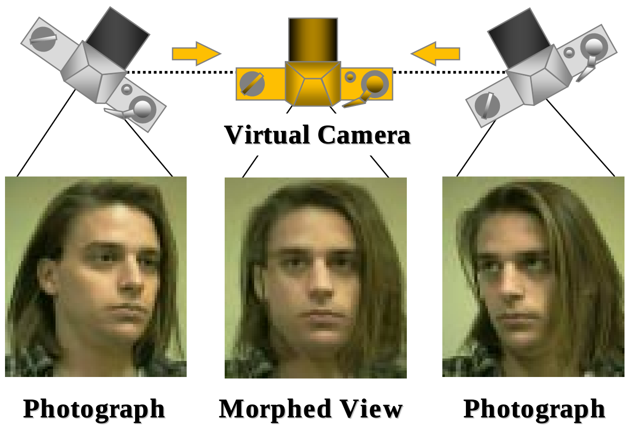
View Morphing (Seitz & Dyer, SIGGRAPH 96)



View interpolation (ala McMillan) but

- no depth
- no camera information

But First: Multi-View Projective Geometry

Last time (single view geometry)

- Vanishing Points
- Points at Infinity
- Vanishing Lines
- The Cross-Ratio

Today (multi-view geometry)

- Point-line duality
- Epipolar geometry
- The Fundamental Matrix

All quantities on these slides are in *homogeneous coordinates* except when specified otherwise

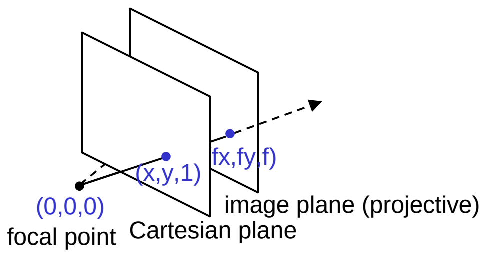
The Projective Plane

Why do we need homogeneous coordinates?

- represent points at infinity, homographies, perspective projection, multi-view relationships

What is the geometric intuition?

- The *projective plane*

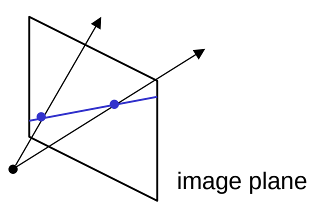


- Each *point* (x,y) on the plane is represented by a *ray* s(x,y,1)
  - Cartesian coordinates $(x,y,z) \rightarrow (x/z, y/z)$

Projective Lines

A point is a ray in projective space

- How would we represent a line?



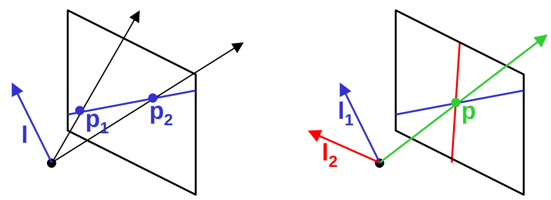
- A line is a *plane* of rays
  - all rays (x,y,z) satisfying: ax + by + cz = 0

$$\text{in vector notation:} \quad 0 = \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{l}^T \mathbf{p}$$

- A line is also represented as a homogeneous 3-vector **l**

Point and Line Duality

- A line $\mathbf{l}$ is a homogeneous 3-vector (a ray)
- It is $\perp$ to every point (ray) $\mathbf{p}$ on the line: $\mathbf{l}^T\mathbf{p}=0$



- What is the line $\mathbf{l}$ spanned by points $\mathbf{p_1}$ and $\mathbf{p_2}$?
  - $\mathbf{l}$ is $\perp$ to $\mathbf{p_1}$ and $\mathbf{p_2}$ $\Rightarrow$ $\mathbf{l} = \mathbf{p_1} \times \mathbf{p_2}$
  - $\mathbf{l}$ is the plane normal
- What is the intersection of two lines $\mathbf{l_1}$ and $\mathbf{l_2}$?
  - $\mathbf{p}$ is $\perp$ to $\mathbf{l_1}$ and $\mathbf{l_2}$ $\Rightarrow$ $\mathbf{p} = \mathbf{l_1} \times \mathbf{l_2}$
- Points and lines are *dual* in projective space
  - every property of points also applies to lines (e.g., cross-ratio)

