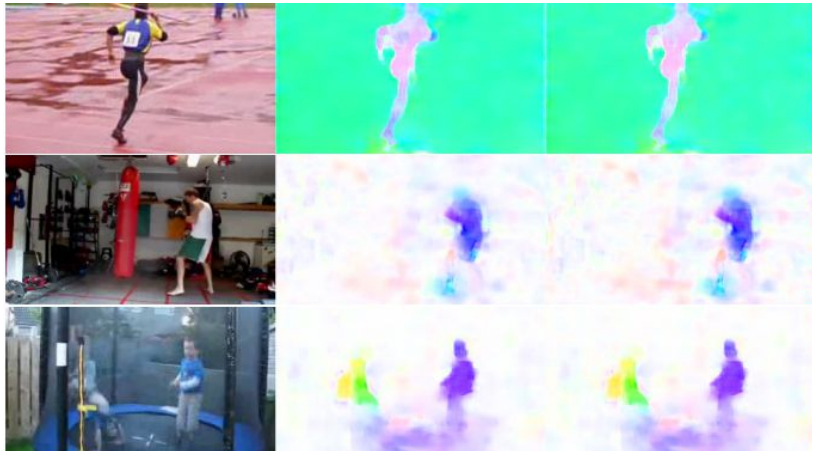
$$\boldsymbol{h}_k = \sum_{p \in \mathcal{U}} H_k[p]p; \;\; \Sigma_k = \sum_{p \in \mathcal{U}} H_k[p](p - \boldsymbol{h}_k)(p - \boldsymbol{h}_k)^\top$$

Unsupervised Keypoint Detection (Heatmap)





Animating Arbitrary Objects via Deep Motion Transfer (CVPR 18 Oral)



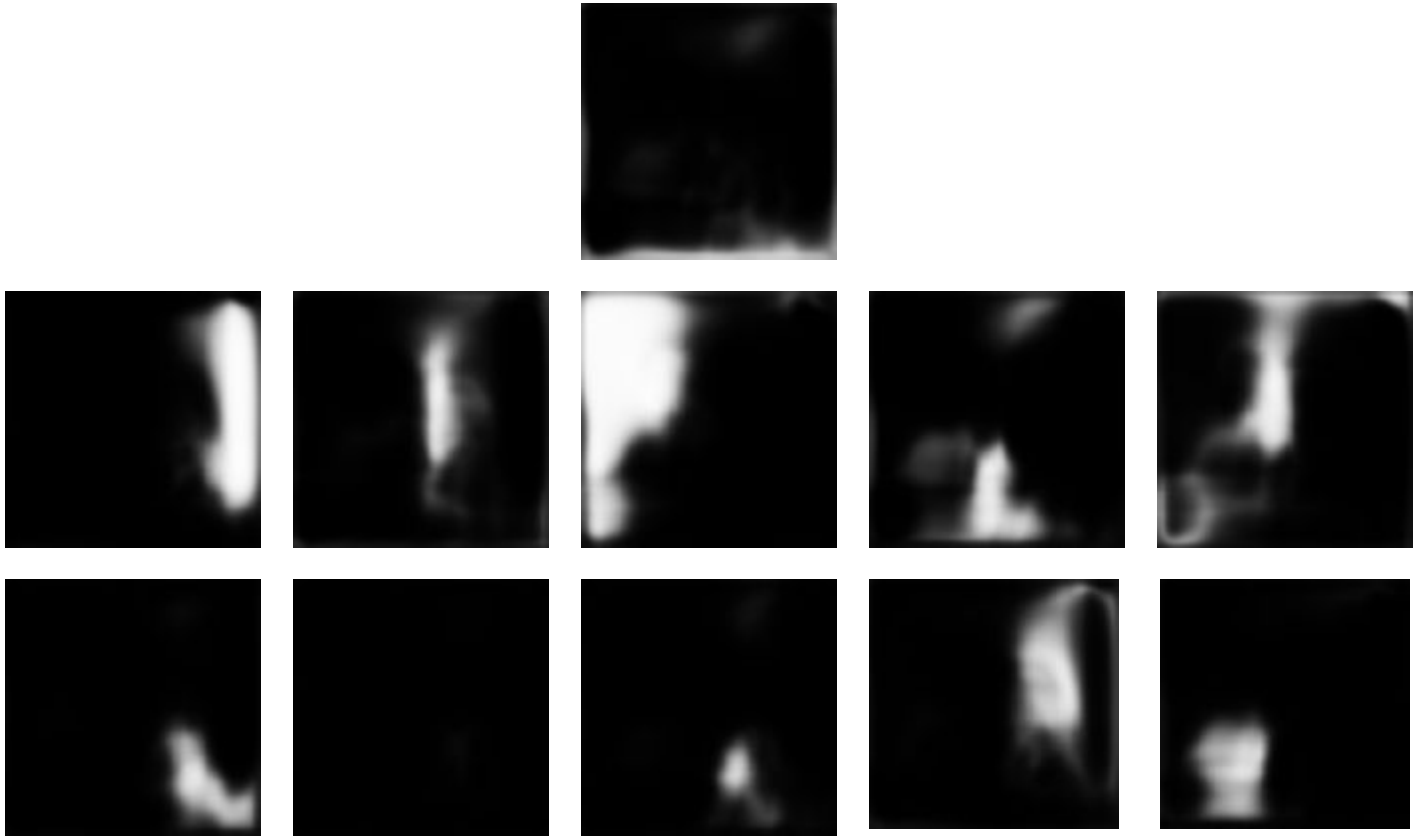
Part-based Formulation

$$\mathcal{F}_{\text{coarse}} = \sum_{k=1}^{K+1} M_k \otimes \rho(h_k)$$

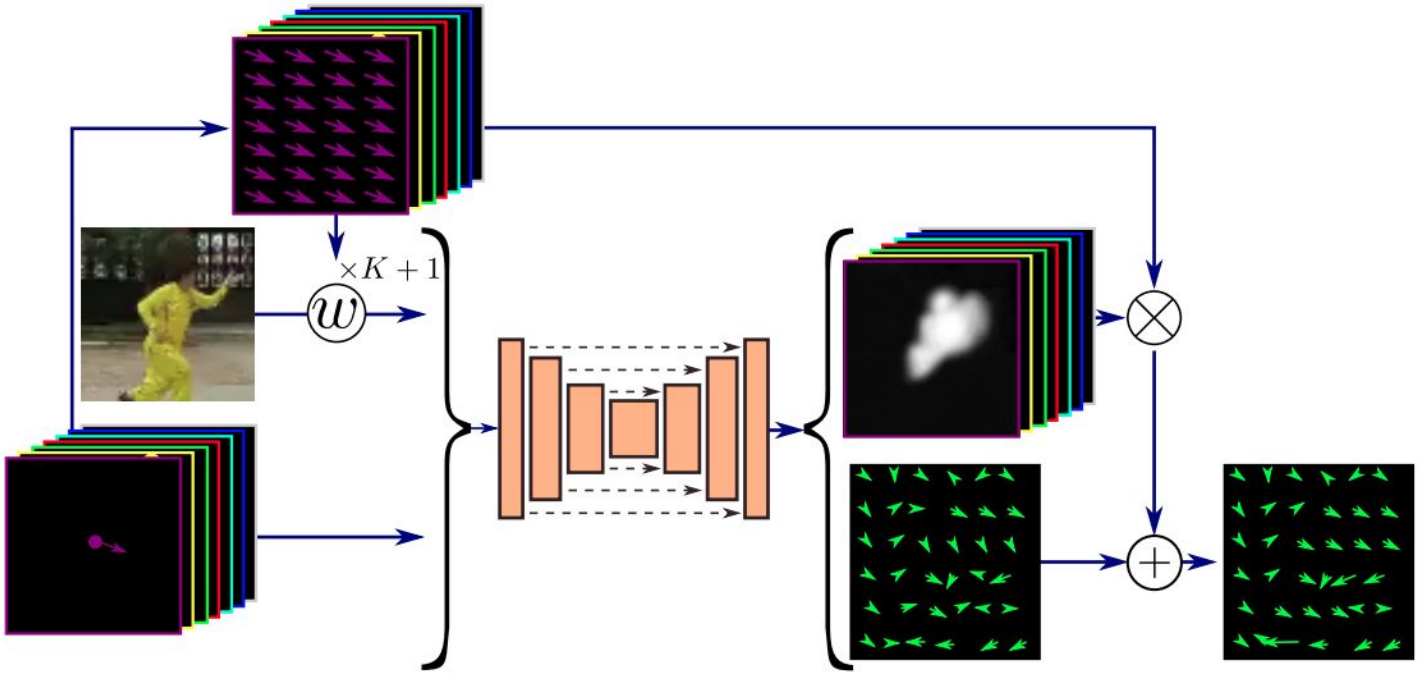
$$\mathcal{F} = \mathcal{F}_{\text{coarse}} + \mathcal{F}_{\text{residual}}$$

