CS-565 Computer Vision

Nazar Khan

Department of Computer Science
University of the Punjab

10. Transformations

Homogenous Coordinates

- Vectors that we use normally are in *Cartesian coordinates* and reside in Cartesian space $\mathbb{R}^d$.
- Appending a 1 as the last element of a Cartesian vector yields a vector in *homogenous coordinates*.

$\mathbf{v}$	$\hat{\mathbf{v}}$
$\begin{bmatrix} x \\ y \end{bmatrix}$	$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

- A homogenous vector resides in the so-called *projective space* $\mathbb{P}^d = \mathbb{R}^{d+1} \setminus \mathbf{0}$.
  - Projective space is just Cartesian space with an additional dimension *but* without an origin.
  - Dimensionality of $\mathbb{P}^d$ is $d + 1$.

Projective Space

- $\mathbb{R}^d$ to $\mathbb{P}^d$: Append by 1.
- $\mathbb{P}^d$ to $\mathbb{R}^d$: Divide by last element to make it 1 and then drop it.

$$\hat{\mathbf{v}} = \begin{bmatrix} x \\ y \\ w \end{bmatrix} \longrightarrow \mathbf{v} = \begin{bmatrix} x/w \\ y/w \end{bmatrix}$$

- This means that in projective space, any vector $\mathbf{v}$ and its scaled version $k\mathbf{v}$ will *project down* to the same Cartesian vector.
- That is, $\mathbf{v}$ is *projectively equivalent* to $k\mathbf{v}$. Written as

$$\mathbf{v} \equiv k\mathbf{v} \tag{1}$$

for $k \neq 0$.

Affine Transformation in $\mathbb{P}^2$

- Consider the following linear transformation from $\mathbb{P}^2$ to $\mathbb{P}^2$

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

- Note that the last component will remain unchanged.
- Every affine transformation is invertible.
- Six degrees of freedom (DoF).
- An affine transformation matrix can perform 2D rotation, scaling, shear or translation.
- Any sequence of affine transformations is still affine (look at the last row).

Affine Transformation

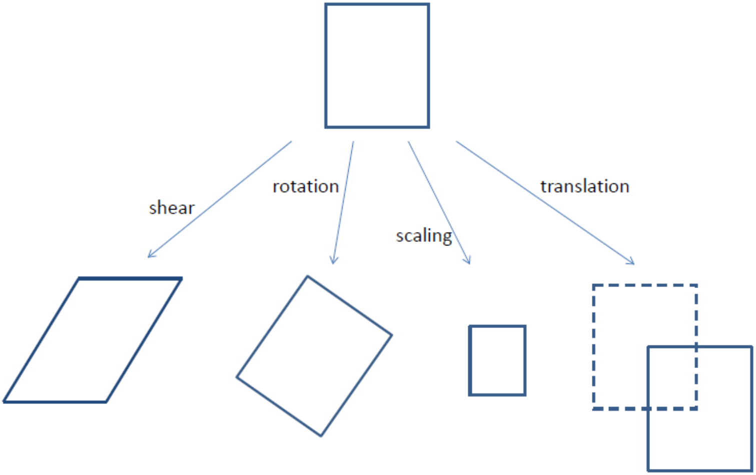


Figure: Capabilities of an affine transformation matrix.

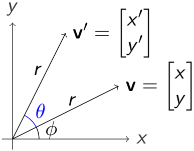
Affine Transformation

Scaling	Shear	Translation	Rotation
$\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & sh_x & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$
$x' = s_x x$	$x' = x + sh_x y$	$x' = x + t_x$	$x' = x\cos\theta - y\sin\theta$
$y' = s_y y$	$y' = y + sh_y x$	$y' = y + t_y$	$y' = x\sin\theta + y\cos\theta$

Note that translation cannot be written in matrix-vector form in Cartesian space.