Homographies of Points and Lines

We've seen lots of names for these

- Planar perspective transformations
- Homographies
- Texture-mapping transformations
- Collineations

Computed by 3x3 matrix multiplication

- To transform a point: $\mathbf{p'} = \mathbf{Hp}$
- To transform a line: $\mathbf{l}^T\mathbf{p}=0 \rightarrow \mathbf{l'}^T\mathbf{p'}=0$
  - $0 = \mathbf{l}^T\mathbf{p} = \mathbf{l}^T\mathbf{H}^{-1}\mathbf{Hp} = \mathbf{l}^T\mathbf{H}^{-1}\mathbf{p'} \Rightarrow \mathbf{l'}^T = \mathbf{l}^T\mathbf{H}^{-1}$
  - lines are transformed by $(\mathbf{H}^{-1})^T$

3D Projective Geometry

These concepts generalize naturally to 3D

- Homogeneous coordinates
  - Projective 3D points have four coords: **X** = (X,Y,Z,W)
- Duality
  - A plane Π is also represented by a 4-vector
  - Points and planes are dual in 3D: $\Pi^T \mathbf{P}=0$
- Projective transformations
  - Represented by 4x4 matrices **T**: $\mathbf{P}' = \mathbf{TP}$, $\Pi' = (\mathbf{T}^{-1})^T \Pi$
- Cross-ratio of planes

However

- Can't use cross-products in 4D. We need new tools
  - Grassman-Cayley Algebra
    - » generalization of cross product, allows interactions between points, lines, and planes via "meet" and "join" operators
  - Won't get into this stuff today

3D to 2D: Perspective Projection

Matrix Projection: $\mathbf{p} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi P}$

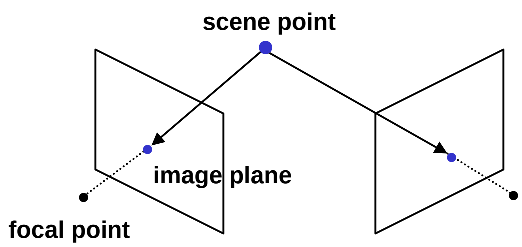
It's useful to decompose $\prod$ into $\mathbf{T} \rightarrow \mathbf{R} \rightarrow$ project $\rightarrow \mathbf{A}$

$$\mathbf{\Pi} = \begin{bmatrix} s_x & 0 & -t_x \\ 0 & s_y & -t_y \\ 0 & 0 & 1/f \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}$$

Then we can write the projection as:

$$\mathbf{p} = \mathbf{\Pi P} = \mathbf{AR(P+T)}$$

Multi-View Projective Geometry



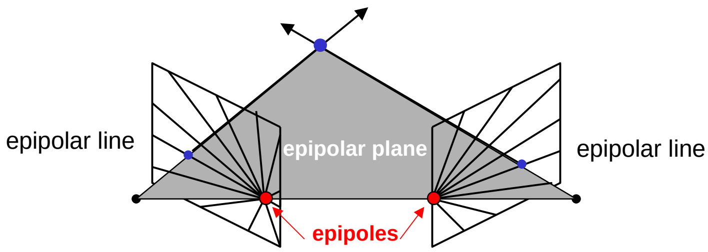
How to relate point positions in different views?

- Central question in image-based rendering
- Projective geometry gives us some powerful tools
  - constraints between two or more images
  - equations to transfer points from one image to another

Epipolar Geometry

What does one view tell us about another?

