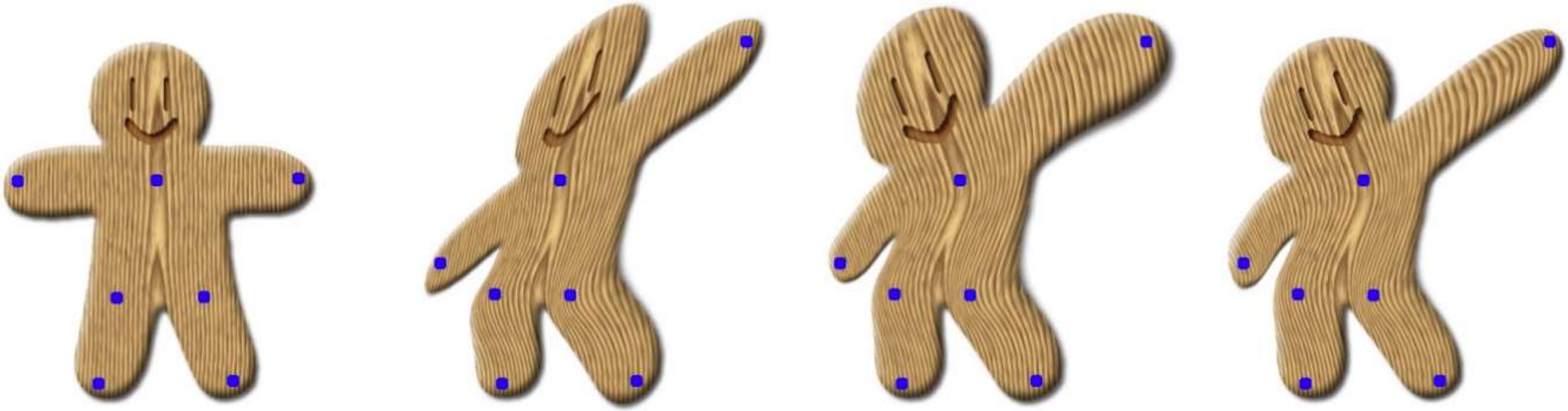
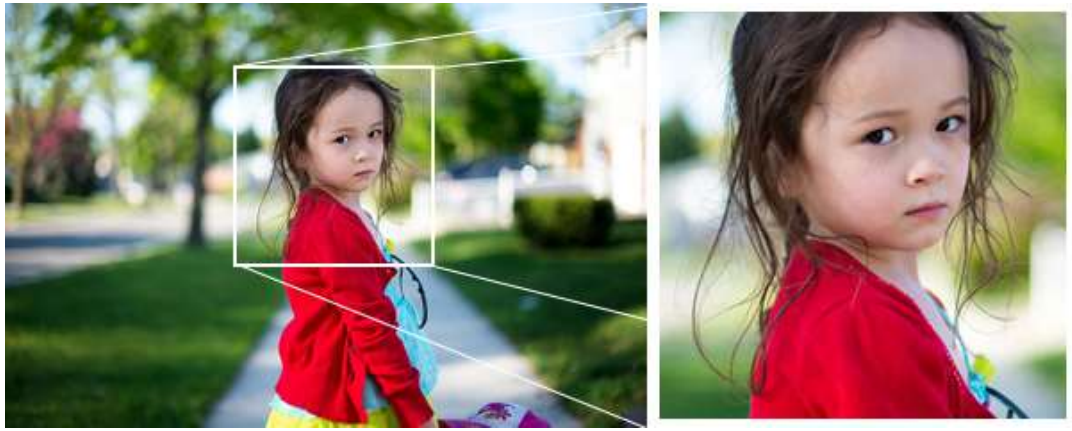
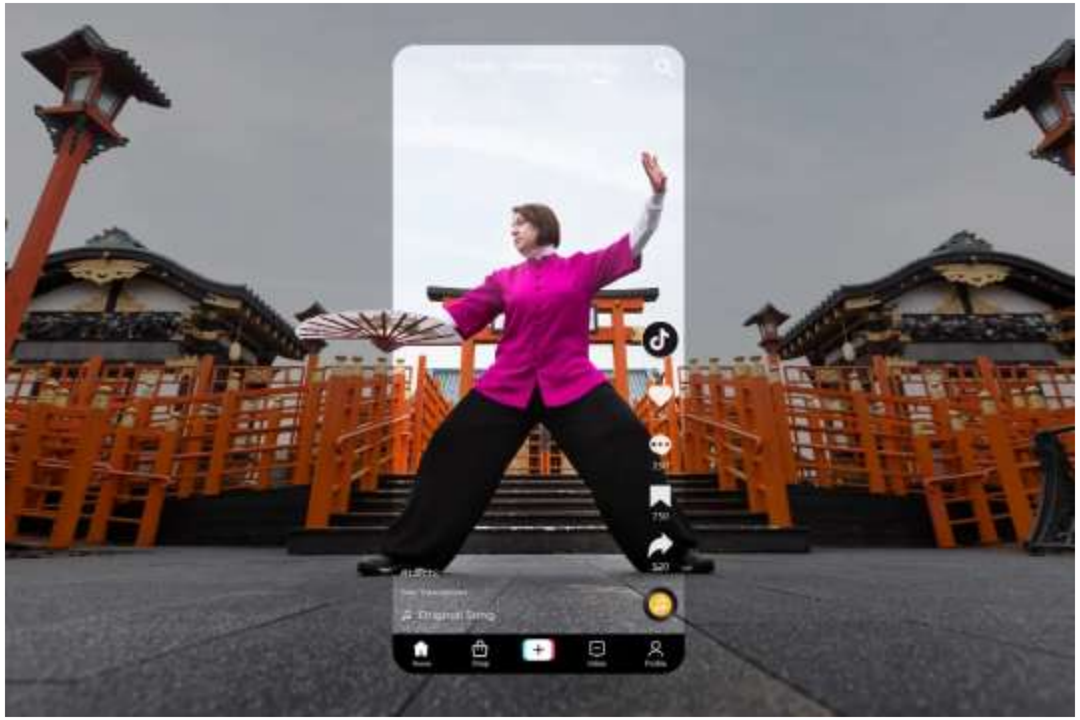
中国科学技术大学
University of Science and Technology of China

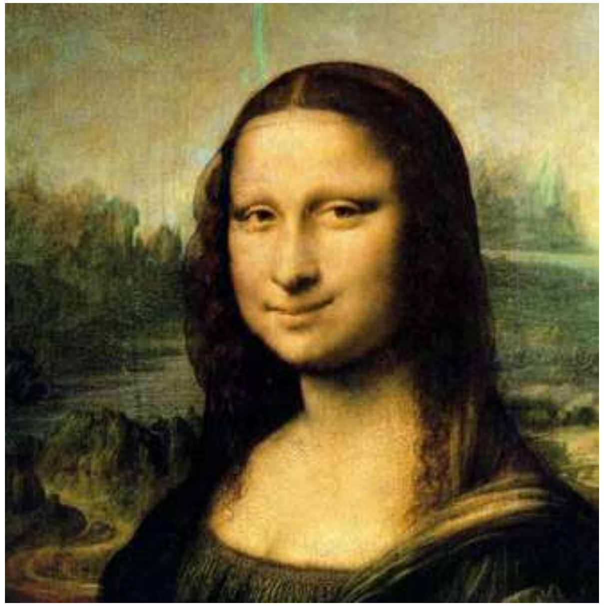
图像几何变换
Geometric Transformation of Images