Rotation Matrix
Derivation

For counter-clockwise rotation of $\mathbf{v}$ *around origin* by θ



$$
\begin{aligned}
x' &= r\cos(\phi + \theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta \\
&= x\cos\theta - y\sin\theta \\
y' &= r\sin(\phi + \theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta \\
&= x\sin\theta + y\cos\theta
\end{aligned}
$$

Therefore

$$
\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2}
$$

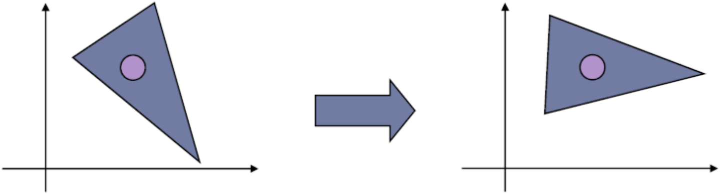
Rotation Matrix
Properties

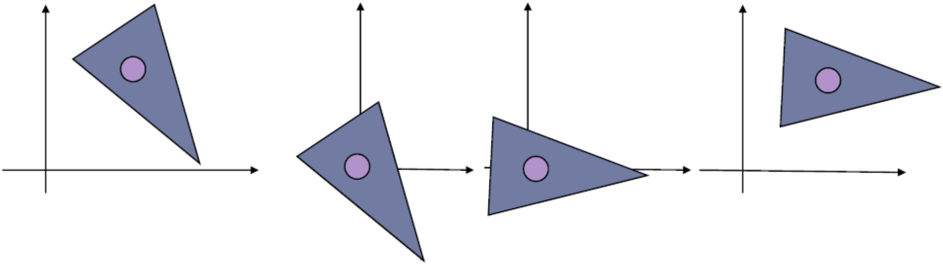
- For any rotation matrix $\mathbf{R}$
  1. Each row is orthogonal to the other. Same for columns.
  2. Each row has unit norm. Same for columns.
- Such matrices are called *orthonormal* matrices.

$$\mathbf{R}^T\mathbf{R} = \mathbf{I}$$

- They preserve length of the vector being transformed.

Rotation around an arbitrary point





Order matters!

Rotation/scaling/shear followed by translation

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & sh_x & 0 \\ sh_y & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & sh_x & t_x \\ sh_y & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

is not the same as translation followed by rotation/scaling/shear.

$$\begin{bmatrix} s_x & sh_x & 0 \\ sh_y & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & sh_x & s_x t_x + sh_x t_y \\ sh_y & s_y & sh_y t_x + s_y t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Projective Transformation

- Last row of affine transformation matrix is always $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$.
- If this condition is relaxed we obtain the so-called *projective transformation*.

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

- Also called *homography* or *collineation* since lines are mapped to lines.

Projective Transformation

- Linear in $\mathbb{P}^2$ but non-linear in $\mathbb{R}^2$ because 3rd coordinate of $\mathbf{v}'$ is not guaranteed to be 1.

$$\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 x + h_2 y + h_3 \\ h_4 x + h_5 y + h_6 \\ h_7 x + h_8 y + h_9 \end{bmatrix} \implies \begin{aligned} x' &= \frac{h_1 x + h_2 y + h_3}{h_7 x + h_8 y + h_9} \\ y' &= \frac{h_4 x + h_5 y + h_6}{h_7 x + h_8 y + h_9} \end{aligned}$$

- The 3rd coordinate is now a function of the inputs x and y and division involving them makes the transformation non-linear.

Projective Transformation

Degrees of Freedom

- Projective transformation has only 8 degrees of freedom.
  - In projective space, $\mathbf{v} \equiv k(\mathbf{v})$ for all $k \neq 0$ because both correspond to the same point in Cartesian space. So

$$k(\mathbf{v}) \equiv \mathbf{v} \implies k(\mathbf{Hv}) \equiv \mathbf{Hv} \implies k\mathbf{Hv} \equiv \mathbf{Hv} \implies k\mathbf{H} \equiv \mathbf{H}$$

  - Let $\mathbf{H}' = \frac{1}{h_9}\mathbf{H}$. Clearly, $h'_9 = 1$ and therefore $\mathbf{H}'$ has 8 free parameters.
  - But since $\mathbf{H}' \equiv \mathbf{H}$, $\mathbf{H}$ must also have only 8 free parameters.

Estimation of Affine Transform

- We are given N corresponding points $\mathbf{x}_1 \Longleftrightarrow \mathbf{x}'_1, \mathbf{x}_2 \Longleftrightarrow \mathbf{x}'_2, \ldots, \mathbf{x}_N \Longleftrightarrow \mathbf{x}'_N$ where $\mathbf{x}'_i = \mathbf{T}\mathbf{x}_i$ represents an affinely transformed point pair.
- Goal is to find the 6 parameters $[a; b; e; c; d; f]$ of the affine transformation $\mathbf{T}$ that maps $\mathbf{x}$ to $\mathbf{x}'$.
- The ith correspondence can be written as