- Point positions in 2nd view must lie along a known line



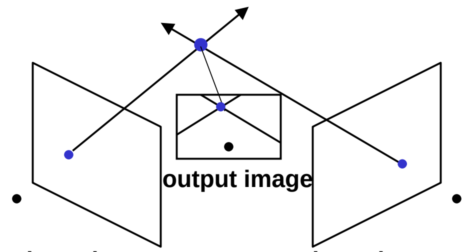
Epipolar Constraint

- Extremely useful for stereo matching
  - Reduces problem to 1D search along *conjugate epipolar lines*
- Also useful for view interpolation...

Transfer from Epipolar Lines

What does one view tell us about another?

- Point positions in 2nd view must lie along a known line



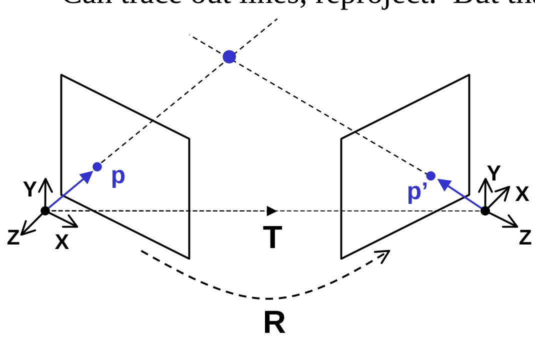
Two views determines point position in a third image

- But doesn't work if point is in the *trifocal plane* spanned by all three cameras
  - bad case: three cameras are colinear

Epipolar Algebra

How do we compute epipolar lines?

- Can trace out lines, reproject. But that is overkill



$\mathbf{p'} = \mathbf{Rp} + \mathbf{T}$

Note that $\mathbf{p'}$ is $\perp$ to $\mathbf{T}\times\mathbf{p'}$

- So $0 = \mathbf{p'}^T\,\mathbf{T}\times\mathbf{p} = \mathbf{p'}^T\,\mathbf{T}\times(\mathbf{Rp} + \mathbf{T}) = \mathbf{p'}^T\,\mathbf{T}\times(\mathbf{Rp})$

Simplifying: $\mathbf{p'}^T \mathbf{T}\times(\mathbf{Rp}) = 0$

We can write a cross-product a×b as a matrix equation

- $\mathbf{a} \times \mathbf{b} = \mathbf{A}_\times \mathbf{b}$ **where** $\begin{bmatrix} x \\ y \\ z \end{bmatrix}_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ y & x & 0 \end{bmatrix}$

Therefore: $0 = \mathbf{p'}^T \mathbf{E}\mathbf{p}$

- Where $\mathbf{E} = \mathbf{T}_\times\mathbf{R}$ is the 3x3 "essential matrix"
- Holds whenever **p** and **p'** correspond to the same scene point

Properties of E

- **Ep** is the epipolar line of **p**; $\mathbf{p'}^T \mathbf{E}$ is the epipolar line of **p'**
  - $\mathbf{p'}^T \mathbf{E}\,\mathbf{p} = 0$ for every pair of corresponding points
  - $\mathbf{0} = \mathbf{Ee} = \mathbf{e'}^T \mathbf{E}$ where **e** and **e'** are the epipoles
    - » **E** has rank < 3, has 5 independent parameters
- **E** tells us *everything* about the epipolar geometry

Linear Multiview Relations

The Essential Matrix: $0 = \mathbf{p'}^T \mathbf{E}\,\mathbf{p}$

- First derived by Longuet-Higgins, Nature 1981
  - also showed how to compute camera **R** and **T** matrices from **E**
  - **E** has only 5 free parameters (three rotation angles, two transl. directions)
- Only applies when cameras have same internal parameters
  - same focal length, aspect ratio, and image center