Estimating masks

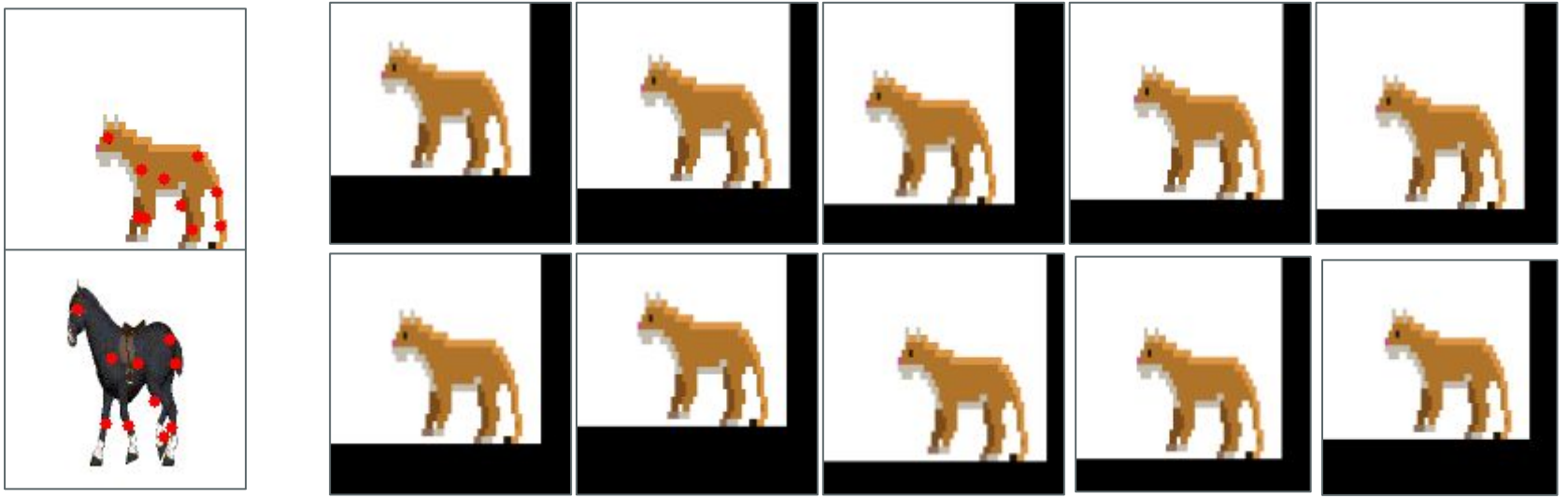
Animating Arbitrary Objects via Deep Motion Transfer (CVPR 18 Oral)



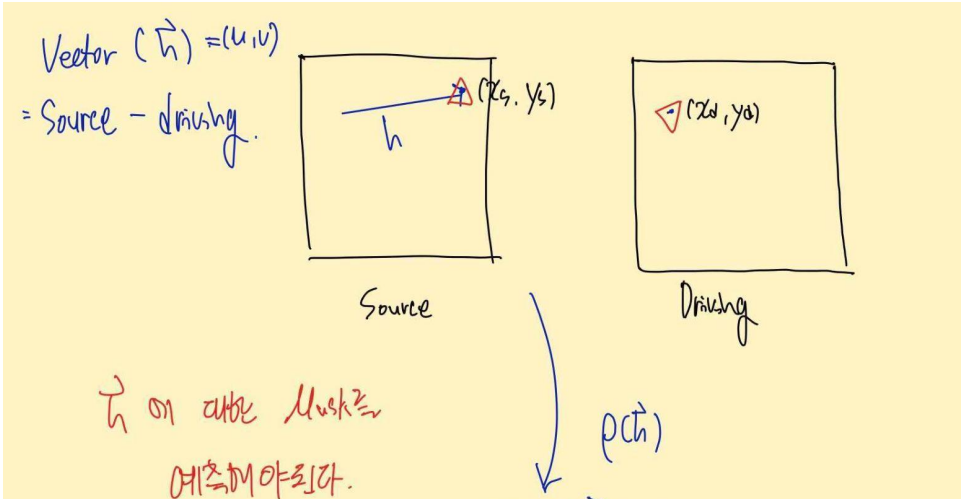
Animating Arbitrary Objects via Deep Motion Transfer (CVPR 18 Oral)

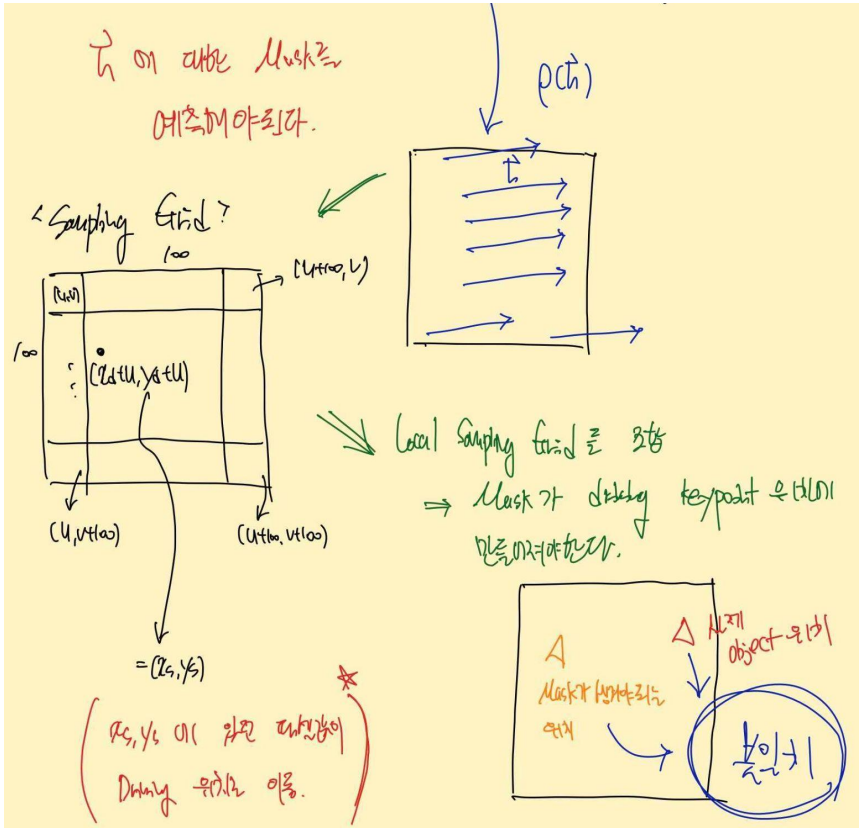


Animating Arbitrary Objects via Deep Motion Transfer (CVPR 18 Oral)