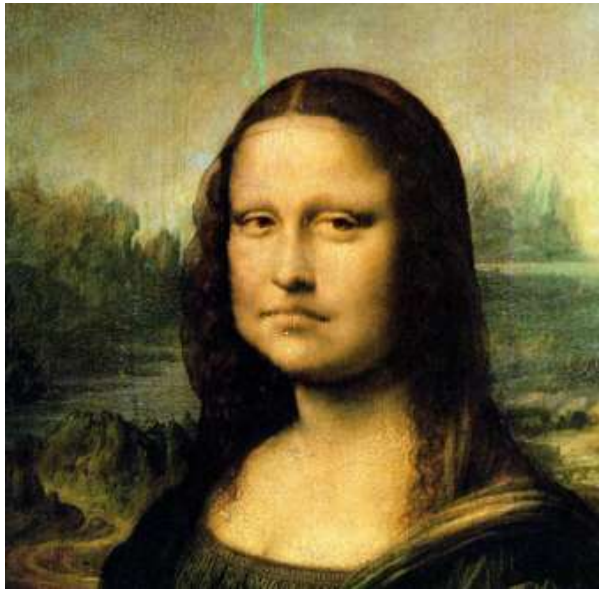


Recall



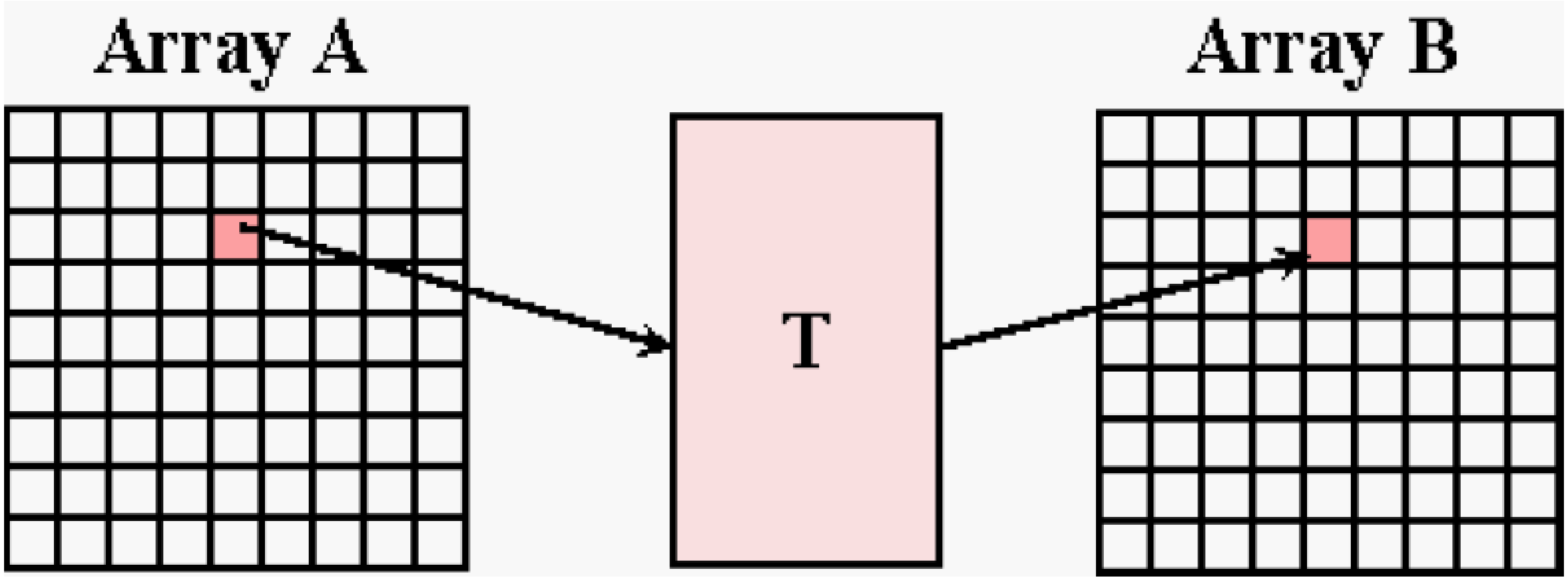




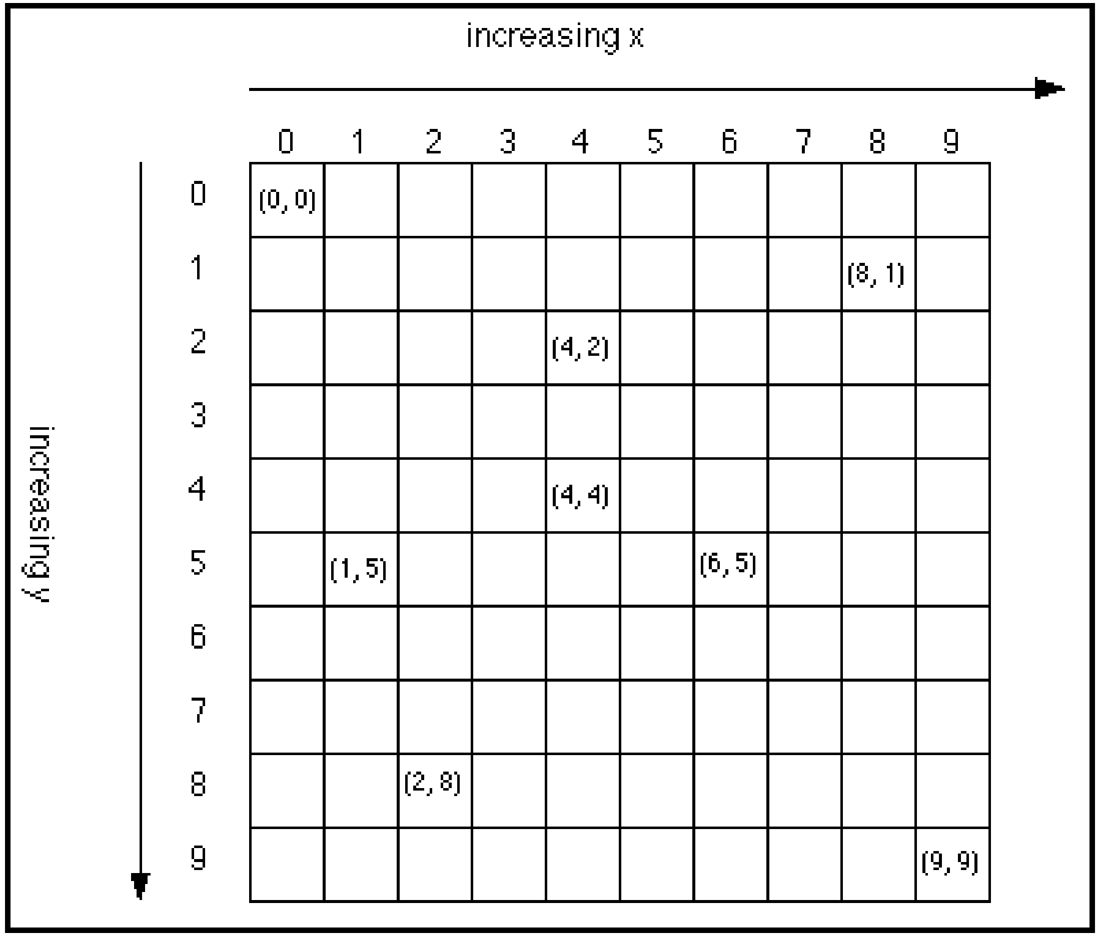


Recall

$$\mathbf{B}[x, y] = T\left[\mathbf{A}[x, y]\right]$$



(OpenCV / PyTorch) Image Coordinate

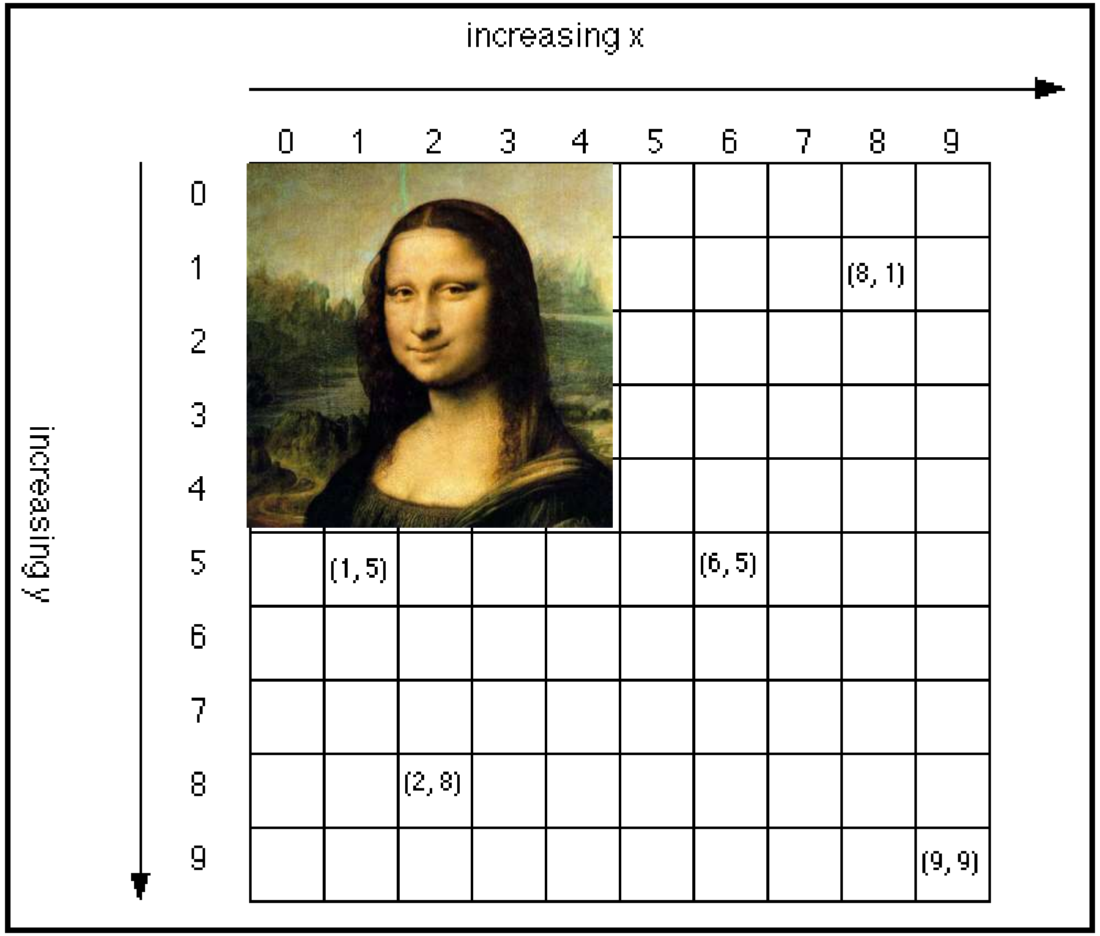



```python
import cv2
import numpy as np

def test_opencv_coordinate():
    img_size = 128
    img = np.zeros((img_size, img_size*2 , 3), dtype=np.uint8)
    for y in range(img_size):
        img[y, :, 0] = int(y/(img_size-1)*255)
    cv2.imwrite('opencv_coord.jpg', img)

test_opencv_coordinate()
```

[Y,X] (B,G,R)

Some Basic Transformations
In PPT

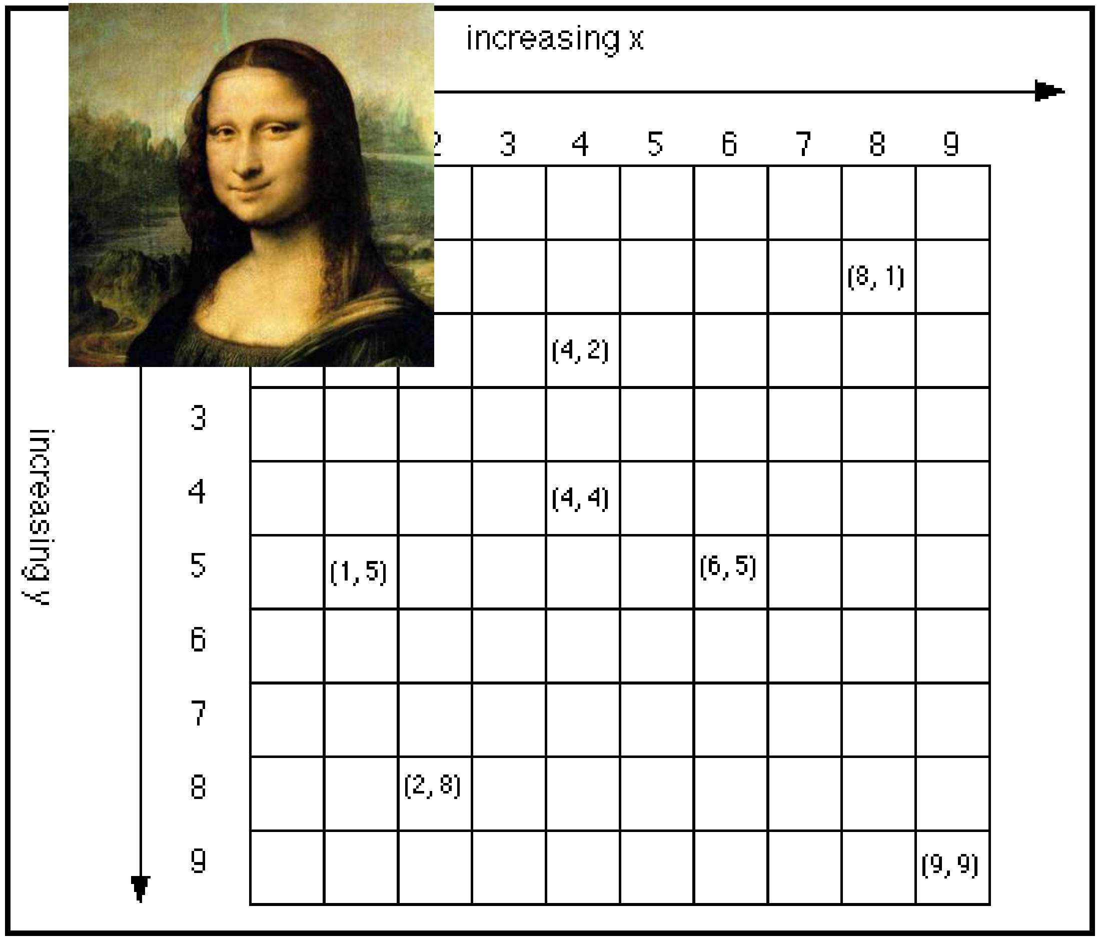
Scaling

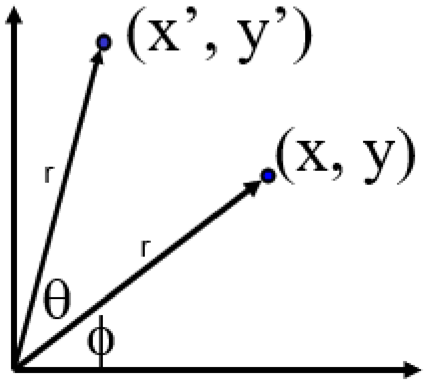


$$x' = ax$$

$$y' = by$$

Rotation (around (0,0))





$x = r \cos(\phi)$
$y = r \sin(\phi)$
$x' = r \cos(\phi + \theta)$
$y' = r \sin(\phi + \theta)$

Trigonometric identity for angle sum

$x' = r \cos(\phi) \cos(\theta) - r \sin(\phi) \sin(\theta)$
$y' = r \sin(\phi) \cos(\theta) + r \cos(\phi) \sin(\theta)$

Substitute…

$x' = x\ \mathbf{cos}(\theta) - y\ \mathbf{sin}(\theta)$
$y' = x\ \mathbf{sin}(\theta) + y\ \mathbf{cos}(\theta)$

Represent them by 2x2 Matrix

$$x' = ax$$

$$y' = by$$

$$x' = x\ \mathbf{cos}(\theta) - y\ \mathbf{sin}(\theta)$$
$$y' = x\ \mathbf{sin}(\theta) + y\ \mathbf{cos}(\theta)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{M} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}}_{\textit{scaling matrix S}} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}}_{\mathbf{R}} \begin{bmatrix} x \\ y \end{bmatrix}$$

Represent transformation by 2x2 Matrix

Mirror

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh_x} \\ \boldsymbol{sh_y} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$

Shear

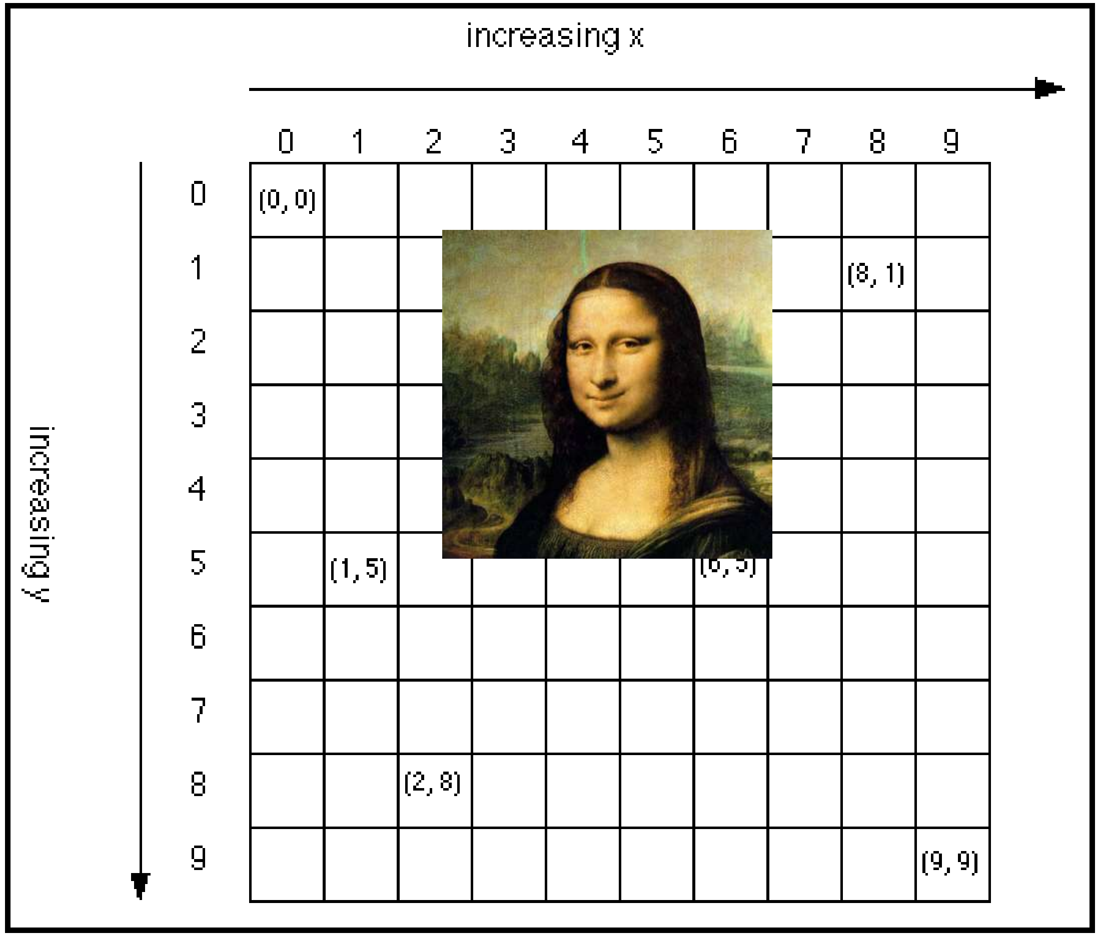
Scale / aspect

$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s_x} & 0 \\ 0 & \boldsymbol{s_y} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{x'} \\ \boldsymbol{y'} \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}$$

Rotation

Can translation be represented with 2x2 Matrix?



$$x' = x + a$$
$$y' = y + b$$

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

?

Homogeneous Coordinates (齐次坐标)

$$(x, y, w) \rightarrow \left(\frac{x}{w}, \frac{y}{w}\right)$$

$$(x, y, 1) \rightarrow (x, y)$$

Represent Translation with 3x3 Matrix

$$x' = x + a$$
$$y' = y + b$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Represent Transformation with 3x3 Matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_{\boldsymbol{x}} & 0 & 0 \\ 0 & \boldsymbol{s}_{\boldsymbol{y}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

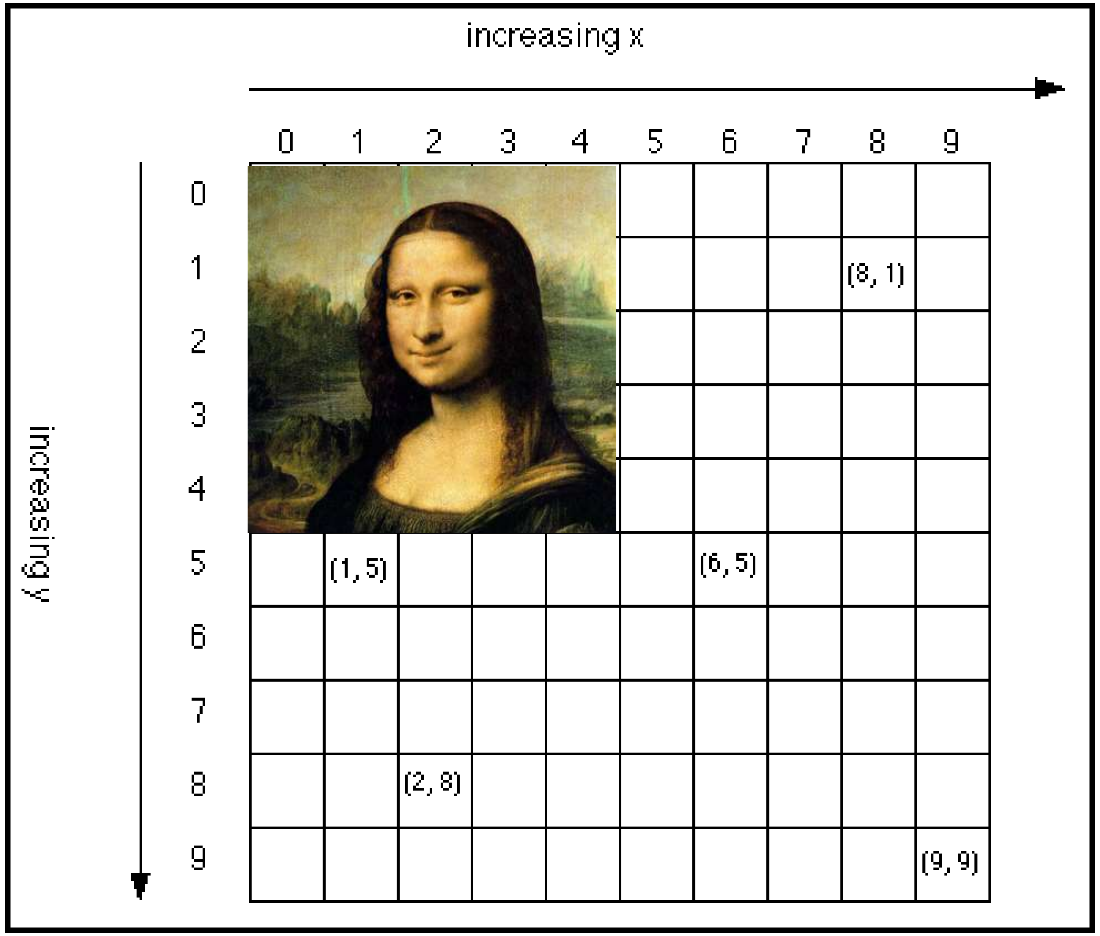
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh}_{\boldsymbol{x}} & 0 \\ \boldsymbol{sh}_{\boldsymbol{y}} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Shear