The Fundamental Matrix: $0 = \mathbf{p'}^T \mathbf{F}\,\mathbf{p}$

- $\mathbf{F} = (\mathbf{A'}^{-1})^T \mathbf{E}\,\mathbf{A}^{-1}$, where $\mathbf{A}_{3x3}$ and $\mathbf{A'}_{3x3}$ contain the internal parameters
- Gives epipoles, epipolar lines
- **F** (like **E**) is defined only up to a scale factor and has rank 2 (7 free params
  - Generalization of the essential matrix
  - Can't uniquely solve for **R** and **T** (or **A** and **A'**) from **F**
  - Can be computed using linear methods
    - » R. Hartley, *In Defence of the 8-point Algorithm*, ICCV 95
  - Or nonlinear methods
    - » Xu & Zhang, *Epipolar Geometry in Stereo, Motion and Object Recognition*, 1996

The Trifocal Tensor

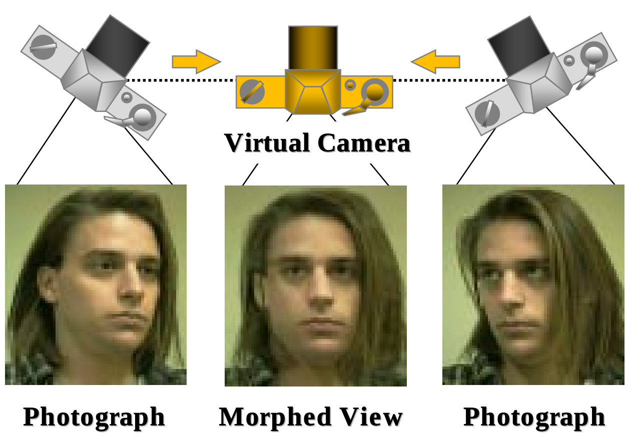
What if you have three views?

- Can compute 3 pairwise fundamental matrices
- However there are more constraints
  - it should be possible to resolve the trifocal problem
- Answer: the *trifocal tensor*
  - introduced by Shashua, Hartley in 1994/1995
  - a 3x3x3 matrix T (27 parameters)
    - » gives all constraints between 3 views
    - » can use to generate new views without trifocal probs. [Shai & Avidan]
    - » linearly computable from point correspondences

How about four views? five views? N views?

- There is a quadrifocal tensor [Faugeras & Morrain, Triggs, 1995]
- But: all the constraints are expressed in the trifocal tensors, obtained by considering every subset of 3 cameras

View Morphing (Seitz & Dyer, SIGGRAPH 96)



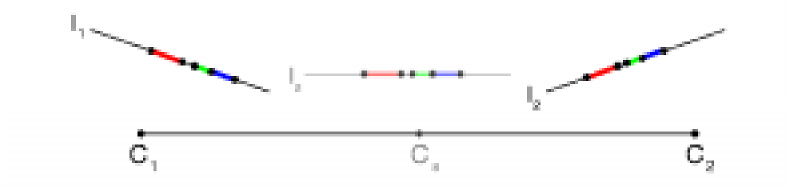
View interpolation (ala McMillan) but

- no depth
- no camera information

Uniqueness Result

Given

- Any two images of a Lambertian scene
- No occlusions

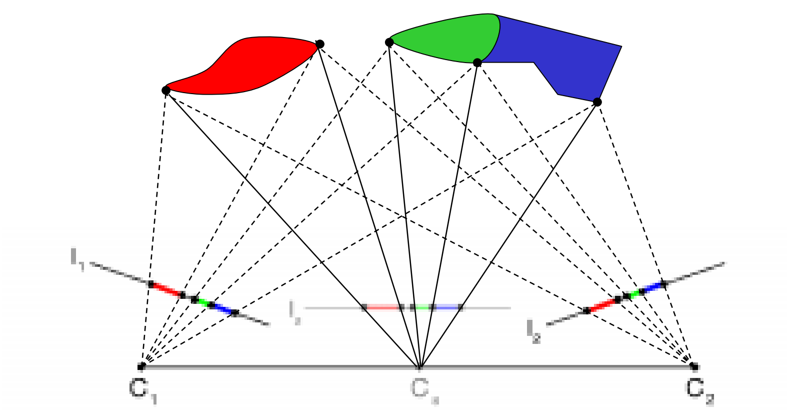


Result: *all views along C_1C_2 are uniquely determined*

View Synthesis is solvable when

- Cameras are uncalibrated
- Dense pixel correspondence is not available
- Shape reconstruction is impossible