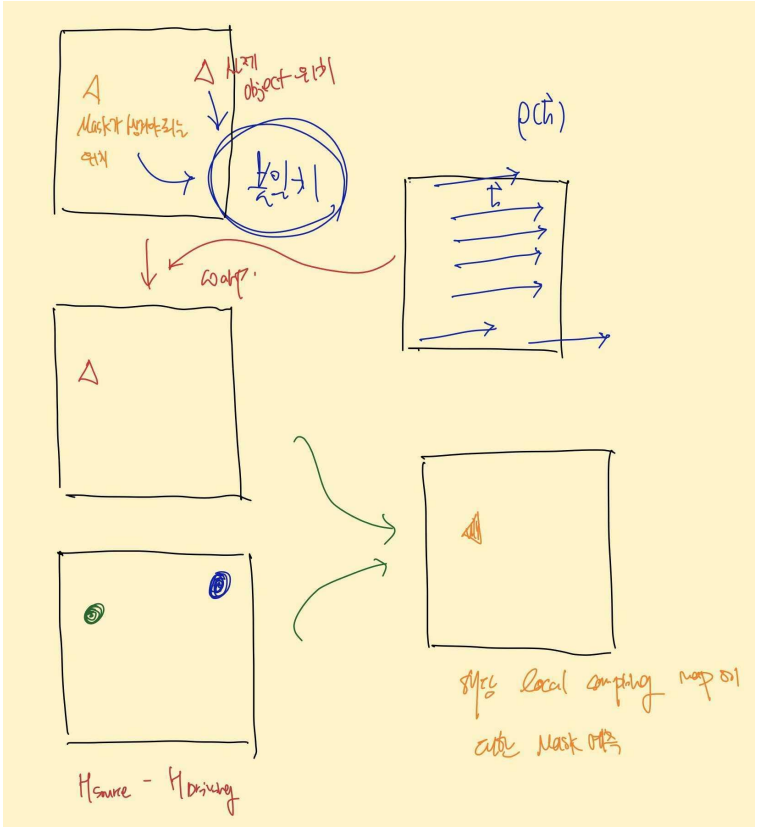


Animating Arbitrary Objects via Deep Motion Transfer (CVPR 18 Oral)





Animating Arbitrary Objects via Deep Motion Transfer (CVPR 18 Oral)



Animating Arbitrary Objects via Deep Motion Transfer (CVPR 18 Oral)



$$\xi'_r = f_w(\xi_r, \mathcal{F})$$

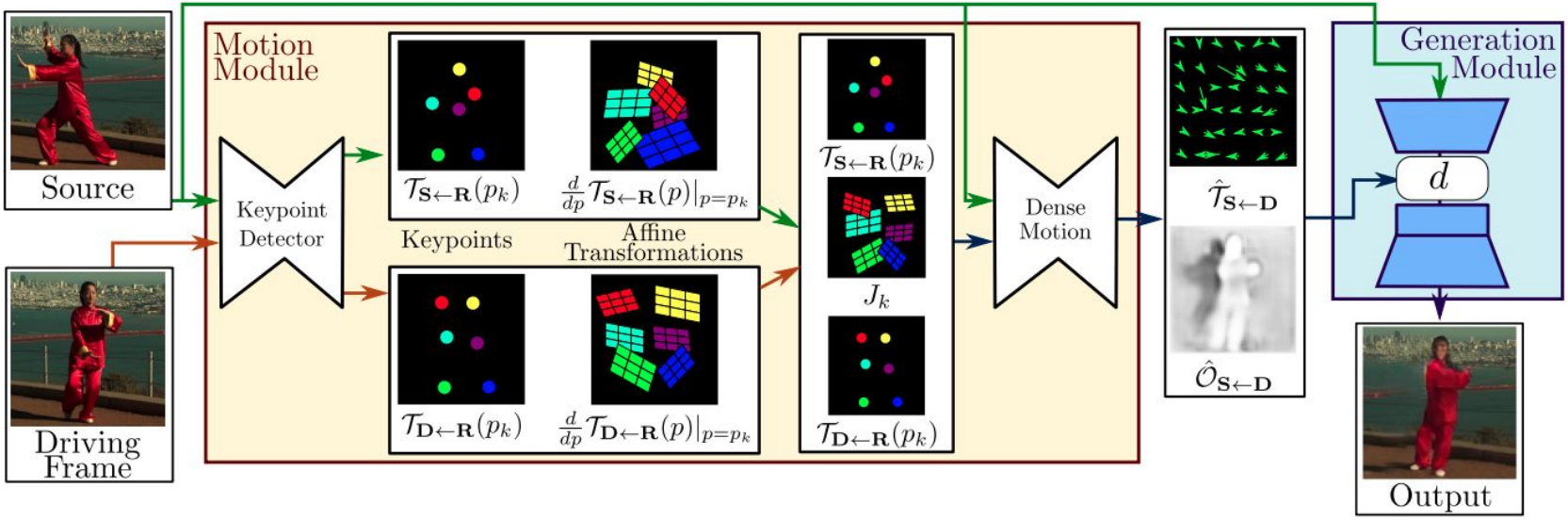
Warping feature maps with Optical Flow F

First Order Motion Model (NeurIPS 19)

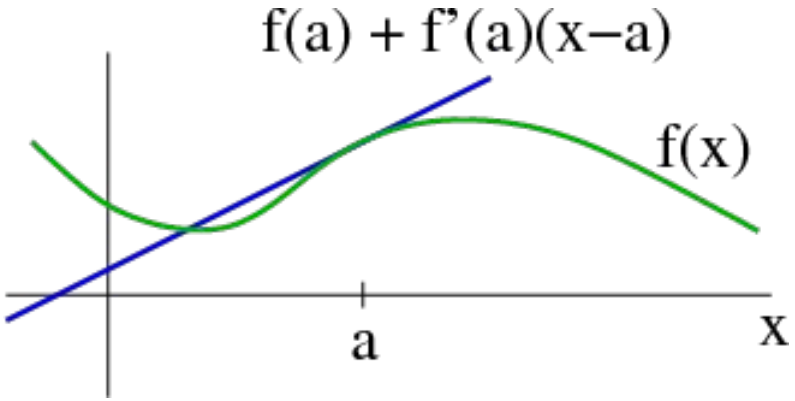




First Order Motion Model (NeurIPS 19)