Transformation Composition

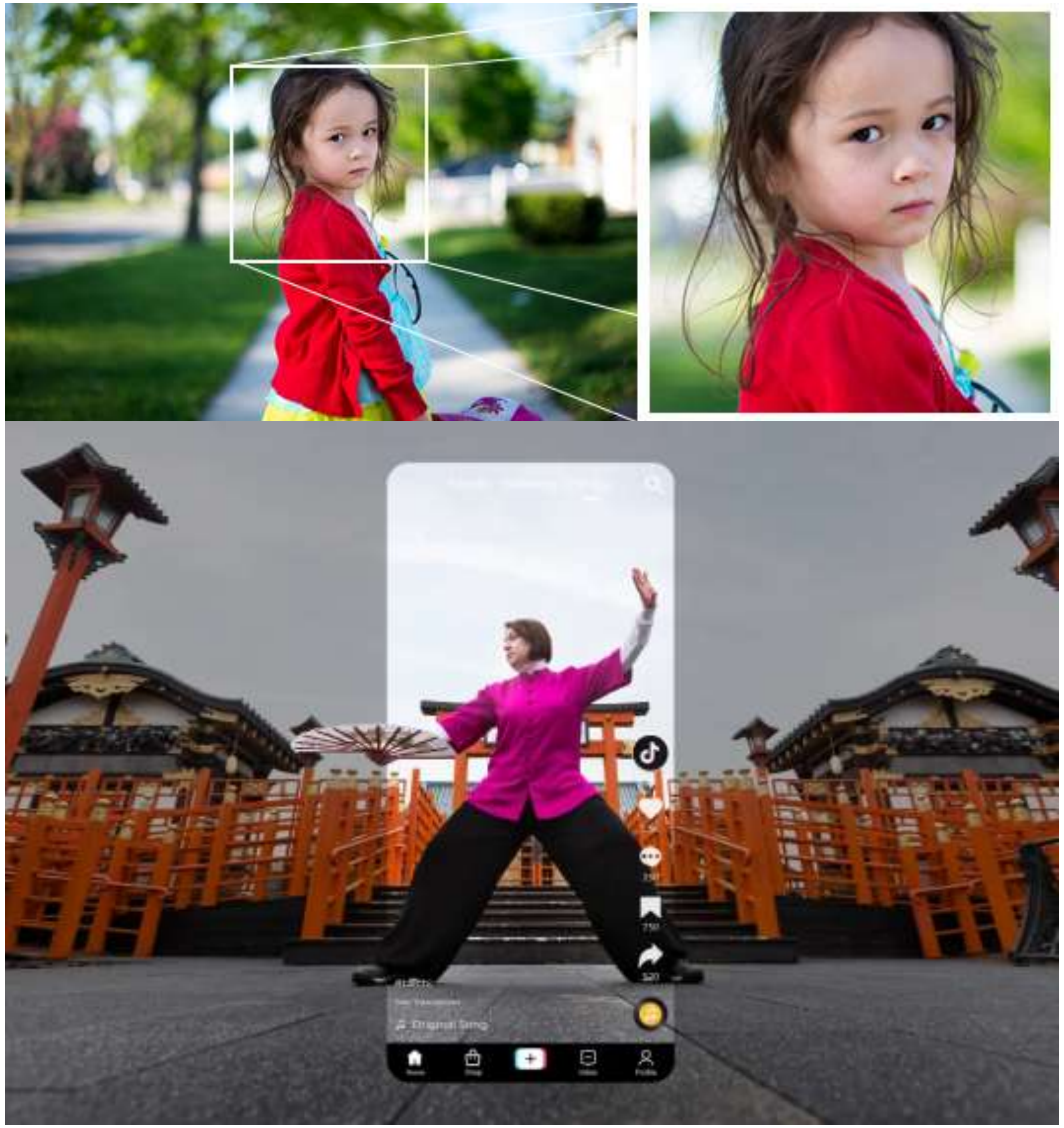


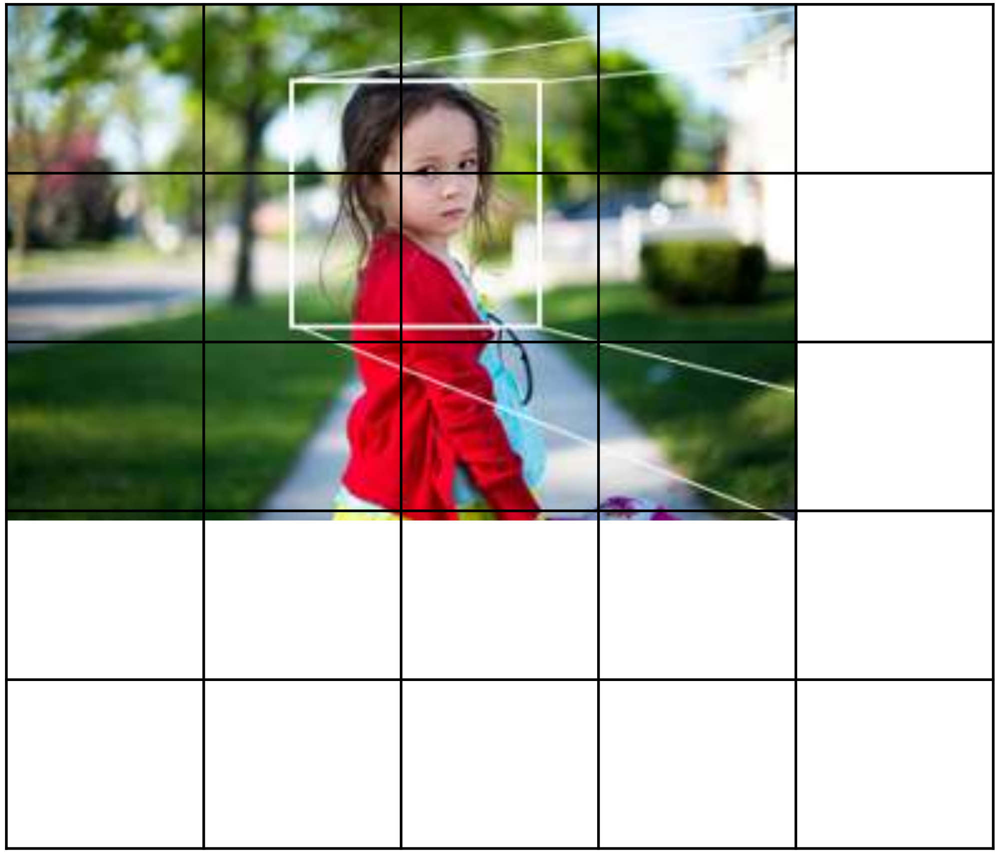
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

$$\mathbf{p}' \quad = \quad T(t_x,t_y) \qquad R(\Theta) \qquad S(s_x,s_y) \qquad \mathbf{p}$$

是否可交换?

Recall the first task





How to Implement them?

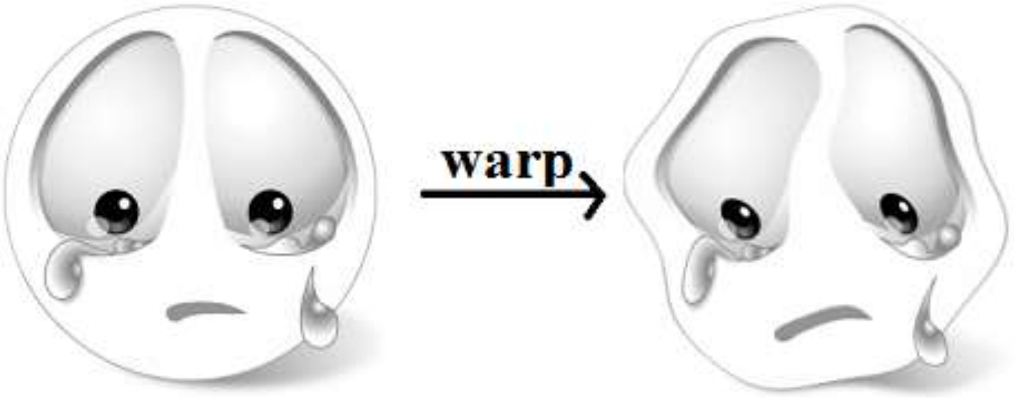
Warp Affine Image

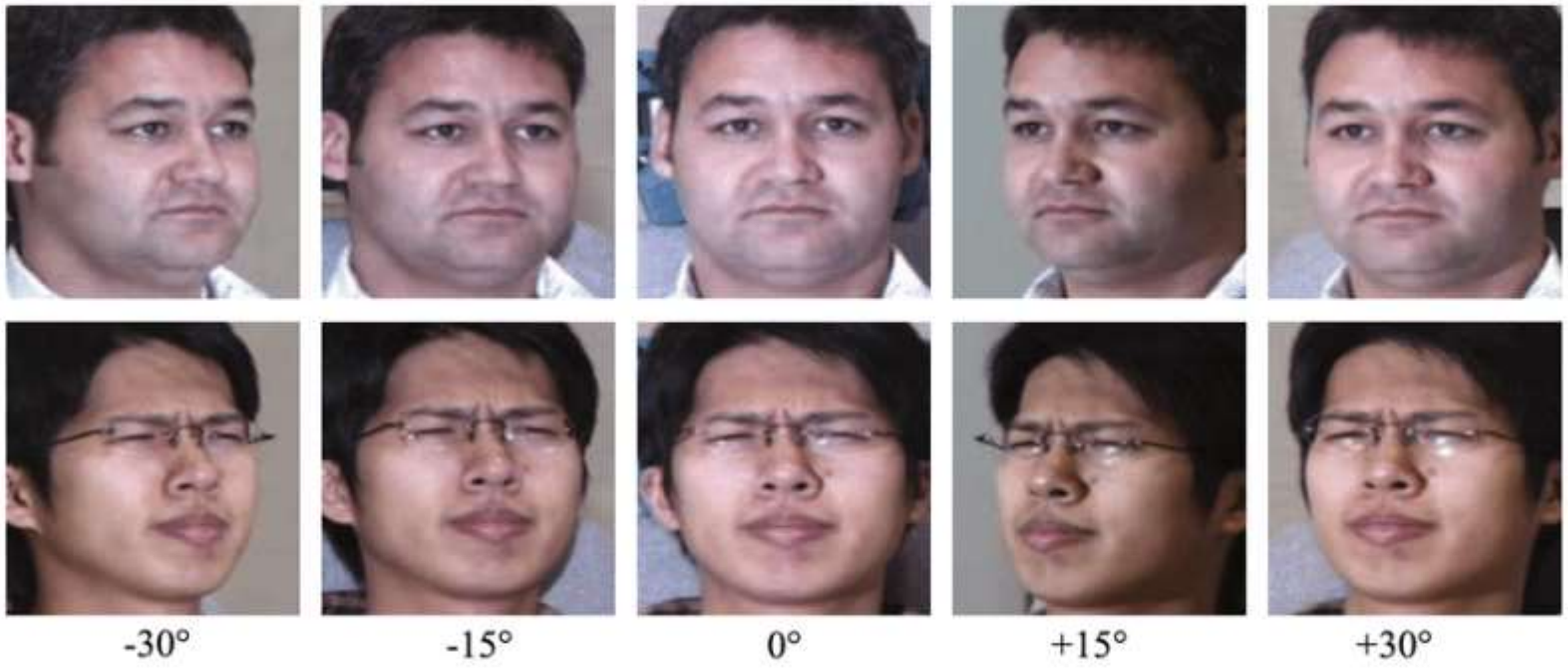
```
# Load your image
img = cv2.imread('image.png')  # Replace with the correct path to your image

# Scaling
scale_mat = np.array([[0.5, 0, 0], [0, 0.8, 0]], dtype=np.float32)
scaled_img = cv2.warpAffine(img, scale_mat, (img.shape[1], img.shape[0]), borderValue=(255,255,255))

# Rotation
theta = 20./180*np.pi
rot_mat = np.array([[np.cos(theta), -np.sin(theta), 0], [np.sin(theta), np.cos(theta), 0]], dtype=np.float32)
rotated_img = cv2.warpAffine(img, rot_mat, (img.shape[1], img.shape[0]), borderValue=(255,255,255))

# Translation
trans_mat = np.array([[1, 0, 100], [0, 1, 100]], dtype=np.float32)
translated_img = cv2.warpAffine(img, trans_mat, (img.shape[1], img.shape[0]), borderValue=(255,255,255))

# Shearing
shear_mat = np.array([[1, 0.3, 0], [0, 1, 0]], dtype=np.float32)
sheared_img = cv2.warpAffine(img, shear_mat, (img.shape[1], img.shape[0]), borderValue=(255,255,255))

# Mirroring (Horizontal flip)
mirror_mat = np.array([[-1, 0, img.shape[1]], [0, 1, 0]], dtype=np.float32)
mirrored_img = cv2.warpAffine(img, mirror_mat, (img.shape[1], img.shape[0]), borderValue=(255,255,255))

pos = (200, 50)
cv2.putText(img, 'Original', pos, cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 2, cv2.LINE_AA)
cv2.putText(scaled_img, 'Scaled', pos, cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 2, cv2.LINE_AA)
cv2.putText(rotated_img, 'Rotated', pos, cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 2, cv2.LINE_AA)
cv2.putText(translated_img, 'Translated', pos, cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 2, cv2.LINE_AA)
cv2.putText(sheared_img, 'Sheared', pos, cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 2, cv2.LINE_AA)
cv2.putText(mirrored_img, 'Mirrored', pos, cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 2, cv2.LINE_AA)


line = np.ones((img.shape[0], 5, 3), dtype=np.uint8) * 0  # Black vertical line
# Concatenate the images with vertical lines between them
img_com = np.hstack((img, line, scaled_img, line, rotated_img, line, translated_img, line, sheared_img, line, mirrored_img))
cv2.imwrite('img_transforms.png', img_com)
```

Composition

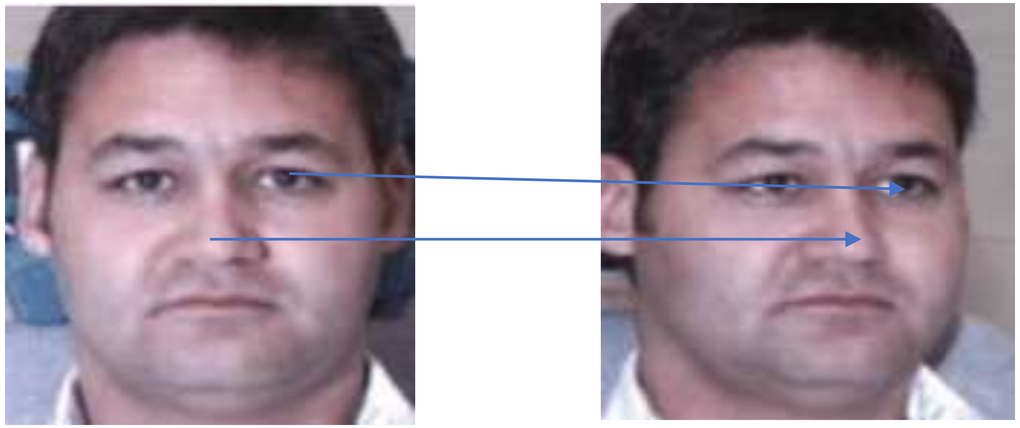
```python
import cv2
import numpy as np

# Load your image
img = cv2.imread('image.png')  # Replace with the correct path to your image

# Scaling
scale_mat = np.array([[0.5, 0, 0], [0, 0.8, 0]], dtype=np.float32)
scaled_img = cv2.warpAffine(img, scale_mat, (img.shape[1], img.shape[0]), borderValue=(255,255,255))

# Rotation
theta = 20./180*np.pi
rot_mat = np.array([[np.cos(theta), -np.sin(theta), 0], [np.sin(theta), np.cos(theta), 0]], dtype=np.float32)
rotated_img = cv2.warpAffine(scaled_img, rot_mat, (img.shape[1], img.shape[0]), borderValue=(255,255,255))

# Translation
trans_mat = np.array([[1, 0, 100], [0, 1, 100]], dtype=np.float32)
translated_img = cv2.warpAffine(rotated_img, trans_mat, (img.shape[1], img.shape[0]), borderValue=(255,255,255))

# Shearing
shear_mat = np.array([[1, 0.3, 0], [0, 1, 0]], dtype=np.float32)
comp_img = cv2.warpAffine(translated_img, shear_mat, (img.shape[1], img.shape[0]), borderValue=(255,255,255))


def pad_row(warp_mat):
    return np.vstack((warp_mat, np.array([0, 0, 1], dtype=np.float32)))

comp_mat = pad_row(shear_mat) @ pad_row(trans_mat) @ pad_row(rot_mat) @ pad_row(scale_mat)
comp_mat_img = cv2.warpAffine(img, comp_mat[:2], (img.shape[1], img.shape[0]), borderValue=(255,255,255))

compare_img = np.vstack((comp_img, comp_mat_img))
cv2.imwrite('comp_compare.png', compare_img)
```

Some Transformation that is not Linear





Correspondence based Transformation



$$x' = x + \Delta x$$
$$y' = y + \Delta y$$

Per Pixel

How to get per pixel correspondence?