Uniqueness Result



Relies on *Monotonicity* Assumption

- Left-to-right ordering of points is the same in both images
  - used often to facilitate stereo matching
- Implies no occlusions on line between C_1 and C_2

Image Morphing

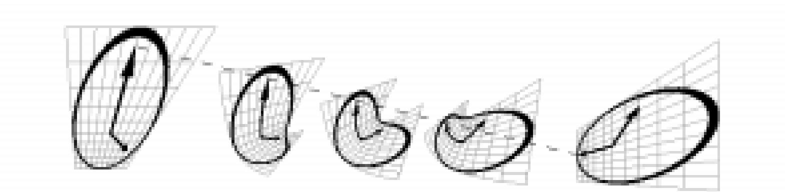
Photograph **Morphed Image** **Photograph**

Linear Interpolation of 2D shape and color

Image Morphing for View Synthesis?

We want to high quality view interpolations

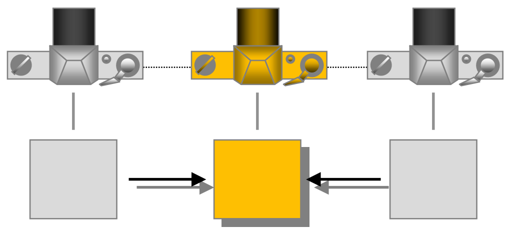
- Can image morphing do this?



Goal: extend to handle changes in viewpoint

- Produce valid camera transitions

Special Case: Parallel Cameras



Morphing parallel views ⇒ new parallel views

- Projection matrices have a special form
  - third rows of projection matrices are equal
- Linear image motion ⇔ linear camera motion

Uncalibrated Prewarping

Parallel cameras have a special epipolar geometry

- Epipolar lines are horizontal
- Corresponding points have the same y coordinate in both images

What fundamental matrix does this correspond to?

$$\hat{\mathbf{F}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

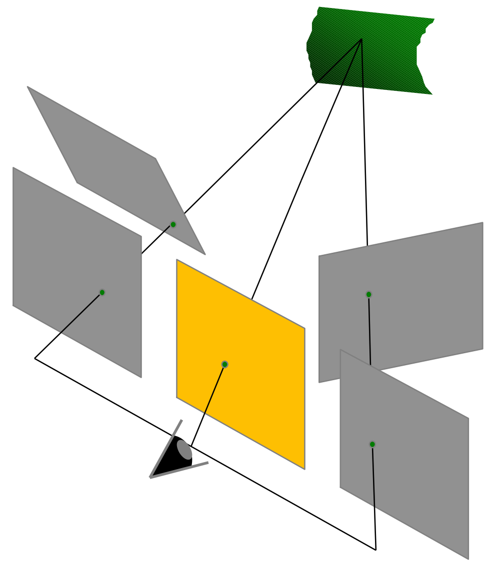
Prewarp procedure:

- Compute F matrix given 8 or more correspondences
- Compute homographies **H** and **H’** such that

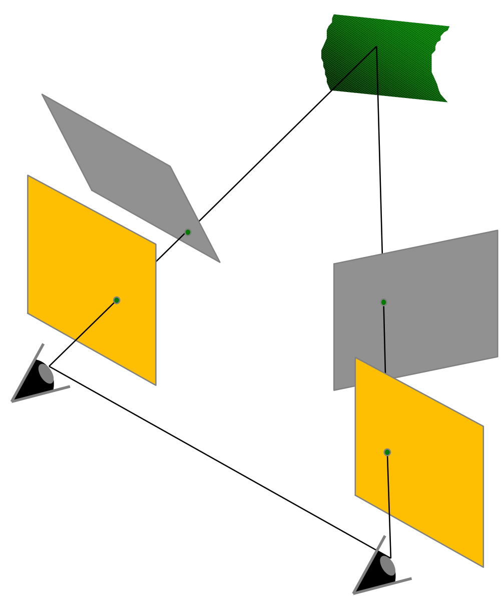
$$\mathbf{H'}^T \mathbf{F}\mathbf{H} = \hat{\mathbf{F}}$$