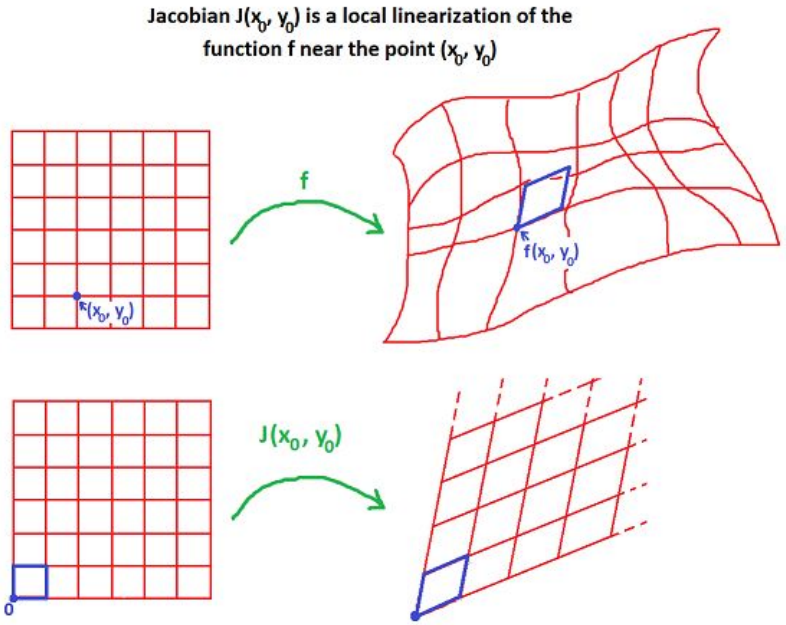
First Order Motion Model (NeurIPS 19)



Taylor Expansion

First Order Motion Model (NeurIPS 19)

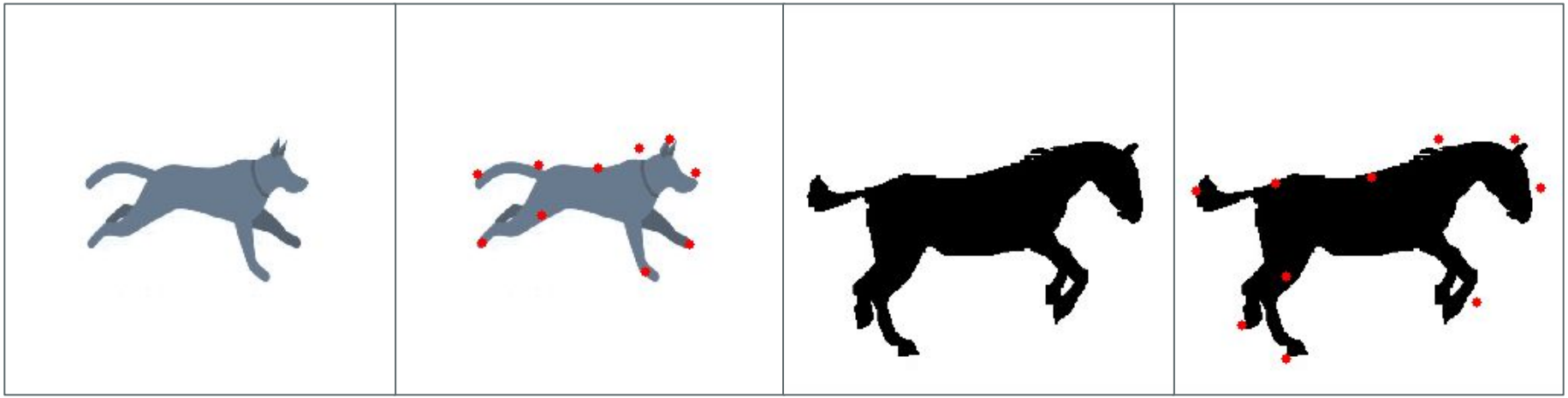




First Order Motion Model (NeurIPS 19)

$$\mathcal{T}_{\mathbf{X}\leftarrow\mathbf{R}}(p) = \mathcal{T}_{\mathbf{X}\leftarrow\mathbf{R}}(p_k) + \left(\frac{d}{dp}\mathcal{T}_{\mathbf{X}\leftarrow\mathbf{R}}(p)\Big|_{p=p_k}\right)(p - p_k)$$

Key-point 에서 선형 변환으로 근사



First Order Motion Model (NeurIPS 19)

$$\mathcal{T}_{\mathbf{X}\leftarrow\mathbf{R}}(p) \simeq \left\{ \left\{ \mathcal{T}_{\mathbf{X}\leftarrow\mathbf{R}}(p_1), \frac{d}{dp}\mathcal{T}_{\mathbf{X}\leftarrow\mathbf{R}}(p)\Big|_{p=p_1} \right\}, \ldots \left\{ \mathcal{T}_{\mathbf{X}\leftarrow\mathbf{R}}(p_k), \frac{d}{dp}\mathcal{T}_{\mathbf{X}\leftarrow\mathbf{R}}(p)\Big|_{p=p_K} \right\} \right\}.$$

K개의 Affine 변환

First Order Motion Model (NeurIPS 19)

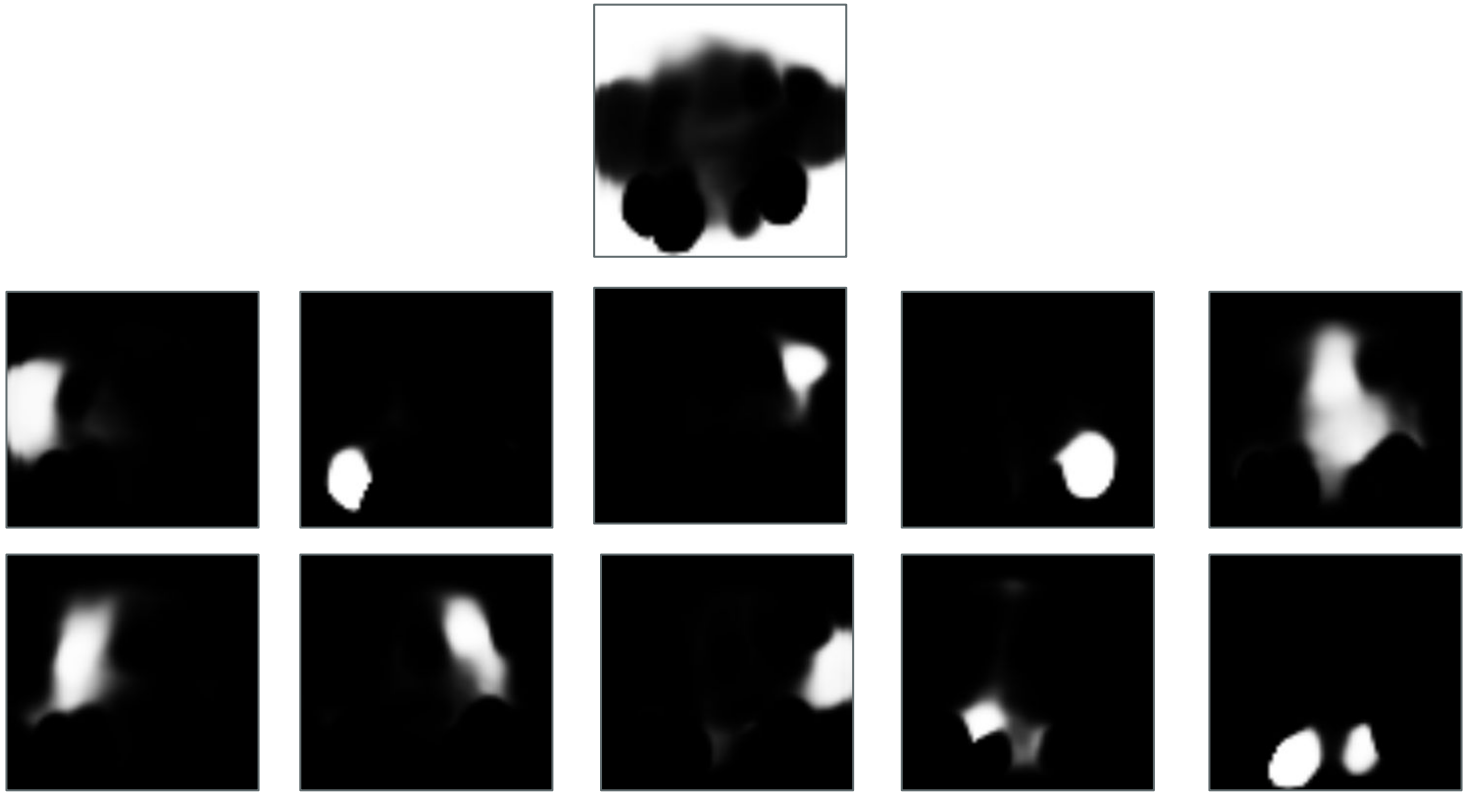
$$\mathcal{T}_{\mathbf{S}\leftarrow\mathbf{D}} = \mathcal{T}_{\mathbf{S}\leftarrow\mathbf{R}} \circ \mathcal{T}_{\mathbf{R}\leftarrow\mathbf{D}} = \mathcal{T}_{\mathbf{S}\leftarrow\mathbf{R}} \circ \mathcal{T}_{\mathbf{D}\leftarrow\mathbf{R}}^{-1},$$