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_i & y_i & 1 \end{bmatrix} \begin{bmatrix} s_x \\ sh_x \\ t_x \\ s_y \\ sh_y \\ t_y \end{bmatrix} = \begin{bmatrix} x'_i \\ y'_i \end{bmatrix}$$

Estimation of Affine Transform

- All N correspondences can be written as

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ & & \vdots & & & \\ x_N & y_N & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_N & y_N & 1 \end{bmatrix}}_{2N \times 6} \underbrace{\begin{bmatrix} s_x \\ sh_x \\ t_x \\ s_y \\ sh_y \\ t_y \end{bmatrix}}_{6 \times 1} = \underbrace{\begin{bmatrix} x'_1 \\ y'_1 \\ \vdots \\ x'_N \\ y'_N \end{bmatrix}}_{2N \times 1}$$

  which can be seen as a linear system $\mathbf{Av} = \mathbf{b}$.
- Can be solved via pseudoinverse

$$\mathbf{Av} = \mathbf{b} \implies \mathbf{A}^T\mathbf{Av} = \mathbf{A}^T\mathbf{b} \implies \mathbf{v} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} = \mathbf{A}^\dagger\mathbf{b}$$

  where $\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ is the $6 \times 2N$ matrix called the *pseudoinverse* of $\mathbf{A}$.

Estimation of Affine Transform
Algorithm

Input: N point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$

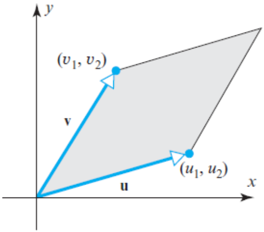
1. Fill in the $2N \times 6$ matrix $\mathbf{A}$ using the $\mathbf{x}_i$.
2. Fill in the $2N \times 1$ vector $\mathbf{b}$ using the $\mathbf{x}'_i$.
3. Compute $6 \times 2N$ pseudo-inverse $\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$.
4. Compute optimal affine transformation parameters as $\mathbf{v}^* = \mathbf{A}^\dagger\mathbf{b}$.

Detour – Cross Product

$$\mathbf{u} \times \mathbf{v} = \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}}_{[\mathbf{u}]_\times} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

- Only defined for 3-dimensional space.
- Matrix $[\mathbf{u}]_\times$ has two linearly independent rows.
  - *Proof*: u_1 row1 $+ u_2$ row2 $+ u_3$ row3 $= \mathbf{0}^T \implies$ any row can be written as a linear combination of the other two rows.
- $\mathbf{u} \times \mathbf{v}$ is another 3-dimensional vector orthogonal to both $\mathbf{u}$ and $\mathbf{v}$.
- $\|\mathbf{u} \times \mathbf{v}\|$ represents the area of the parallelogram formed by $\mathbf{u}$ and $\mathbf{v}$.

Detour – Cross Product



- If **u** and **v** point in the same direction, then no parallelogram will be formed.
- Therefore $\|\mathbf{u} \times \mathbf{v}\|$ will be 0.
- The only vector with norm 0 is the $\mathbf{0}$ vector.
- Therefore, $\mathbf{u} \times \mathbf{v} = \mathbf{0}$ when **u** and **v** point in the same direction.

Estimation of Projective Transform

- We are given N corresponding points

$$\begin{aligned} \mathbf{x}_1 &\leftrightarrow \mathbf{x}'_1 \\ \mathbf{x}_2 &\leftrightarrow \mathbf{x}'_2 \\ &\vdots \\ \mathbf{x}_N &\leftrightarrow \mathbf{x}'_N \end{aligned}$$

  where $\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$ represents a projectively transformed point pair.
- Goal is to find the 8 parameters $h_1, h_2 \ldots, h_8$ of the projective transformation $\mathbf{H}$ that maps the $\mathbf{x}$ points to the $\mathbf{x}'$ points.
- Parameter h_9 can be fixed to be 1.
- The ith correspondence can be written as $\mathbf{x}'_i \equiv \mathbf{H}\mathbf{x}_i$ in projective space.