Optical Flow

$$I'[y + \Delta y, x + \Delta x] = I[y, x]$$

Horn&Schunck Optical Flow

Brightness constancy assumption

$$f(x,y,t) = f(x+dx, y+dy, t+dt)$$

Taylor Series

$$f(x,y,t) = f(x,y,t) + \frac{\partial f}{\partial x}dx + \frac{\partial f}{\partial y}dy + \frac{\partial f}{\partial t}dt$$

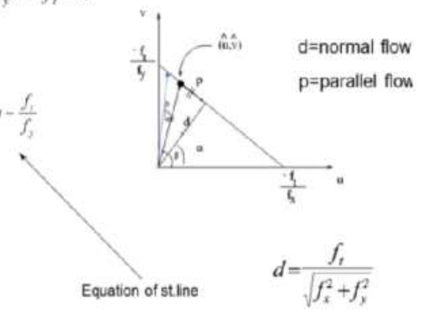
$$f_x dx + f_y dy + f_t dt = 0$$

$$f_x u + f_y v + f_t = 0$$

Interpretation of optical flow eq

$$f_x u + f_y v + f_t = 0$$

$$v = -\frac{f_x}{f_y}u - \frac{f_t}{f_y}$$



$$d = \frac{f_t}{\sqrt{f_x^2 + f_y^2}}$$

Lucas & Kanade (Least Squares)

- Optical flow eq

$$f_x u + f_y v = -f_t$$

- Consider 3 by 3 window

$$f_{x1}u + f_{y1}v = -f_{t1}$$
$$\vdots$$
$$f_{x9}u + f_{y9}v = -f_{t9}$$

$$\begin{bmatrix} f_{x1} & f_{y1} \\ \vdots & \vdots \\ f_{x9} & f_{y9} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f_{t1} \\ \vdots \\ -f_{t9} \end{bmatrix}$$

$$\boxed{\mathbf{Au} = \mathbf{f}_t}$$

$$\mathbf{Au} = \mathbf{f}_t$$

$$\mathbf{A}^T\mathbf{Au} = \mathbf{A}^T\mathbf{f}_t$$

$$\mathbf{u} = \boxed{(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T}\mathbf{f}_t$$

Pseudo Inverse

$$\min \sum (f_{xi}u + f_{yi}v + f_t)^2$$

$$\sum (f_{xi}u + f_{yi}v + f_{ti})f_{xi} = 0$$

$$\sum (f_{xi}u + f_{yi}v + f_{ti})f_{yi} = 0$$

Lucas & Kanade

$$\sum (f_{xi}u + f_{yi}v + f_{ti})f_{xi} = 0$$

$$\sum (f_{xi}u + f_{yi}v + f_{ti})f_{yi} = 0$$

$$\sum f_{xi}^2 u + \sum f_{xi}f_{yi}v = -\sum f_{xi}f_{ti}$$

$$\sum f_{xi}f_{yi}u + \sum f_{yi}^2 v = -\sum f_{yi}f_{ti}$$

$$\begin{bmatrix} \sum f_{xi}^2 & \sum f_{xi}f_{yi} \\ \sum f_{xi}f_{yi} & \sum f_{yi}^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum f_{xi}f_{ti} \\ -\sum f_{yi}f_{ti} \end{bmatrix}$$

$$u = \frac{-\sum f_{yi}^2 \sum f_{xi}f_{ti} + \sum f_{xi}f_{yi} \sum f_{yi}f_{ti}}{\sum f_{xi}^2 \sum f_{yi}^2 - (\sum f_{xi}f_{yi})^2}$$

$$v = \frac{\sum f_{xi}f_{ti} \sum f_{xi}f_{yi} - \sum f_{xi}^2 \sum f_{yi}f_{ti}}{\sum f_{xi}^2 \sum f_{yi}^2 - (\sum f_{xi}f_{yi})^2}$$

Least Squares Fit

Lucas-Kanade without pyramids

Fails in areas of large motion

Lucas-Kanade with Pyramids

Taken from the Lecture video from the UCF CRCV course by Prof Mubarak Shah:

https://www.youtube.com/watch?v=KoMTYnlNNnc

Optical Flow

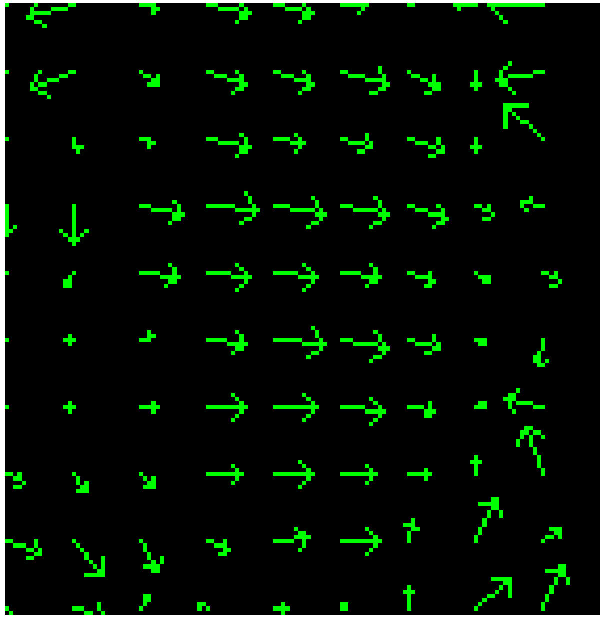
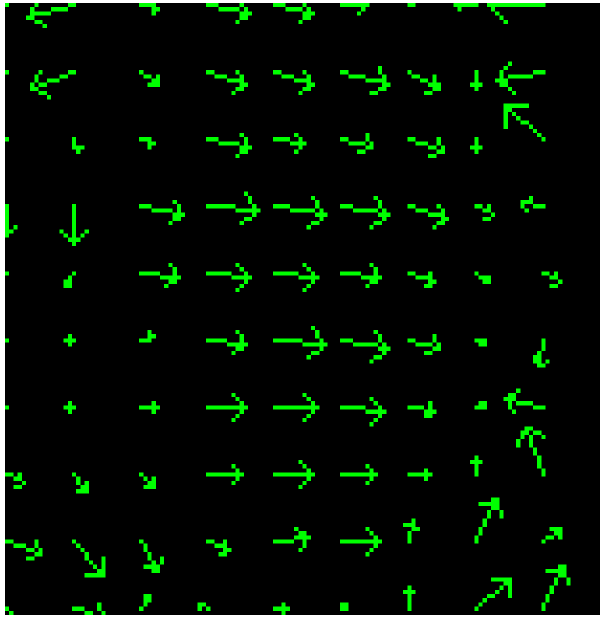
  

Image Warping based on Optical Flow

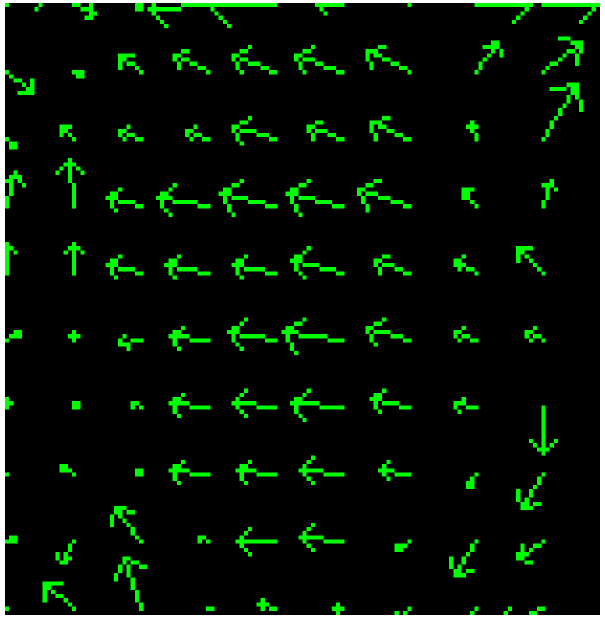


$$I'[y + \Delta y, x + \Delta x] = I[y, x]$$

for [y, x] in Grid

Image Warping based on Optical Flow

Backward Warping



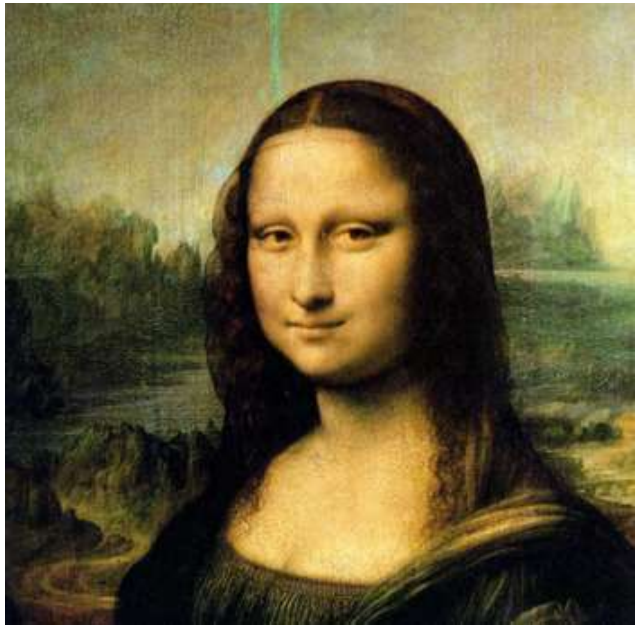
$I'[y',x'] = I[y' + \Delta y', x' + \Delta x']$

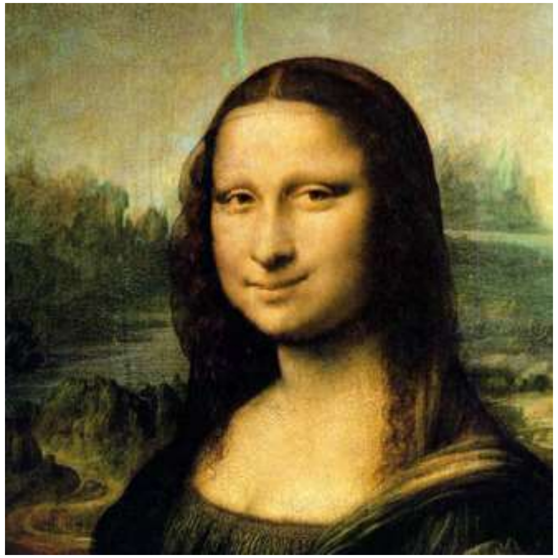
for [y’, x’] in Grid

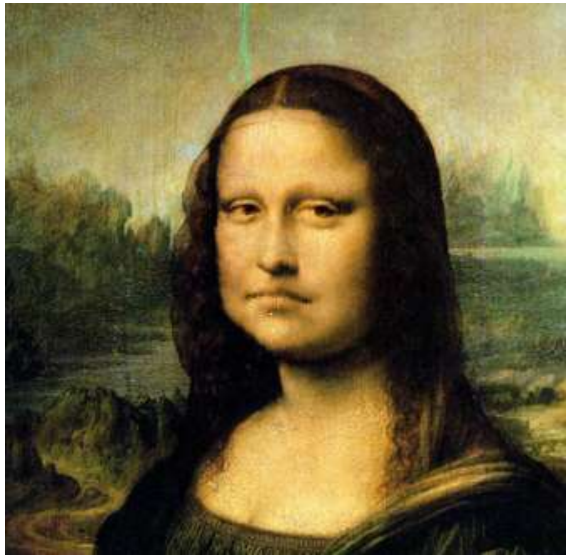
Backward Warping

Most times better than forward warping

How to solve the second task?

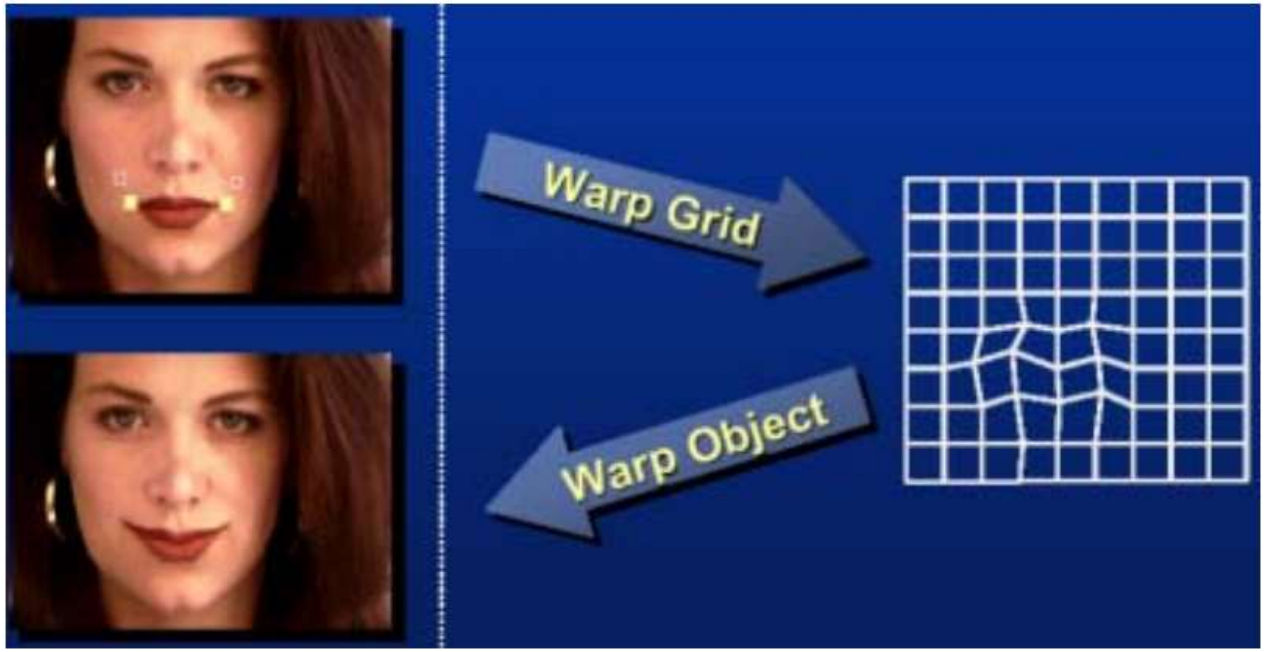


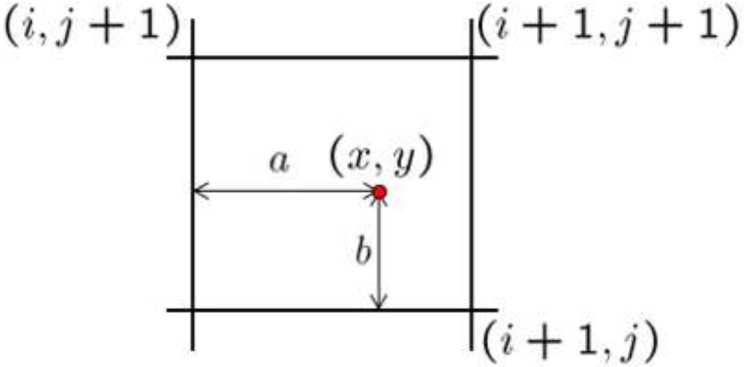




Control

Grid based Image Warping



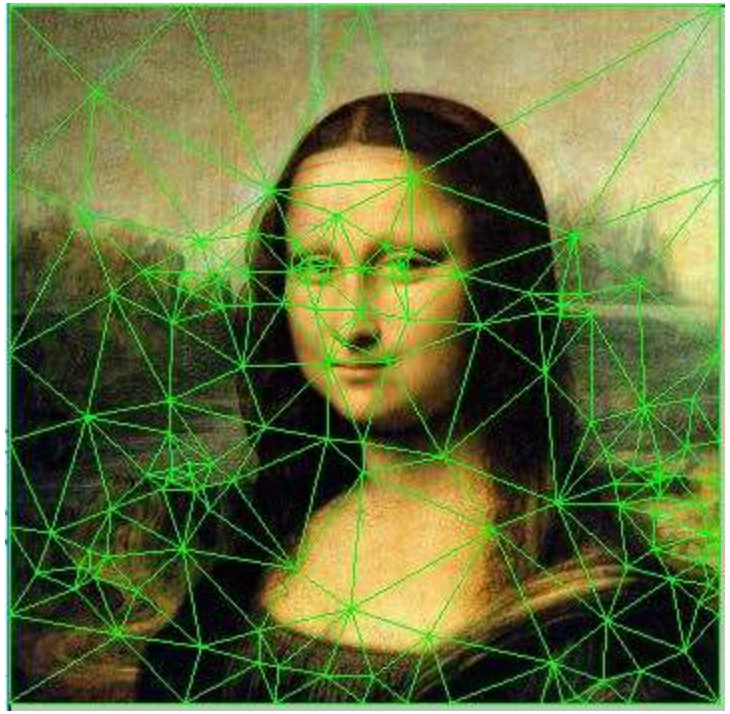


$$
\begin{aligned}
f(x,y) = \;& (1-a)(1-b) && f[i,j] \\
& +a(1-b) && f[i+1,j] \\
& +ab && f[i+1,j+1] \\
& +(1-a)b && f[i,j+1]
\end{aligned}
$$