Reference Frame 을 매개로 Driving 에서 Source 로의 Local Affine 변환

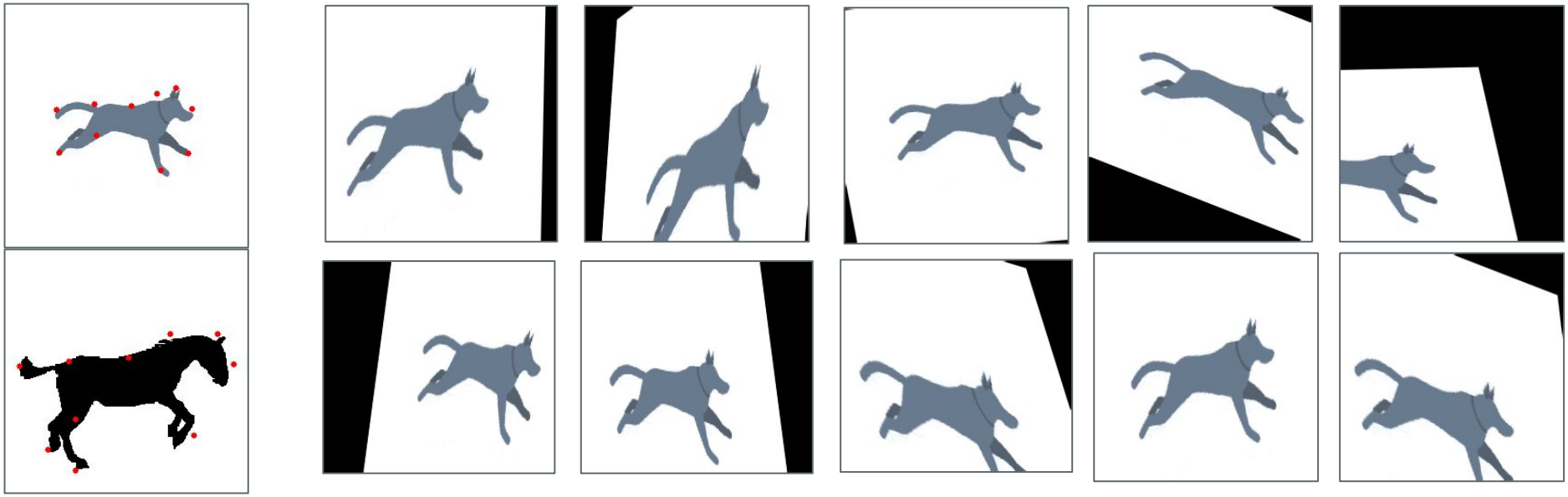
$$\hat{\mathcal{T}}_{\mathbf{S}\leftarrow\mathbf{D}}(z) = \mathbf{M}_0 z + \sum_{k=1}^{K} \mathbf{M}_k \left(\mathcal{T}_{\mathbf{S}\leftarrow\mathbf{R}}(p_k) + J_k(z - \mathcal{T}_{\mathbf{D}\leftarrow\mathbf{R}}(p_k))\right)$$

Part-based Formulation

First Order Motion Model (NeurIPS 19)



First Order Motion Model (NeurIPS 19)



First Order Motion Model (NeurIPS 19)



Occlusion mask



Dense Warped

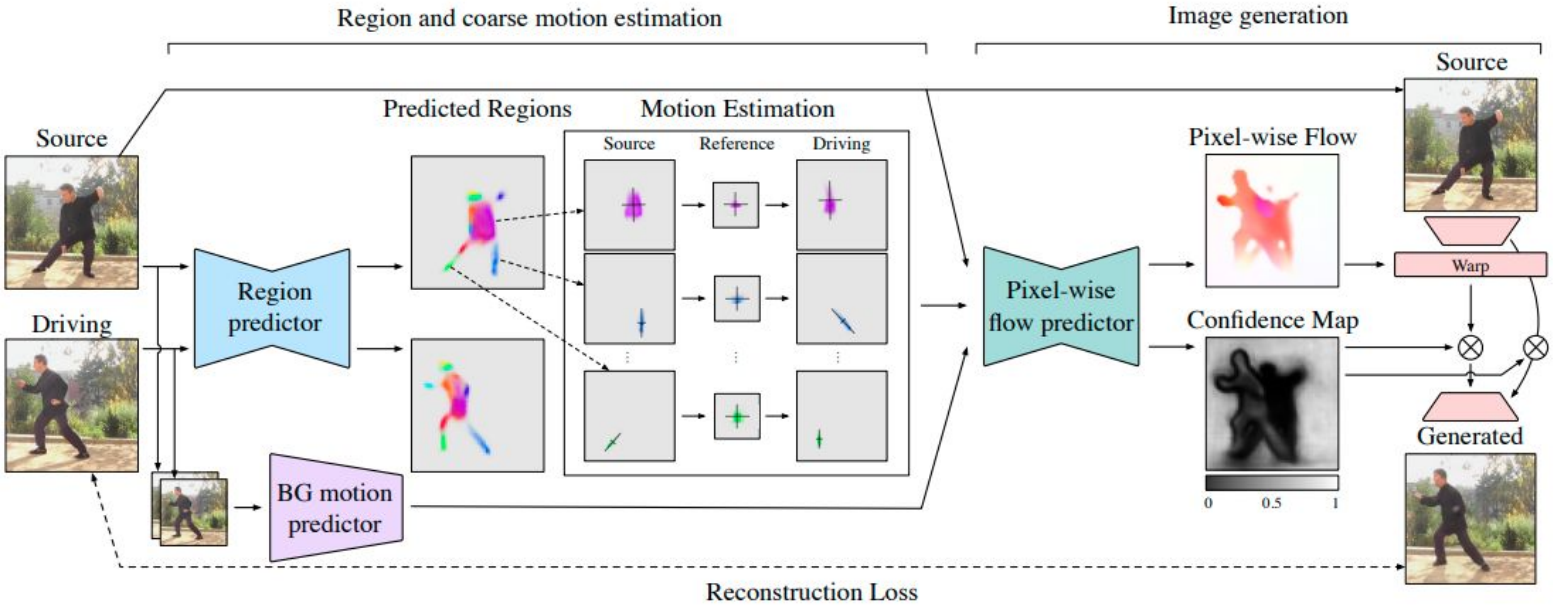


Generated

Motion Representation for Articulated Animations (CVPR 21)