Estimation of Projective Transform

- This implies that the 3-dimensional vectors $\mathbf{x}'_i$ and $\mathbf{H}\mathbf{x}_i$ point in the same direction. Their cross-product will be the zero vector.

$$
\begin{aligned}
& \mathbf{x}'_i \times \mathbf{H}\mathbf{x}_i = \mathbf{0} \\
\implies & \begin{bmatrix} x'_i \\ y'_i \\ w'_i \end{bmatrix} \times \begin{bmatrix} \mathbf{h}^{1T} \\ \mathbf{h}^{2T} \\ \mathbf{h}^{3T} \end{bmatrix} \mathbf{x}_i = \mathbf{0} \text{ where } \mathbf{h}^{jT} \text{ is the } j\text{-th row of } \mathbf{H} \\
\implies & \begin{bmatrix} 0 & -w'_i & y'_i \\ w'_i & 0 & -x'_i \\ -y'_i & x'_i & 0 \end{bmatrix} \begin{bmatrix} \mathbf{h}^{1T}\mathbf{x}_i \\ \mathbf{h}^{2T}\mathbf{x}_i \\ \mathbf{h}^{3T}\mathbf{x}_i \end{bmatrix} = \mathbf{0} \\
\implies & \begin{bmatrix} y'_i\mathbf{h}^{3T}\mathbf{x}_i - w'_i\mathbf{h}^{2T}\mathbf{x}_i \\ w'_i\mathbf{h}^{1T}\mathbf{x}_i - x'_i\mathbf{h}^{3T}\mathbf{x}_i \\ x'_i\mathbf{h}^{2T}\mathbf{x}_i - y'_i\mathbf{h}^{1T}\mathbf{x}_i \end{bmatrix} = \begin{bmatrix} y'_i\mathbf{x}_i^T\mathbf{h}^3 - w'_i\mathbf{x}_i^T\mathbf{h}^2 \\ w'_i\mathbf{x}_i^T\mathbf{h}^1 - x'_i\mathbf{x}_i^T\mathbf{h}^3 \\ x'_i\mathbf{x}_i^T\mathbf{h}^2 - y'_i\mathbf{x}_i^T\mathbf{h}^1 \end{bmatrix} = \mathbf{0} \\
\implies & \begin{bmatrix} \mathbf{0}^T & -w'_i\mathbf{x}_i^T & y'_i\mathbf{x}_i^T \\ w'_i\mathbf{x}_i^T & \mathbf{0}^T & -x'_i\mathbf{x}_i^T \\ -y'_i\mathbf{x}_i^T & x'_i\mathbf{x}_i^T & \mathbf{0}^T \end{bmatrix}_{3\times 9} \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix}_{9\times 1} = \mathbf{A}_i\mathbf{h} = \mathbf{0}
\end{aligned}
$$

Estimation of Projective Transform

- Matrix $\mathbf{A}_i$ has only 2 linearly independent rows.
- So one row can be discarded. Let's denote the resulting 2×9 matrix by $\mathbf{A}_i$ as well.
- So one correspondence $\mathbf{x}_i \Longleftrightarrow \mathbf{x}'_i$ yields 2 equations.
- Since 8 unknowns require atleast 8 equations, we will need $N \geq 4$ corresponding point pairs.

> The points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ must be non-collinear. Similarly, $\mathbf{x}'_1, \ldots, \mathbf{x}'_N$ must also be non-collinear.

Estimation of Projective Transform

- This will yield the homogenous system $\mathbf{Ah} = \mathbf{0}$ where size of $\mathbf{A}$ is $2N \times 9$.
- It can be shown that $\text{rank}(\mathbf{A}) = 8$ and $\dim(\mathbf{A}) = 9$.
- So nullity of $\mathbf{A}$ is 1 and therefore $\mathbf{h}$ can be found as the null space of $\mathbf{A}$.
- However, when measurements contain noise (which is always the case with pixel locations) or $N > 4$, then no $\mathbf{h}$ will exist that satisfies $\mathbf{Ah} = \mathbf{0}$ exactly.
- In such cases, the best one can do is to find an $\mathbf{h}$ that makes $\mathbf{Ah}$ as close to $\mathbf{0}$ as possible. This can be achieved via

$$\mathbf{h}^* = \arg\min_{\mathbf{h}} \|\mathbf{Ah}\|^2 \text{ s.t. } \|\mathbf{h}\|^2 = 1$$

Take-home Quiz 3: Show that $\mathbf{h}^*$ must be the eigenvector of $\mathbf{A}^T\mathbf{A}$ corresponding to the smallest eigenvalue.

Estimation of Projective Transform

- This can be done via singular value decomposition.

$$[\mathbf{U}, \mathbf{D}, \mathbf{V}] = \mathsf{svd}(\mathbf{A})$$

and $\mathbf{h}$ is the last column of the matrix $\mathbf{V}$.