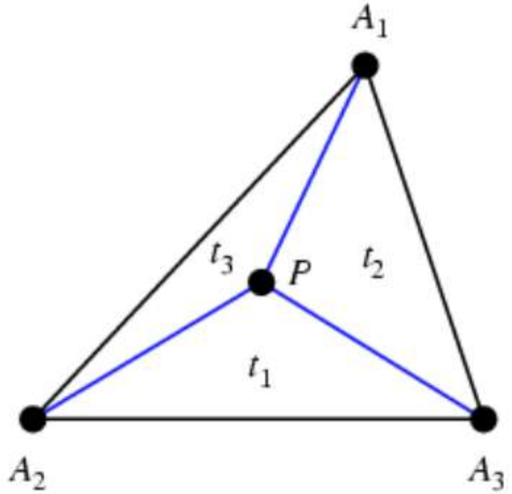
Image Grid Warping

Bilinear Interpolation

Mesh based Image Warping



Delaunay Triangulation

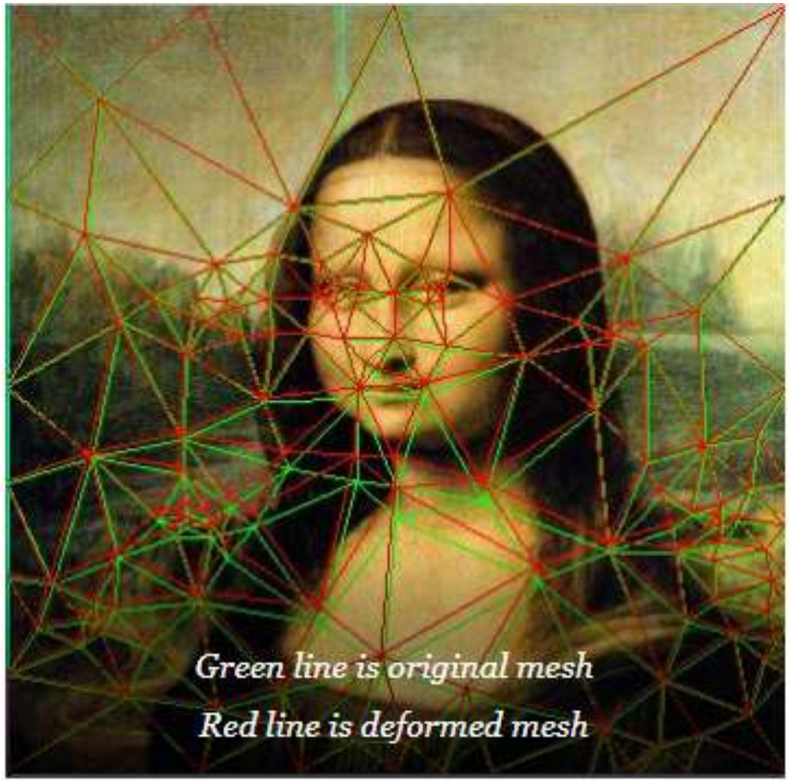


$$P = t_1 A_1 + t_2 A_2 + t_3 A_3$$

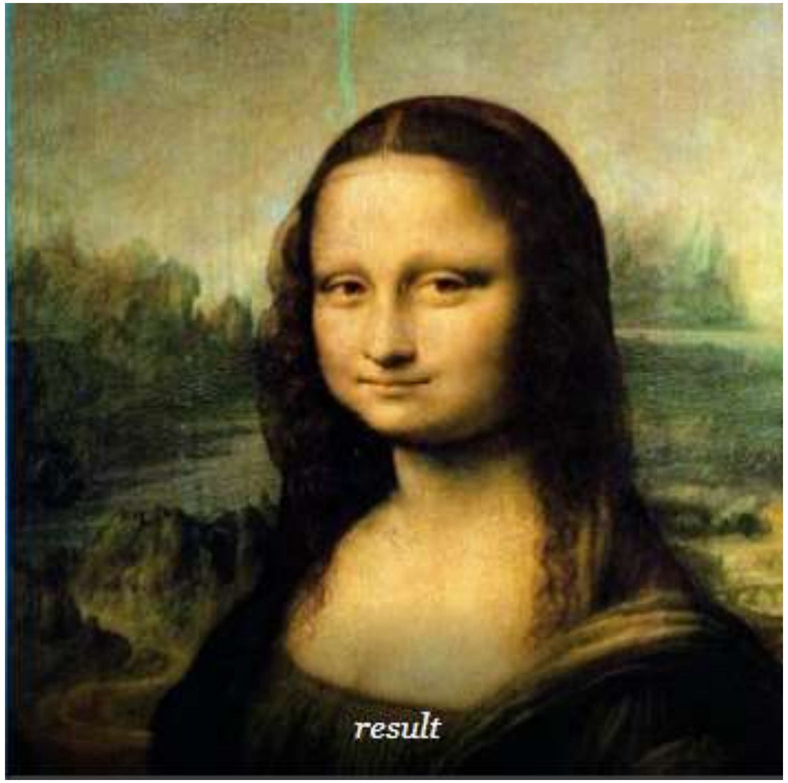
$$t_1 + t_2 + t_3 = 1$$

Barycentric coordinates

Mesh based Image Warping



Mesh Deformation

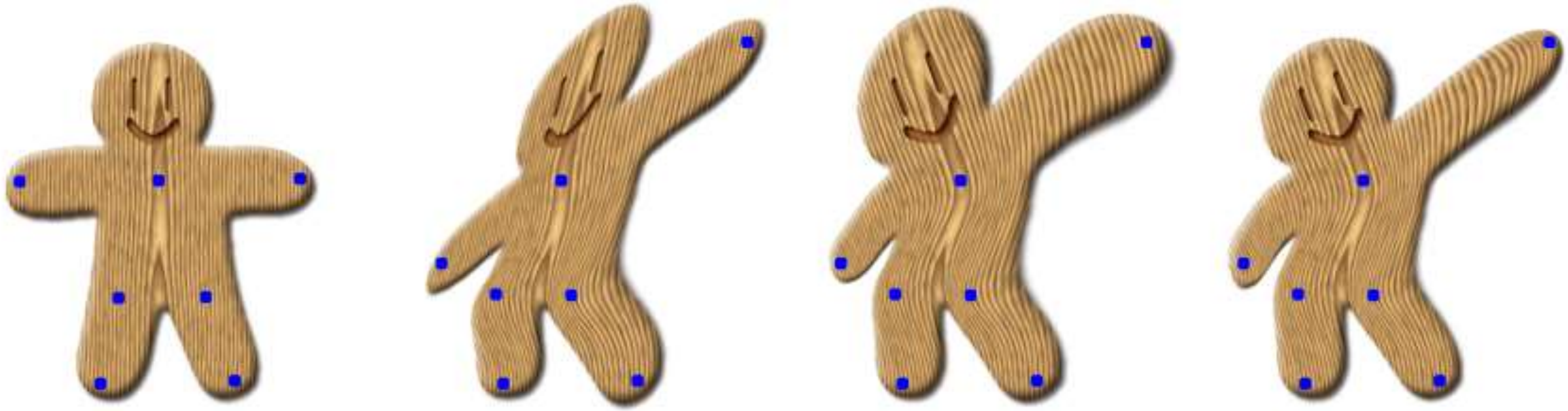


Barycentric Interpolation

Point guided deformation

Image Deformation Using Moving Least Squares

Scott Schaefer[*] Texas A&M University
Travis McPhail[†] Rice University
Joe Warren[‡] Rice University



(a) (b) (c) (d)

Schaefer et al. TOG 2006

Moving Least Squares

What is a good local warping function $T(p) \rightarrow q$?

- Interpolation: need to satisfy control points $T(p_i) = q_i$
- Smoothness: T should be smooth;
- Identity: if $p_i = q_i$, T should be an identity mapping

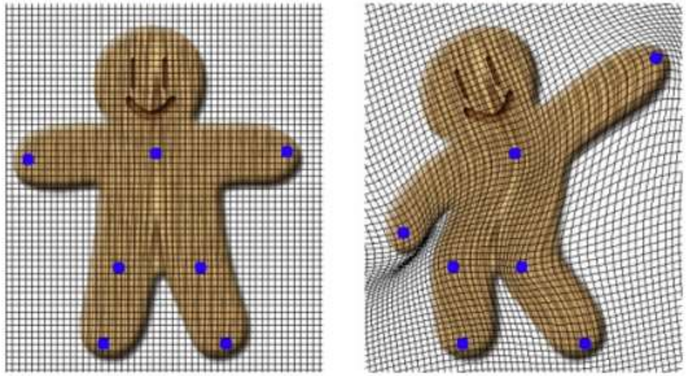
Triangulation-based methods:

$argmin_T \left|\left|T(p_i) - q_i\right|\right|^2$ for 3 vertices in each triangle

Moving least squares:

$argmin_T \, w_i \left|\left|T(p_i) - q_i\right|\right|^2$ for all the control points

Where $w_i = \frac{1}{|p_i - v|^{2\alpha}}$



v: current pixel

α: hyper-parameters

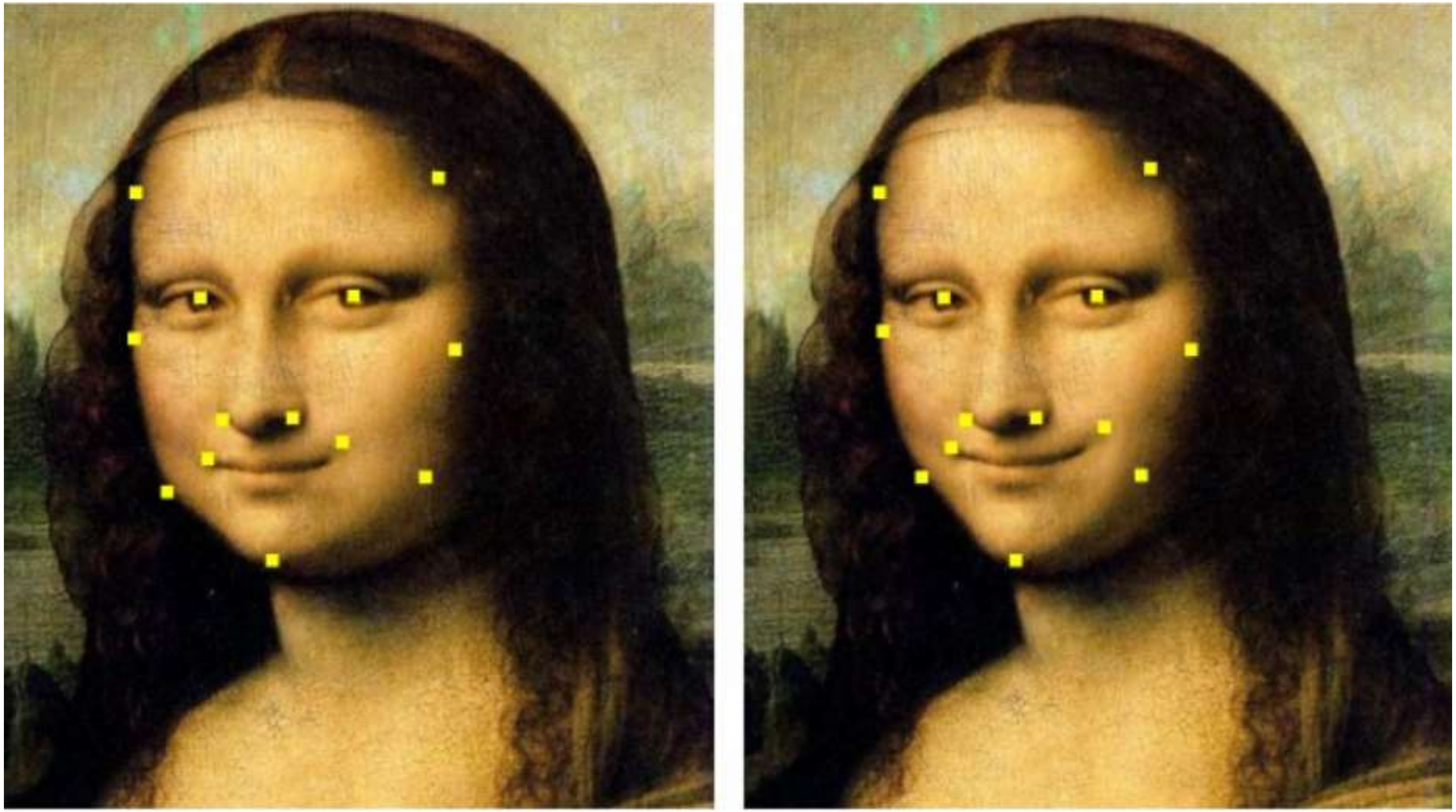
p_i: source control points

q_i: target control points

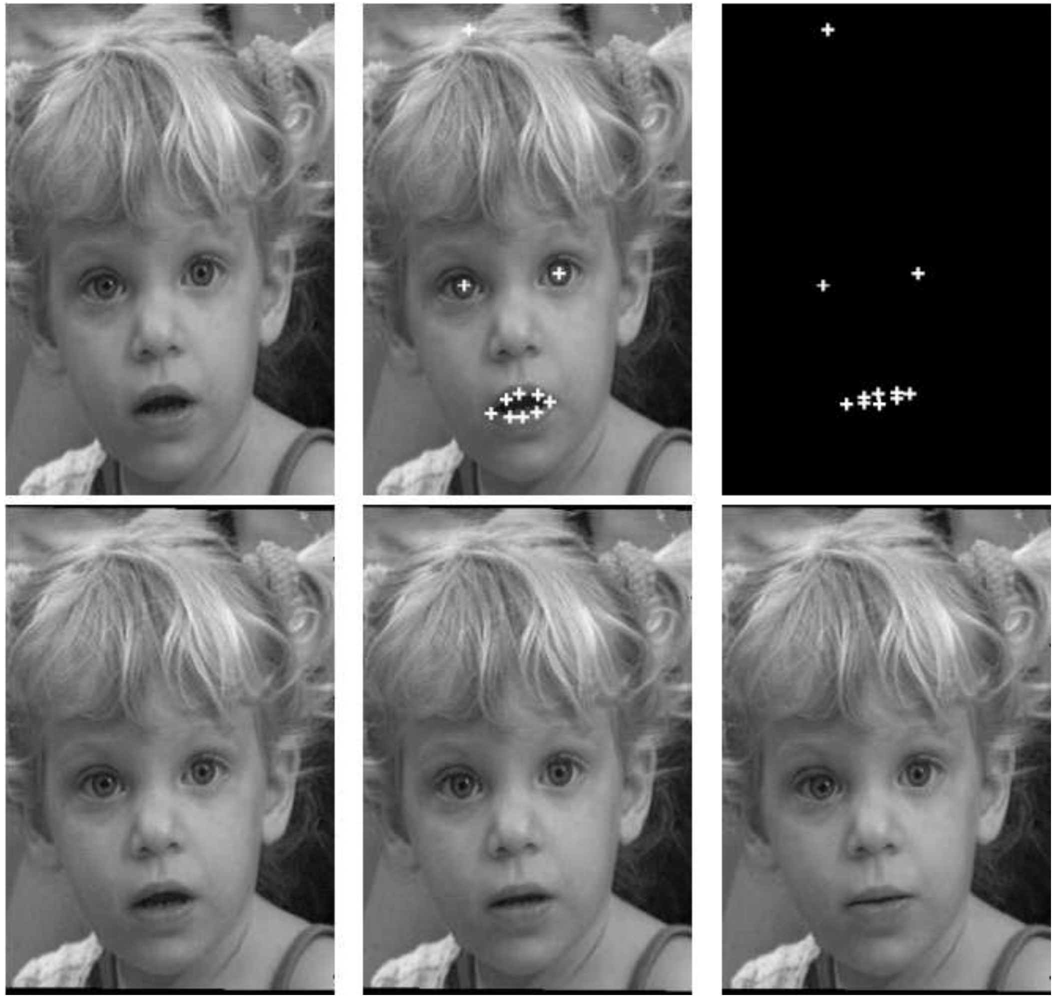
Image Deformation Using Moving Least Squares. Schaefer et al. SIGGRAPH 2006

Moving Least Squares

Moving Least Squares



Radial Basis Functions



Recall that our mapping is defined for each coordinate separately, therefore we are looking for a transformation $T = (T_U(x,y), T_V(x,y))$ such that

$$T_U \in \left\{ f \;\middle|\; f(x^i, y^i) = u^i \quad , \quad i = 1, 2, \ldots, N \right\}$$

$$T_V \in \left\{ f \;\middle|\; f(x^i, y^i) = v^i \quad , \quad i = 1, 2, \ldots, N \right\}$$

$$J(T_U) + J(T_V) \text{ is minimal.}$$

This is another approximation to the actual underlying variational problem, namely the minimization of the total warping induced by the mapping. Minimizing $J(T_U) + J(T_V)$ can be performed by the separate minimization of $J(T_U)$ and $J(T_V)$.