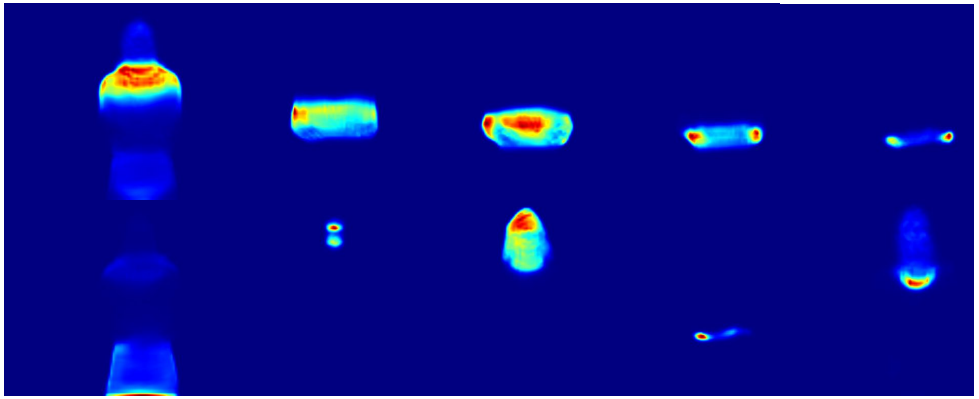


Motion Representation for Articulated Animations (CVPR 21)

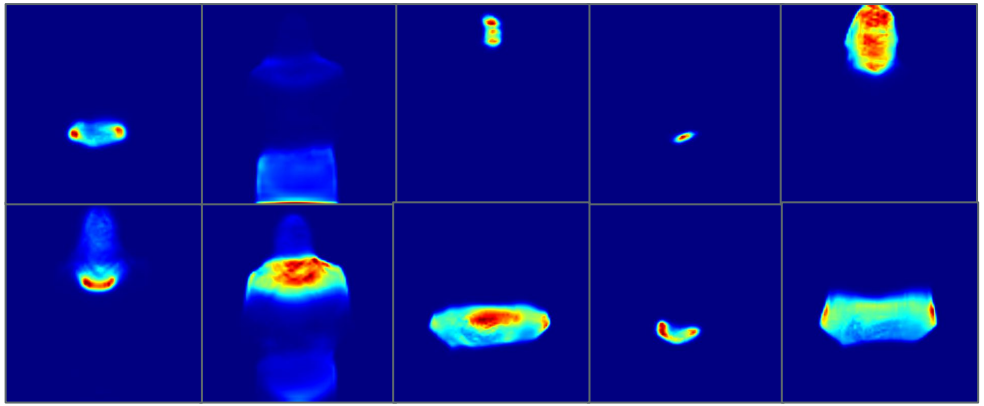


Motion Representation for Articulated Animations (CVPR 21)









Motion Representation for Articulated Animations (CVPR 21)

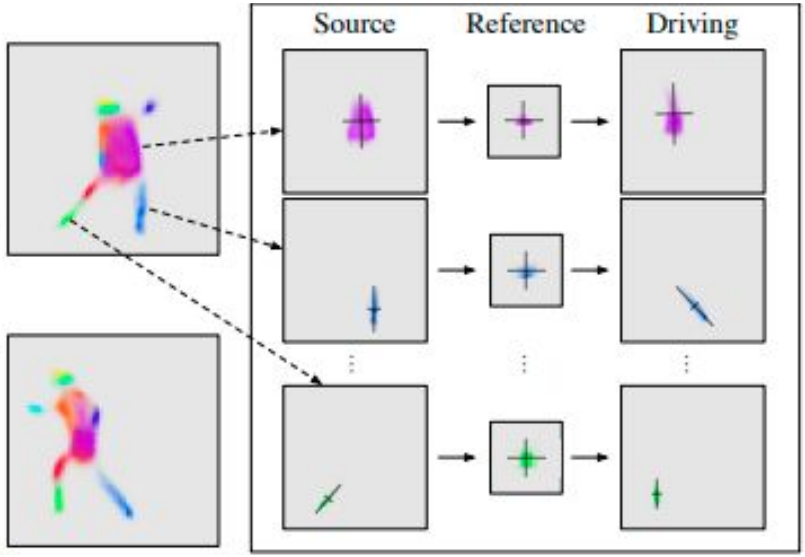
$$\mu^k = \sum_{z \in \mathcal{Z}} \mathbf{M}^k(z) z,$$

$$U^k S^k V^k = \sum_{z \in \mathcal{Z}} \mathbf{M}^k(z) \left(z - \mu^k\right) \left(z - \mu^k\right)^{\mathsf{T}} \text{ (SVD)},$$

$$A^k_{\mathbf{X} \leftarrow \mathbf{R}} = \left[U^k S^{k\,\frac{1}{2}}, \mu^k\right].$$

Affine Transformation by PCA

Motion Representation for Articulated Animations (CVPR 21)



Affine Transformation

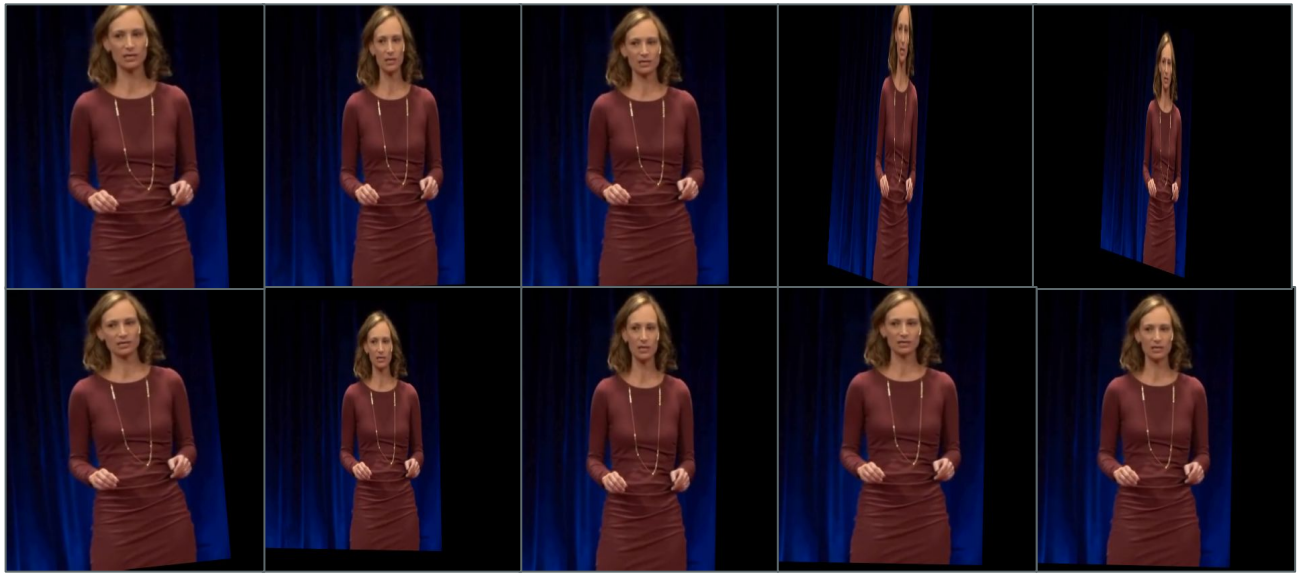
Motion Representation for Articulated Animations (CVPR 21)





Motion Representation for Articulated Animations (CVPR 21)





Motion Representation for Articulated Animations (CVPR 21)