Estimation of Projective Transform

Algorithm

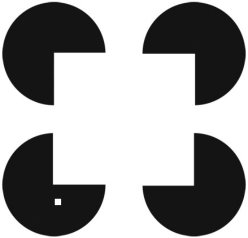
Input: N point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$

1. Fill in the $2N \times 9$ matrix $\mathbf{A}$ using the $\mathbf{x}_i$ and $\mathbf{x}'_i$.
2. Compute $[\mathbf{U}, \mathbf{D}, \mathbf{V}] = \text{svd}(\mathbf{A})$.
3. Optimal projective transformation parameters $\mathbf{h}^*$ are the last column of matrix $\mathbf{V}$.

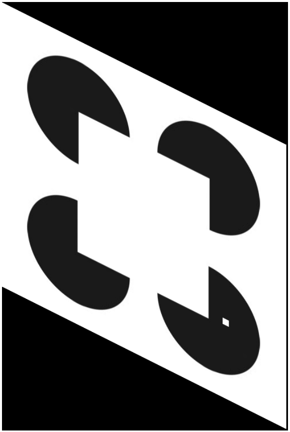
This algorithm is known as the *Direct Linear Transform (DLT)*.[1]

[1] For some practical tips, please refer to slides 14 – 17 from
http://www.ele.puc-rio.br/~visao/Homographies.pdf

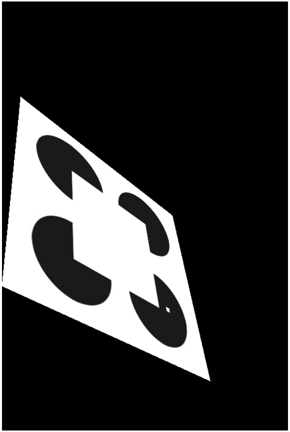
Image Warping



Original



Affine



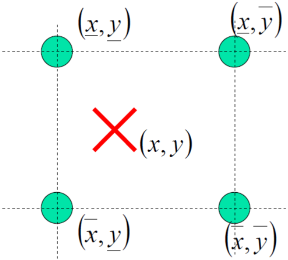
Projective

Image Warping

- Inputs: Image I and transformation matrix $\mathbf{H}$.
- Output: Transformed image $I' = \mathbf{H}I$.
- Obvious approach:
  - For each pixel $\mathbf{x}$ in image I
  - Find transformed point $\mathbf{x}' = \mathbf{H}\mathbf{x}$
  - Divide by 3rd coordinate and move to Cartesian space
  - Copy the pixel color as $I'(\mathbf{x}') = I(\mathbf{x})$.
- Problem: Can leave holes in I'. Why?
- Solution:
  - For each pixel $\mathbf{x}'$ in image I'
  - Find transformed point $\mathbf{x} = \mathbf{H}^{-1}\mathbf{x}'$
  - Divide by 3rd coordinate and move to Cartesian space
  - Copy the pixel color as $I'(\mathbf{x}') = I(\mathbf{x})$.
- Problem: Transformed point $\mathbf{x}$ is not necessarily integer valued.

Image Warping
Bilinear Interpolation

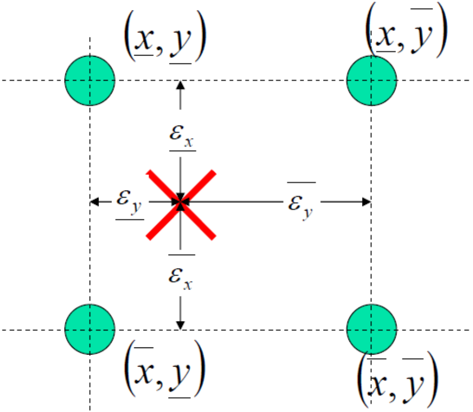
Find 4 nearest pixel locations around (x, y)



where

$$\underline{x} = \lfloor x \rfloor$$
$$\underline{y} = \lfloor y \rfloor$$
$$\bar{x} = \lfloor x \rfloor + 1$$
$$\bar{y} = \lfloor y \rfloor + 1$$

Image Warping
Bilinear Interpolation



$$I(x, y) = \bar{\epsilon}_x \bar{\epsilon}_y I(\underline{x}, \underline{y}) + \underline{\epsilon}_x \bar{\epsilon}_y I(\bar{x}, \underline{y}) + \bar{\epsilon}_x \underline{\epsilon}_y I(\underline{x}, \bar{y}) + \underline{\epsilon}_x \underline{\epsilon}_y I(\bar{x}, \bar{y})$$