With this formulation in mind, it is known that the choice $g(t) = t^2 \log t$ (with $g(0) = 0$) provides a uniquely solvable interpolation problem (3) − (4) with $m = 1$, the solution of which minimizes the functional J [6]. Thus the transformation $T = (T_U, T_V)$ will be of the form

$$T(x,y) = \left(\alpha_1 + \alpha_2 x + \alpha_3 y + \sum_{i=1}^{N} a_i g_i(x,y) \quad , \quad \beta_1 + \beta_2 x + \beta_3 y + \sum_{i=1}^{N} b_i g_i(x,y) \right) \qquad (5)$$

with $g_i(x,y) = \| (x - x^i, y - y^i) \|^2 \cdot \log(\| (x - x^i, y - y^i) \|)$. The computation of the coefficients in (5) involves the solution of two square linear systems of size $N + 3$ (with the same matrix in each case). An algebraic treatment of the mapping (5) is given in [2].

Image Warping by Radial Basis Functions Application to Facial Expressions. Graphical Models and Image Processing, 1994.

Recall Research

文献阅读

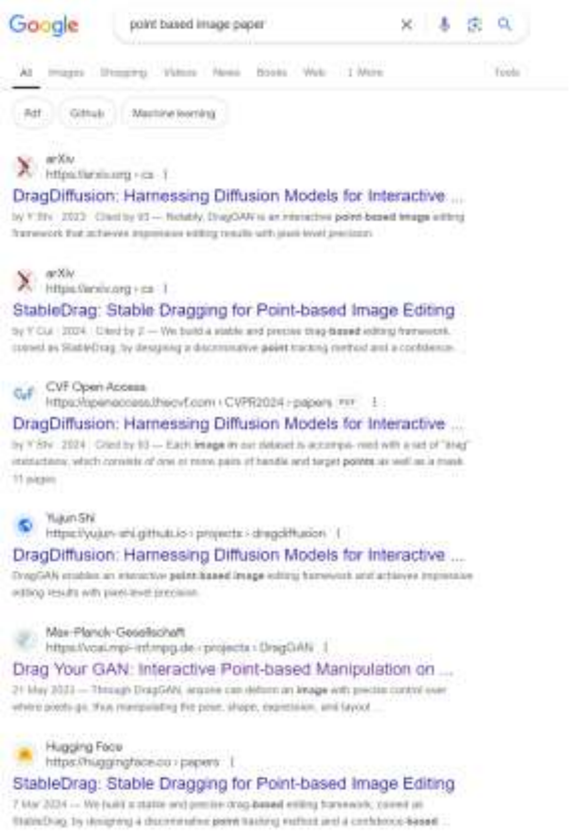
数学建模

算法实现

如何阅读文献

如何找到相关的合适的文献

Key Words: “Image Warping/Manipulation”
+ “Point based”
+ “Paper” + “Github”



https://huggingface.co/papers

Drag Your Gan

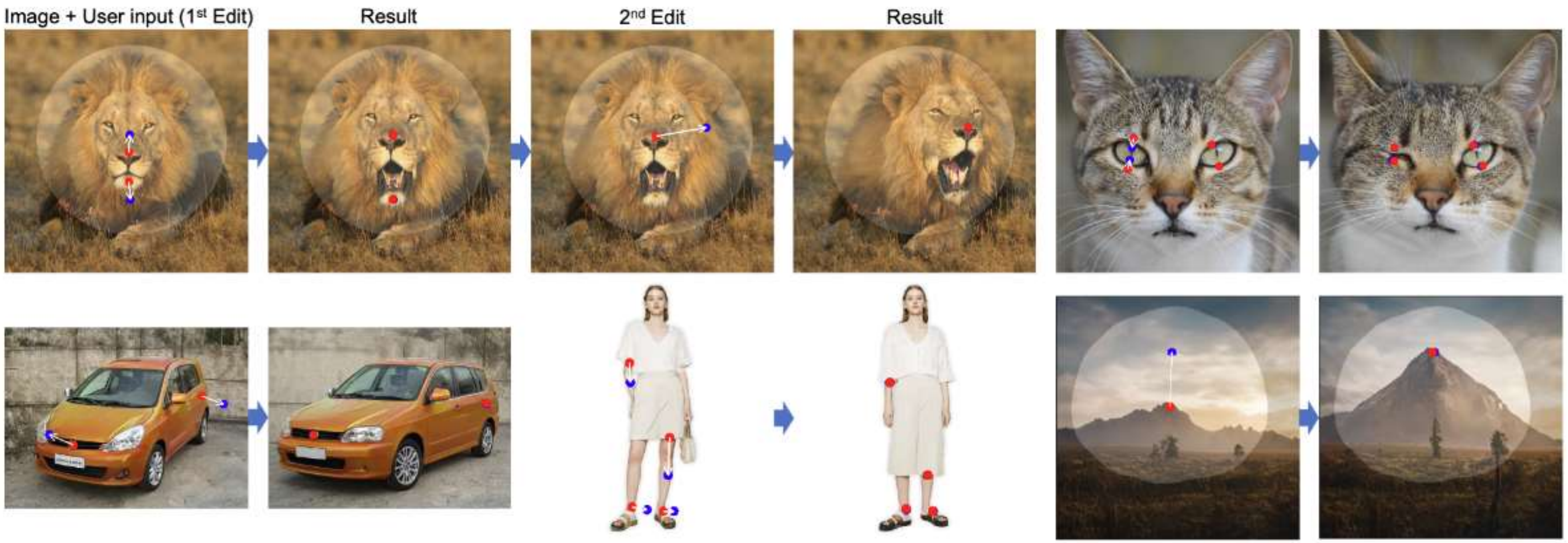
作者
论文发表信息

Drag Your GAN: Interactive Point-based Manipulation on the Generative Image Manifold

Xingang Pan [1,2] Ayush Tewari [3] Thomas Leimkühler [1] Lingjie Liu [1,4] Abhimitra Meka [5] Christian Theobalt [1,2]

[1]Max Planck Institute for Informatics [2]Saarbrücken Research Center for Visual Computing, Interaction and AI [3]MIT [4]University of Pennsylvania [5]Google AR/VR

SIGGRAPH 2023 Conference Proceedings



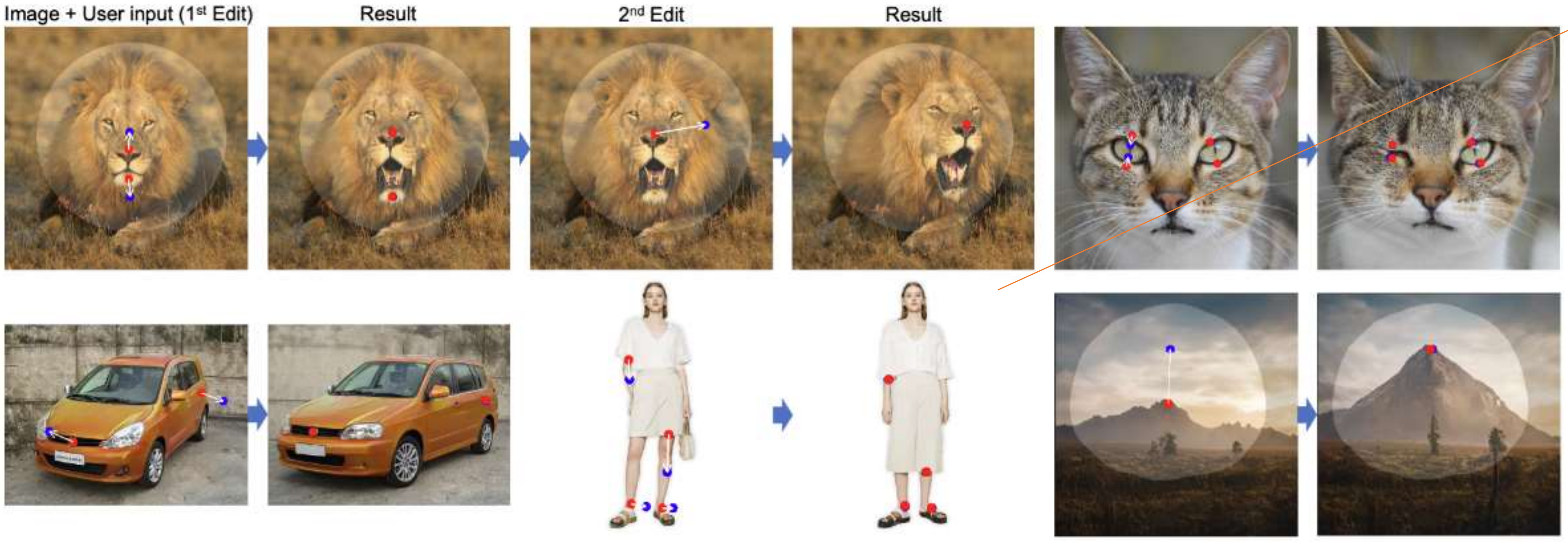
Drag Your Gan

Drag Your GAN: Interactive Point-based Manipulation on the Generative Image Manifold

Xingang Pan [1,2] Ayush Tewari [3] Thomas Leimkühler [1] Lingjie Liu [1,4] Abhimitra Meka [5] Christian Theobalt [1,2]

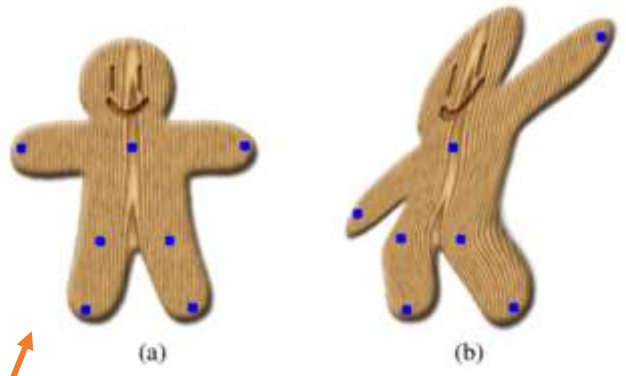
[1]Max Planck Institute for Informatics [2]Saarbrücken Research Center for Visual Computing, Interaction and AI [3]MIT [4]University of Pennsylvania [5]Google AR/VR

SIGGRAPH 2023 Conference Proceedings

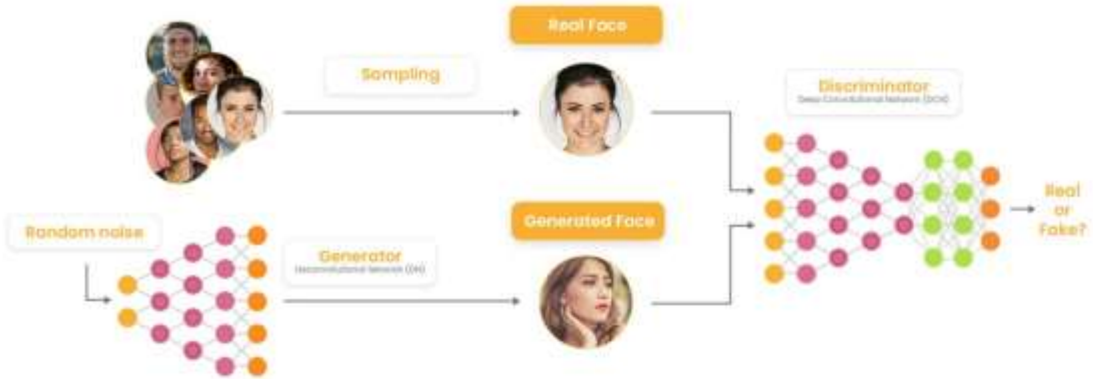


Title & Teaser

带着思考看论文



Drag Your GAN: Interactive Point-based Manipulation on the Generative Image Manifold



利用GAN生成模型的先验

带着观点看Pipeline

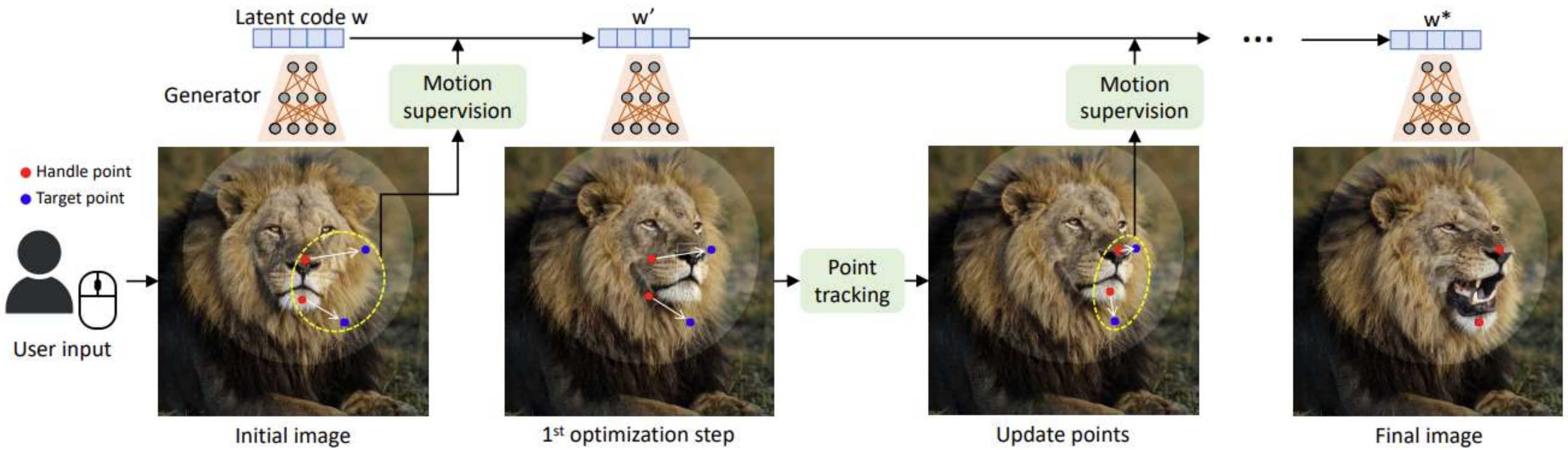


Fig. 2. Overview of our pipeline. Given a GAN-generated image, the user only needs to set several handle points (red dots), target points (blue dots), and optionally a mask denoting the movable region during editing (brighter area). Our approach iteratively performs *motion supervision* (Sec. 3.2) and *point tracking* (Sec. 3.3). The motion supervision step drives the handle points (red dots) to move towards the target points (blue dots) and the point tracking step updates the handle points to track the object in the image. This process continues until the handle points reach their corresponding target points.