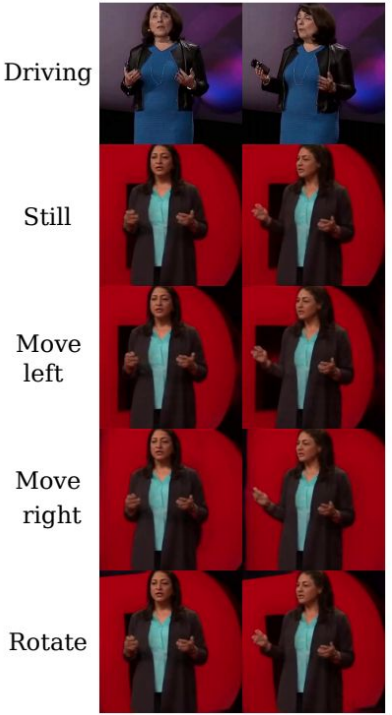




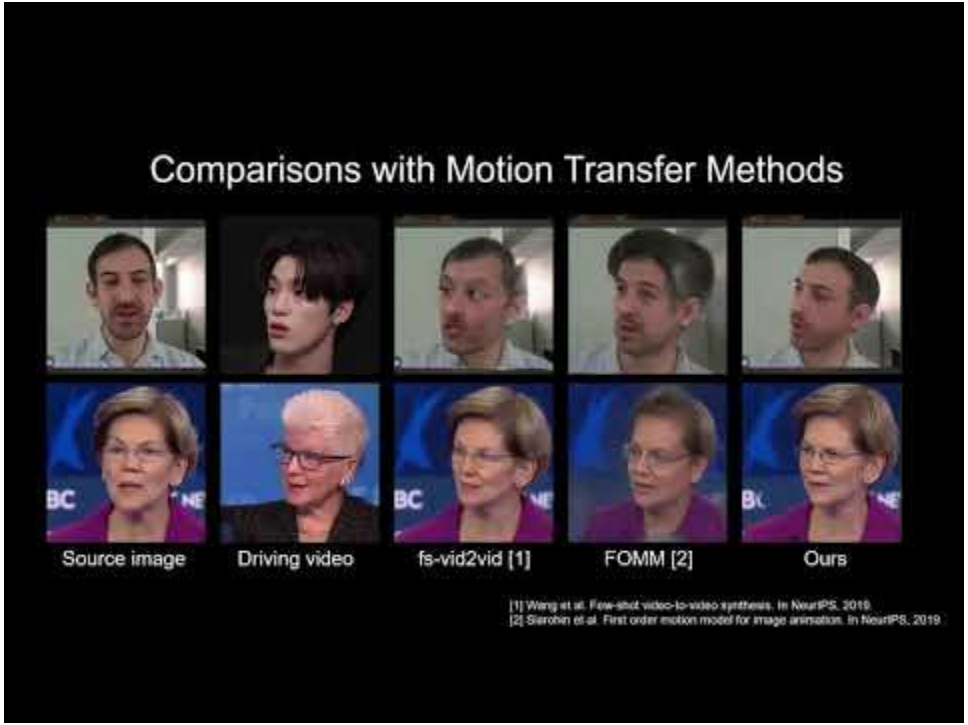
Motion Representation for Articulated Animations (CVPR 21)



(a) Comparisons of motion/part representations

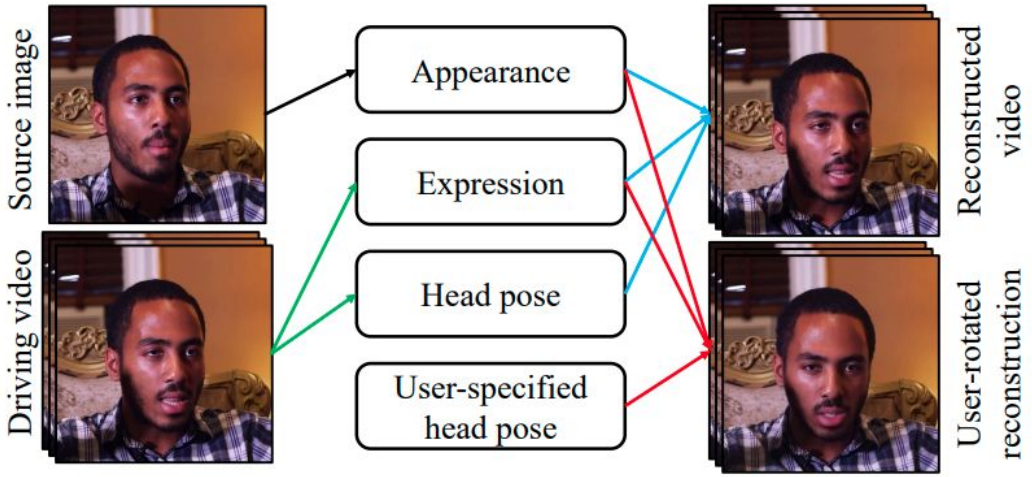


One-shot Free-View Neural Talking-Head Synthesis (CVPR 21 Oral)

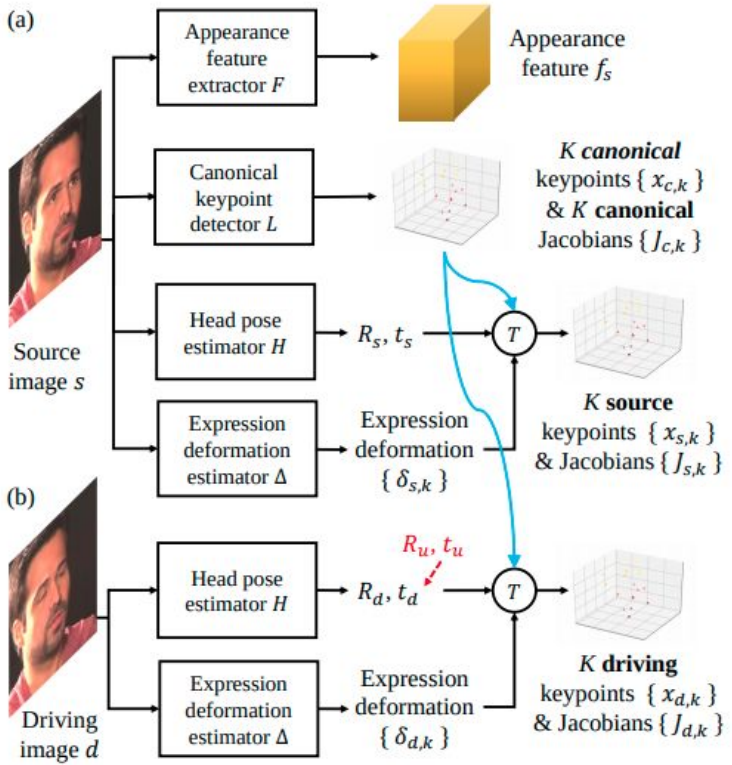


Demo

One-shot Free-View Neural Talking-Head Synthesis (CVPR 21 Oral)



One-shot Free-View Neural Talking-Head Synthesis (CVPR 21 Oral)