精度论文 （如何条理清晰地写论文）

Synthesizing visual content that meets users' needs often requires flexible and precise controllability of the pose, shape, expression, and layout of the generated objects. Existing approaches gain controllability of generative adversarial networks (GANs) via manually annotated training data or a prior 3D model, which often lack flexibility, precision, and generality. In this work, we study a powerful yet much less explored way of controlling GANs, that is, to "drag" any points of the image to precisely reach target points in a user-interactive manner, as shown in Fig.1. To achieve this, we propose *DragGAN*, which consists of two main components: 1) a feature-based motion supervision that drives the handle point to move towards the target position, and 2) a new point tracking approach that leverages the discriminative generator features to keep localizing the position of the handle points. Through *DragGAN*, anyone can deform an image with precise control over where pixels go, thus manipulating the pose, shape, expression, and layout of diverse categories such as animals, cars, humans, landscapes, *etc.* As these manipulations are performed on the learned generative image manifold of a GAN, they tend to produce realistic outputs even for challenging scenarios such as hallucinating occluded content and deforming shapes that consistently follow the object's rigidity. Both qualitative and quantitative comparisons demonstrate the advantage of *DragGAN* over prior approaches in the tasks of image manipulation and point tracking. We also showcase the manipulation of real images through GAN inversion.

CCS Concepts: • **Computing methodologies** → **Computer vision**.

Additional Key Words and Phrases: GANs, interactive image manipulation, point tracking

Abstract
针对什么问题，大概如何解决

精度论文（如何条理清晰地写论文）

1 INTRODUCTION

Deep generative models such as generative adversarial networks (GANs) [Goodfellow et al. 2014] have achieved unprecedented success in synthesizing random photorealistic images. In real-world applications, a critical functionality requirement of such learning-based image synthesis methods is the controllability over the synthesized visual content. For example, social-media users might want to adjust the position, shape, expression, and body pose of a human or animal in a casually-captured photo; professional movie pre-visualization and media editing may require efficiently creating sketches of scenes with certain layouts; and car designers may want to interactively modify the shape of their creations. To satisfy these diverse user requirements, an *ideal* controllable image synthesis approach should possess the following properties 1) *Flexibility*: it should be able to control different spatial attributes including position, pose, shape, expression, and layout of the generated objects or animals; 2) *Precision*: it should be able to control the spatial attributes with high precision; 3) *Generality*: it should be applicable to different object categories but not limited to a certain category. While previous works only satisfy one or two of these properties, we target to achieve them all in this work.

Most previous approaches gain controllability of GANs via prior 3D models [Deng et al. 2020; Ghosh et al. 2020; Tewari et al. 2020] or supervised learning that relies on manually annotated data [Abdal et al. 2021; Isola et al. 2017; Ling et al. 2021; Park et al. 2019; Shen et al. 2020]. Thus, these approaches fail to generalize to new object categories, often control a limited range of spatial attributes or provide little control over the editing process. Recently, text-guided image synthesis has attracted attention [Ramesh et al. 2022; Rombach et al. 2021; Saharia et al. 2022]. However, text guidance lacks precision and flexibility in terms of editing spatial attributes. For example, it cannot be used to move an object by a specific number of pixels.

To achieve flexible, precise, and generic controllability of GANs, in this work, we explore a powerful yet much less explored interactive point-based manipulation. Specifically, we allow users to click any number of handle points and target points on the image and the goal is to drive the handle points to reach their corresponding

both motion supervision and precise point tracking. Specifically, the motion supervision is achieved via a shifted feature patch loss that optimizes the latent code. Each optimization step leads to the handle points shifting closer to the targets; thus point tracking is then performed through nearest neighbor search in the feature space. This optimization process is repeated until the handle points reach the targets. *DragGAN* also allows users to optionally draw a region of interest to perform region-specific editing. Since *DragGAN* does not rely on any additional networks like RAFT [Teed and Deng 2020], it achieves efficient manipulation, only taking a few seconds on a single RTX 3090 GPU in most cases. This allows for live, interactive editing sessions, in which the user can quickly iterate on different layouts till the desired output is achieved.

We conduct an extensive evaluation of *DragGAN* on diverse datasets including animals (lions, dogs, cats, and horses), humans (face and whole body), cars, and landscapes. As shown in Fig.1, our approach effectively moves the user-defined handle points to the target points, achieving diverse manipulation effects across many object categories. Unlike conventional shape deformation approaches that simply apply warping [Igarashi et al. 2005], our deformation is performed on the learned image manifold of a GAN, which tends to obey the underlying object structures. For example, our approach can hallucinate occluded content, like the teeth inside a lion's mouth, and can deform following the object's rigidity, like the bending of a horse leg. We also develop a GUI for users to interactively perform the manipulation by simply clicking on the image. Both qualitative and quantitative comparison confirms the advantage of our approach over UserControllableLT. Furthermore, our GAN-based point tracking algorithm also outperforms existing point tracking approaches such as RAFT [Teed and Deng 2020] and PIPs [Harley et al. 2022] for GAN-generated frames. Furthermore, by combining with GAN inversion techniques, our approach also serves as a powerful tool for real image editing.

2 RELATED WORK

2.1 Generative Models for Interactive Content Creation

Most current methods use generative adversarial networks (GANs) or diffusion models for controllable image synthesis.

Controllability using Unconditional GANs. Several methods have been proposed for editing unconditional GANs by manipulating the input latent vectors. Some approaches find meaningful latent directions via supervised learning from manual annotations or prior 3D models [Abdal et al. 2021; Leimkühler and Drettakis 2021; Patashnik et al. 2021; Shen et al. 2020; Tewari et al. 2020]. Other approaches compute the important semantic directions in the latent space in an unsupervised manner [Härkönen et al. 2020; Shen and Zhou 2020; Zhu et al. 2023]. Recently, the controllability of coarse object position is achieved by introducing intermediate "blobs" [Epstein et al. 2022] or heatmaps [Wang et al. 2022b]. All of these approaches enable editing of either image-aligned semantic attributes such as appearance, or coarse geometric attributes such as object position and pose. While Editing-in-Style [Collins et al. 2020] showcases some spatial attributes editing capability, it can only achieve this by transferring local semantics between different samples. In contrast to these methods, our approach allows users to perform fine-grained control over the spatial attributes using point-based editing.

GANWarping [Wang et al. 2022a] also use point-based editing, however, they only enable out-of-distribution image editing. A few warped images can be used to update the generative model such that all generated images demonstrate similar warps. However, this method does not ensure that the warps lead to realistic images. Further, it does not enable controls such as changing the 3D pose of the object. Similar to us, UserControllableLT [Endo 2022] enables point-based [illegible] forming latent vectors of a GAN. However, this ap[illegible] ports editing using a single point being dragged on the image and does not handle multiple-point constraints well. In addition, the control is not precise, *i.e.*, after editing, the target point is often not reached.

Conventional approaches solve optimization problems with hand-crafted criteria [Brox and Malik 2010; Sundaram et al. 2010], while deep learning-based approaches started to dominate the field in recent years due to better performance [Dosovitskiy et al. 2015; Ilg et al. 2017; Teed and Deng 2020]. These deep learning-based approaches typically use synthetic data with ground truth optical flow to train the deep neural networks. Among them, the most widely used method now is RAFT [Teed and Deng 2020], which estimates optical flow via an iterative algorithm. Recently, Harley *et al.* [2022] combines this iterative algorithm with a conventional "particle video" approach, giving rise to a new point tracking method named PIPs. PIPs considers information across multiple frames and thus handles long-range tracking better than previous approaches.

In this work, we show that point tracking on GAN-generated images can be performed without using any of the aforementioned approaches or additional neural networks. We reveal that the feature spaces of GANs are discriminative enough such that tracking can be achieved simply via feature matching. While some previous works also leverage the discriminative feature in semantic segmentation [Tritrong et al. 2021; Zhang et al. 2021], we are the first to connect the point-based editing problem to the intuition of discriminative GAN features and design a concrete method. Getting rid of additional tracking models allows our approach to run much more efficiently to support interactive editing. Despite the simplicity of our approach, we show that it outperforms the state-of-the-art point tracking approaches including RAFT and PIPs in our experiments.

3 METHOD

This work aims to develop an interactive image manipulation method for GANs where users only need to click on the images to define

Introduction (&Related Work)

回顾研究领域，指出具体问题，具体解决方案 （突出优势）

精度论文（如何条理清晰地写论文）

3 METHOD

This work aims to develop an interactive image manipulation method for GANs where users only need to click on the images to define some pairs of (handle point, target point) and drive the handle points to reach their corresponding target points. Our study is based on the StyleGAN2 architecture [Karras et al. 2020]. Here we briefly introduce the basics of this architecture.

StyleGAN Terminology. In the StyleGAN2 architecture, a 512 dimensional latent code $\boldsymbol{z} \in \mathcal{N}(0, \boldsymbol{I})$ is mapped to an intermediate latent code $\boldsymbol{w} \in \mathbb{R}^{512}$ via a mapping network. The space of $\boldsymbol{w}$ is commonly referred to as $\mathcal{W}$. $\boldsymbol{w}$ is then sent to the generator G to produce the output image $\mathbf{I} = G(\boldsymbol{w})$. In this process, $\boldsymbol{w}$ is copied several times and sent to different layers of the generator G to control different levels of attributes. Alternatively, one can also use different $\boldsymbol{w}$ for different layers, in which case the input would be $\boldsymbol{w} \in \mathbb{R}^{l \times 512} = \mathcal{W}^+$, where l is the number of layers. This less constrained $\mathcal{W}^+$ space is shown to be more expressive [Abdal et al. 2019]. As the generator G learns a mapping from a low-dimensional latent space to a much higher dimensional image space, it can be seen as modelling an image manifold [Zhu et al. 2016].

3.1 Interactive Point-based Manipulation

An overview of our image manipulation pipeline is shown in Fig. 2. For any image $\mathbf{I} \in \mathbb{R}^{3 \times H \times W}$ generated by a GAN with latent code $\boldsymbol{w}$, we allow the user to input a number of handle points $\{\boldsymbol{p}_i = (x_{p,i}, y_{p,i}) | i = 1, 2, ..., n\}$ and their corresponding target points $\{\boldsymbol{t}_i = (x_{t,i}, y_{t,i}) | i = 1, 2, ..., n\}$ (*i.e.*, the corresponding target point of $\boldsymbol{p}_i$ is $\boldsymbol{t}_i$). The goal is to move the object in the image such that the

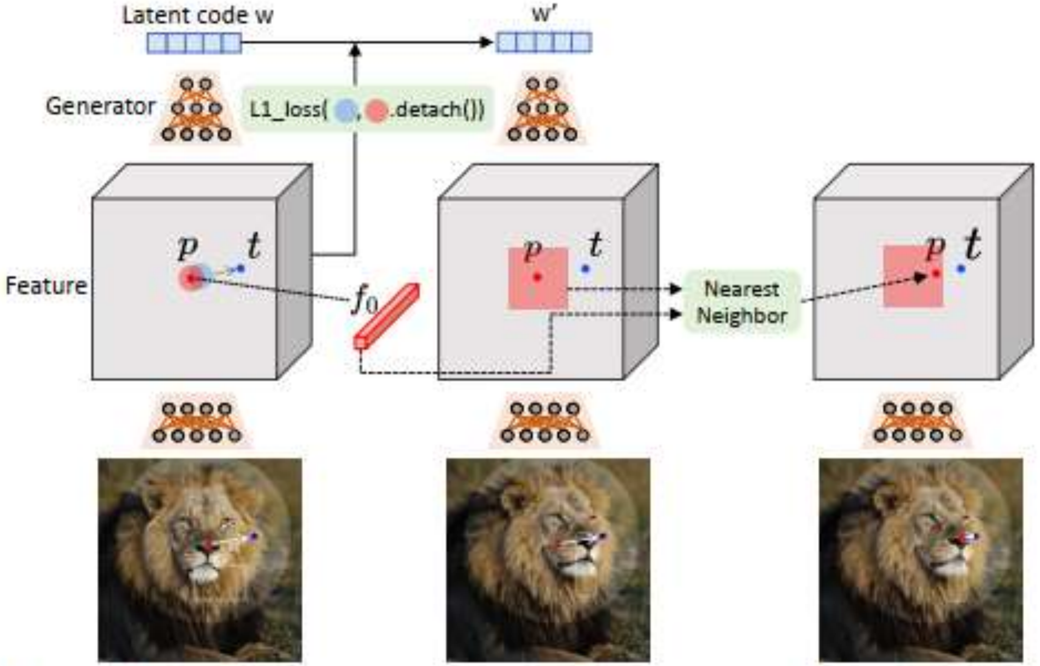


Fig. 3. Method. Our motion supervision is achieved via a shifted patch loss on the feature maps of the generator. We perform point tracking on the same feature space via the nearest neighbor search.

Method

具体实现方法，介绍背景知识，复杂模块图示

精度论文（如何条理清晰地写论文）

Fig. 4. Qualitative comparison of our approach to UserControllableLT [Endo 2022] on the task of moving handle points (red dots) to target points (blue dots). Our approach achieves more natural and superior results on various datasets. More examples are provided in Fig. 10.