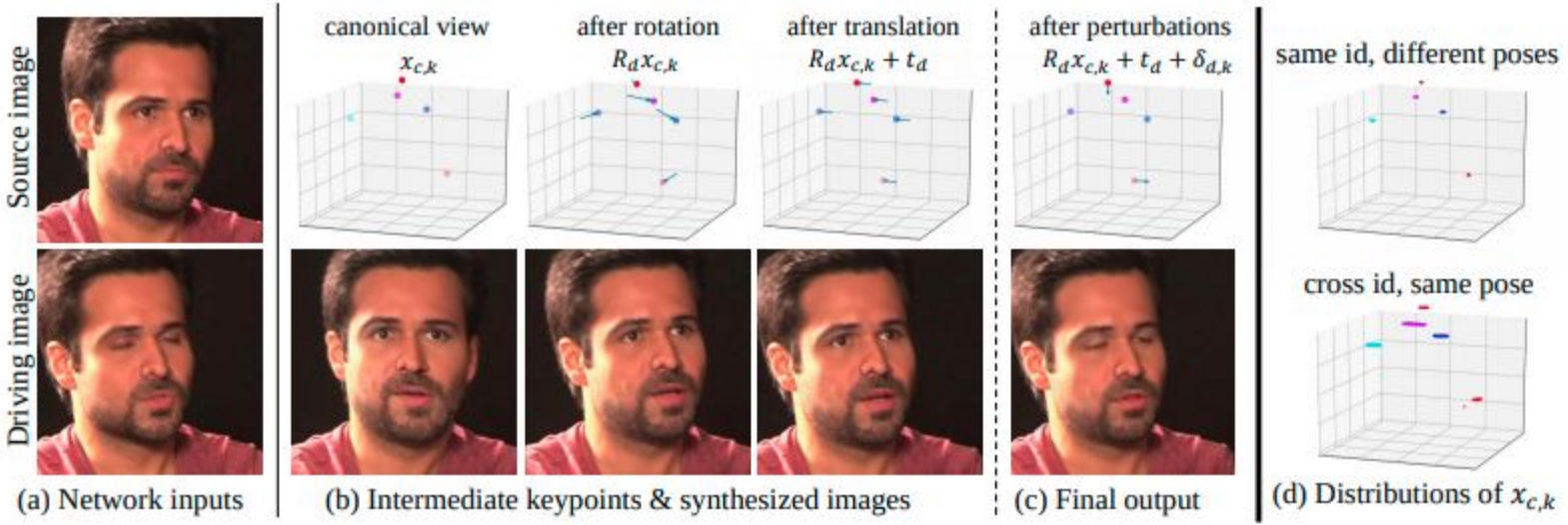


One-shot Free-View Neural Talking-Head Synthesis (CVPR 21 Oral)

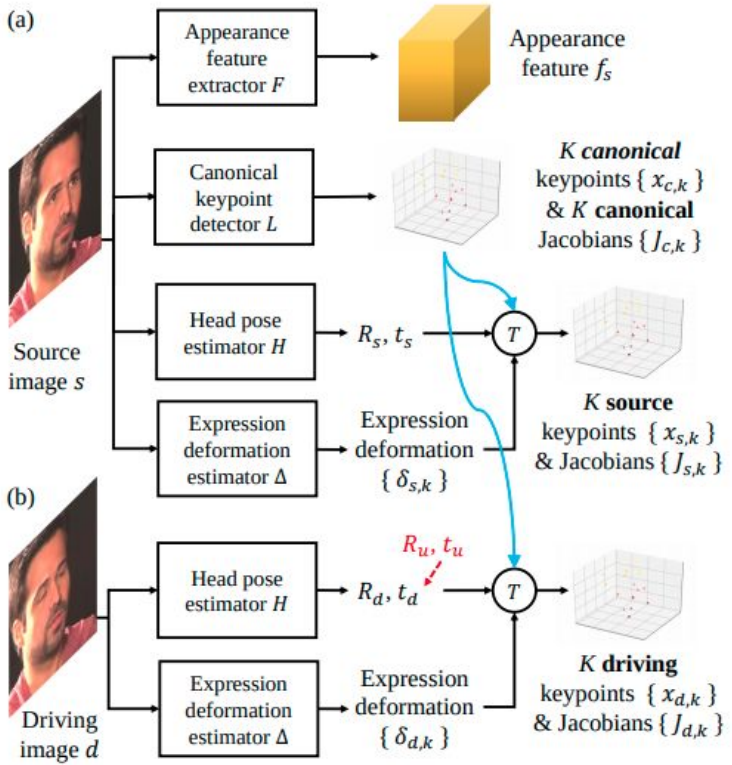
$$x_{s,k} = T_x(x_{c,k}, R_s, t_s, \delta_{s,k}) \equiv R_s x_{c,k} + t_s + \delta_{s,k}$$
$$J_{s,k} = T_J(J_{c,k}, R_s) \equiv R_s J_{s,k}.$$

Image specific Key-points and Jacobians (Source)

One-shot Free-View Neural Talking-Head Synthesis (CVPR 21 Oral)



One-shot Free-View Neural Talking-Head Synthesis (CVPR 21 Oral)

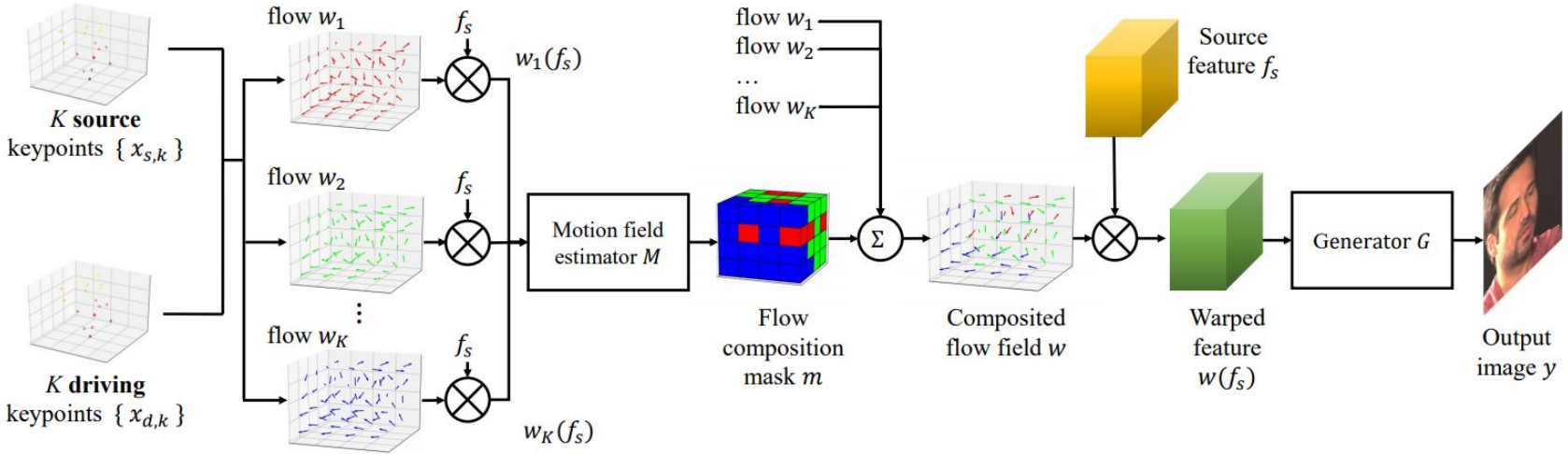


One-shot Free-View Neural Talking-Head Synthesis (CVPR 21 Oral)

$$x_{d,k} = T_x(x_{c,k}, R_d, t_d, \delta_{d,k}) = R_d x_{c,k} + t_d + \delta_{d,k}$$
$$J_{d,k} = T_J(J_{c,k}, R_d) = R_d J_{c,k}.$$

Image specific Key-points and Jacobians (Driving)

One-shot Free-View Neural Talking-Head Synthesis (CVPR 21 Oral)



Hairstyle Transfer with Image Warping