Result

和相关工作对比，突出核心优势

精度论文（如何更好地展现论文）

Main demo (accelerated)

Lion

酷炫（可体验）Demo

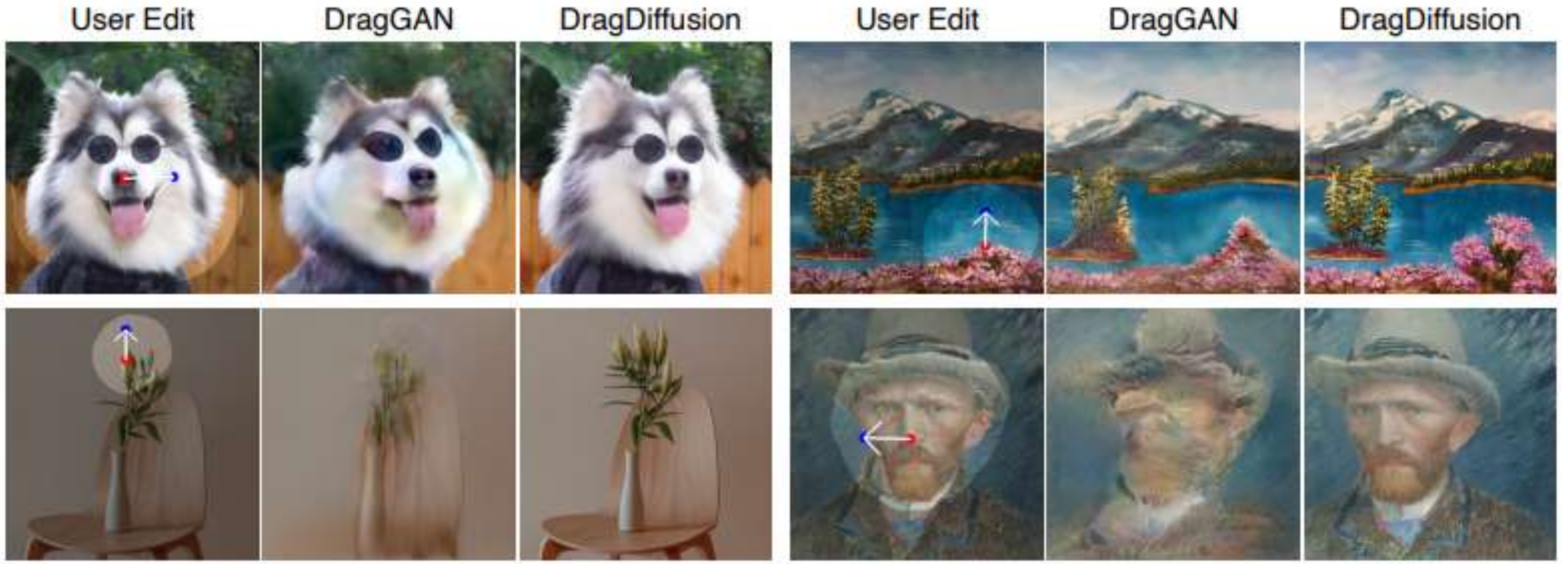
另一个例子 — 快读类似论文

DragDiffusion: Harnessing Diffusion Models for Interactive Point-based Image Editing

Yujun Shi[1] Chuhui Xue[2] Jun Hao Liew[2]
Jiachun Pan[1] Hanshu Yan[2] Wenqing Zhang[2]
Vincent Y. F. Tan[1] Song Bai[2]

[1] National University of Singapore [2] ByteDance

[Paper] [Code] [Video]



另一个例子 — 快读类似论文

DragDiffusion: Harnessing Diffusion Models for Interactive Point-based Image Editing

Yujun Shi[1] Chuhui Xue[2] Jun Hao Liew[2]
Jiachun Pan[1] Hanshu Yan[2] Wenqing Zhang[2]
Vincent Y. F. Tan[1] Song Bai[2]

[1] National University of Singapore [2] ByteDance

[Paper] [Code] [Video]

Abstract

Accurate and controllable image editing is a challenging task that has attracted significant attention recently. Notably, DRAGGAN developed by Pan et al. (2023) [31] is an interactive point-based image editing framework that achieves impressive editing results with pixel-level precision. However, due to its reliance on generative adversarial networks (GANs), its generality is limited by the capacity of pretrained GAN models. In this work, we extend this editing framework to diffusion models and propose a novel approach DRAGDIFFUSION. By harnessing large-scale pretrained diffusion models, we greatly enhance the applicability of interactive point-based editing on both real and diffusion-generated images. Unlike other diffusion-based editing methods that provide guidance on diffusion latents of multiple time steps, our approach achieves efficient yet accurate spatial control by optimizing the latent of only one time step. This novel design is motivated by our observations that UNet features at a specific time step provides

另一个例子 — 快读类似论文

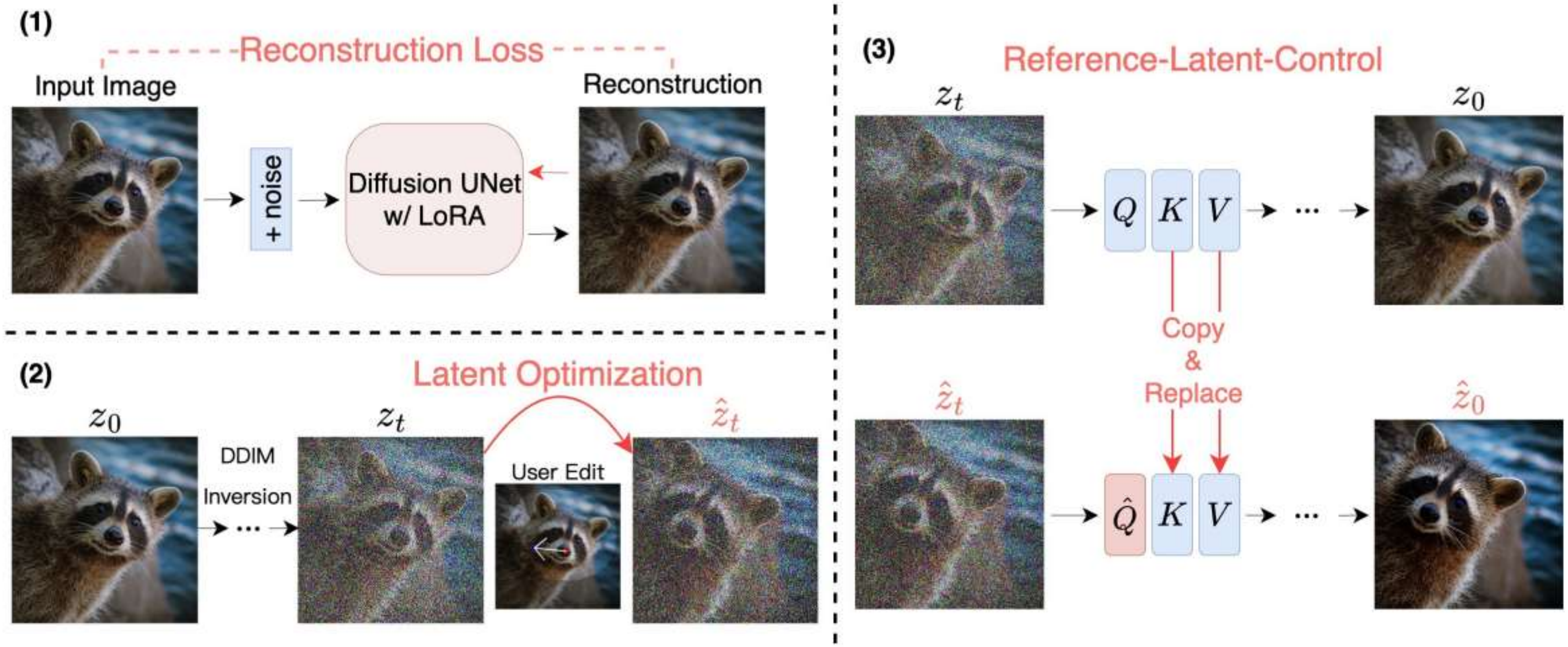
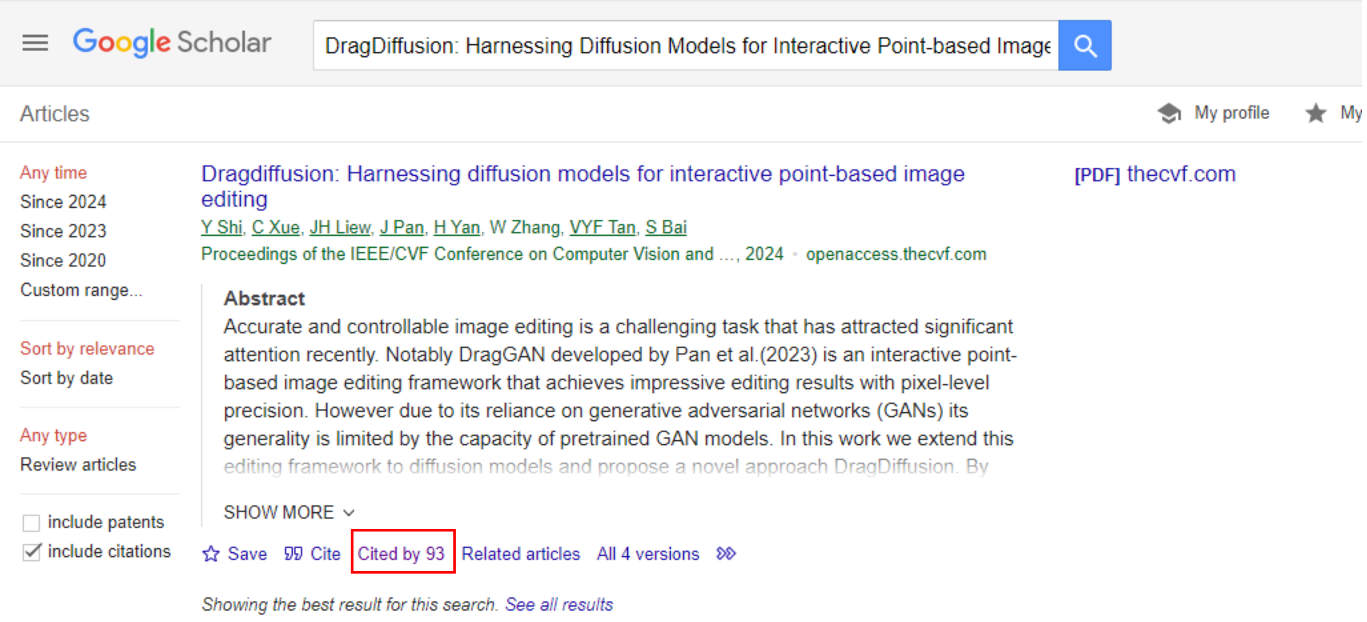
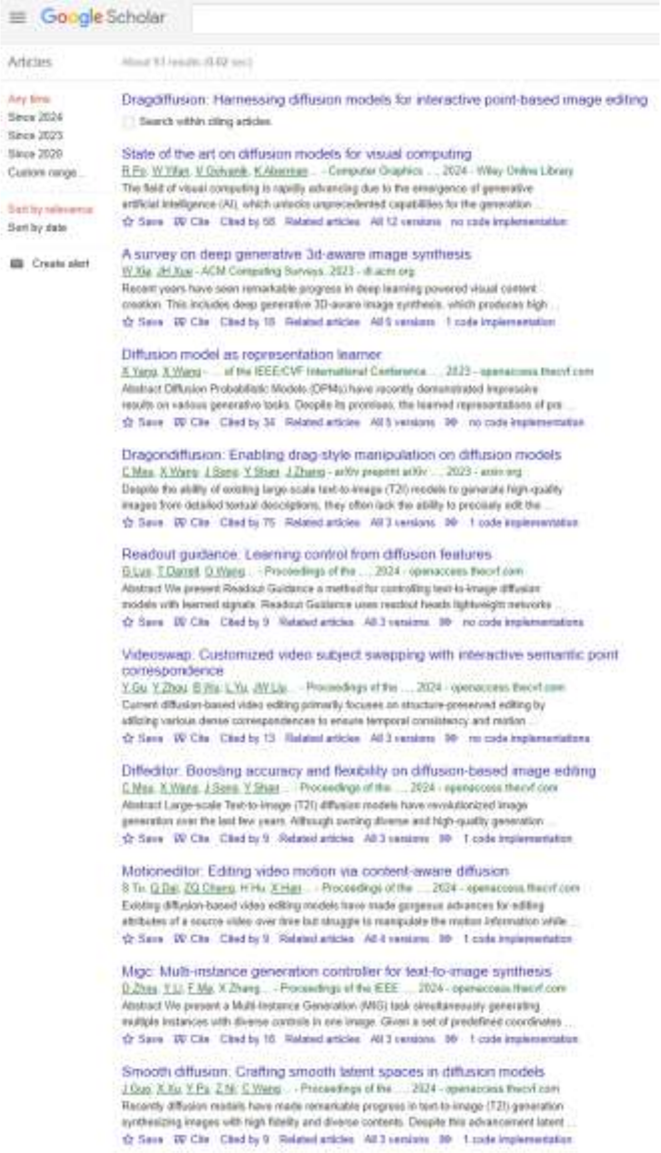


Figure 3. **Overview of DRAGDIFFUSION.** Our approach constitutes three steps: firstly, we conduct identity-preserving fine-tuning on the UNet of the diffusion model given the input image. Secondly, according to the user's dragging instruction, we optimize the latent obtained from DDIM inversion on the input image. Thirdly, we apply DDIM denoising guided by our reference-latent-control on $\hat{z}_t$ to obtain the final editing result $\hat{z}_0$. Figure best viewed in color.

最后一步— 速览同类论文



Google Scholar
DragDiffusion: Harnessing Diffusion Models for Interactive Point-based Image
Articles
My profile
My
Any time
Since 2024
Since 2023
Since 2020
Custom range...
Sort by relevance
Sort by date
Any type
Review articles
include patents
include citations
Dragdiffusion: Harnessing diffusion models for interactive point-based image editing
[PDF] thecvf.com
Y Shi, C Xue, JH Liew, J Pan, H Yan, W Zhang, VYF Tan, S Bai
Proceedings of the IEEE/CVF Conference on Computer Vision and ..., 2024 · openaccess.thecvf.com
Abstract
Accurate and controllable image editing is a challenging task that has attracted significant attention recently. Notably DragGAN developed by Pan et al.(2023) is an interactive point-based image editing framework that achieves impressive editing results with pixel-level precision. However due to its reliance on generative adversarial networks (GANs) its generality is limited by the capacity of pretrained GAN models. In this work we extend this editing framework to diffusion models and propose a novel approach DragDiffusion. By
SHOW MORE
Save
Cite
Cited by 93
Related articles
All 4 versions
Showing the best result for this search. See all results



Google Scholar
Articles
Dragdiffusion: Harnessing diffusion models for interactive point-based image editing
State of the art on diffusion models for visual computing
A survey on deep generative 3d-aware image synthesis
Diffusion model as representation learner
Dragondiffusion: Enabling drag-style manipulation on diffusion models
Readout guidance: Learning control from diffusion features
Videoswap: Customized video subject swapping with interactive semantic point correspondence
Diffeditor: Boosting accuracy and flexibility on diffusion-based image editing
Motioneditor: Editing video motion via content-aware diffusion
Migc: Multi-instance generation controller for text-to-image synthesis
Smooth diffusion: Crafting smooth latent spaces in diffusion models

最后一步— 速览同类论文

StableDrag: Stable Dragging for Point-based Image Editing

Anonymous Authors*
Anonymous Institutions

Paper Supplementary Code arXiv

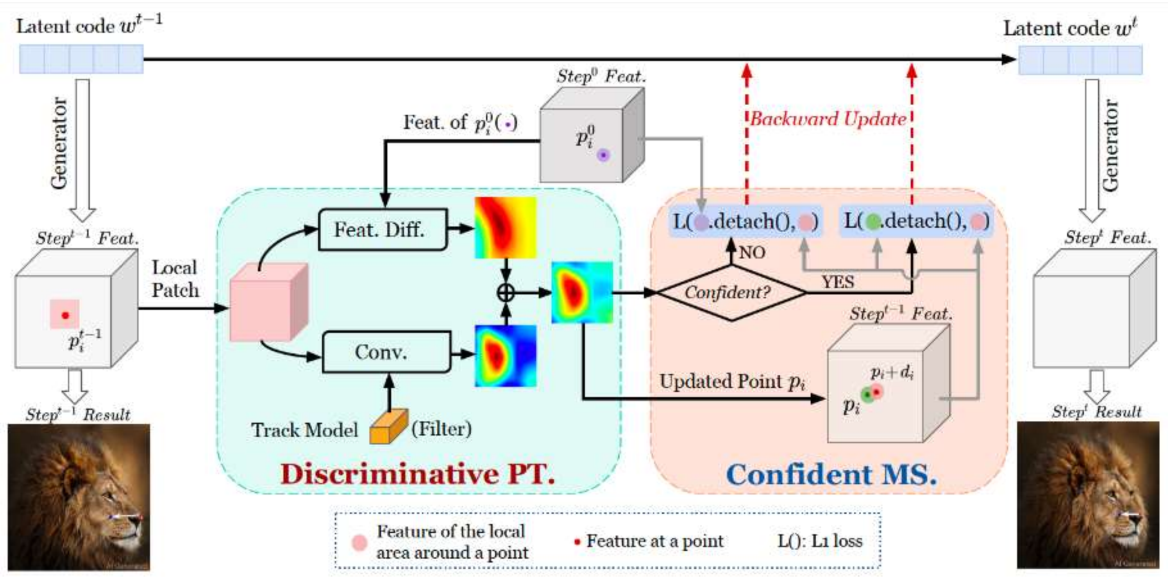


Figure 1. Illustration of our dragging scheme for an intermediate single-step optimization. The core of the dragging pipeline illustrated herein is based on GAN, whereas the one based on diffusion models remains the same.

DragonDiffusion: Enabling Drag-style Manipulation on Diffusion Models

Chong Mou[1], Xintao Wang[2], Jiechong Song[1], Ying Shan[2], Jian Zhang[1✉],
[1]School of Electronic and Computer Engineering, Shenzhen Graduate School, Peking University,
[2]ARC Lab, Tencent PCG

arXiv Github (Improving)

Saturday (9.14) Preview

Homework01 (估计提前在github放出来)

（大概）如何实现作业/文章

作业/课程 答疑

Q & A