  - each homography composes two rotations, a scale, and a translation
- Transform first image by $\mathbf{H}^{-1}$, second image by $\mathbf{H'}^{-1}$

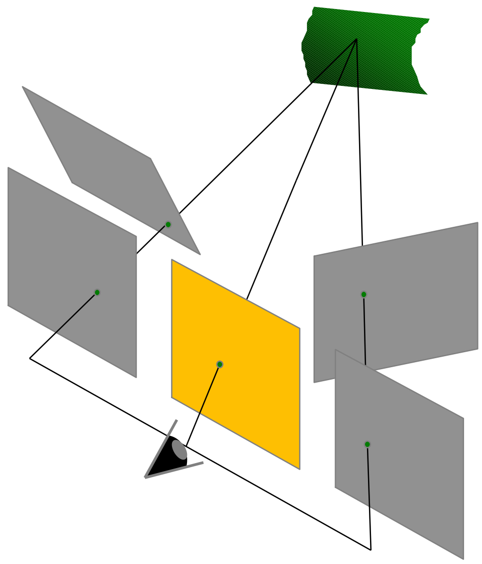
1. Prewarp
   - *align views*
2. Morph
   - *move camera*



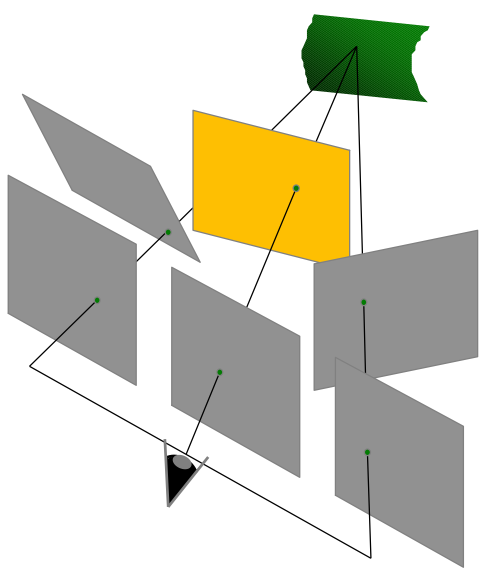
1. Prewarp
   - *align views*



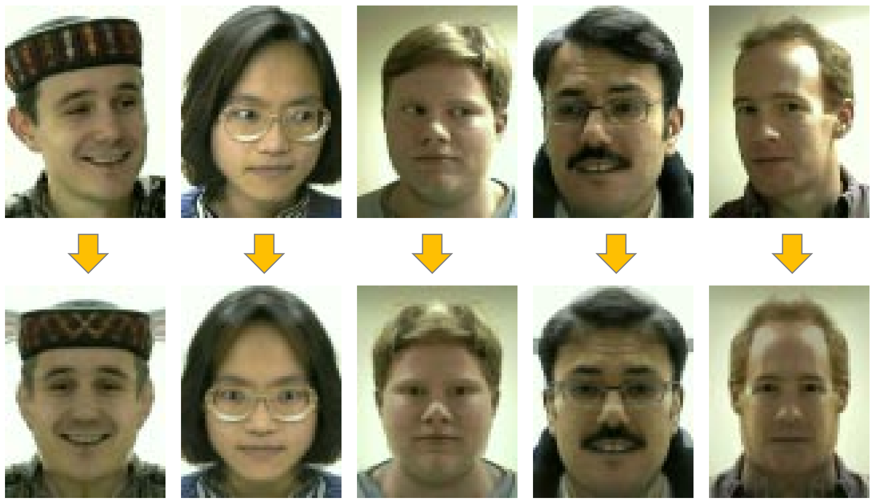
1. Prewarp
   - *align views*
2. Morph
   - *move camera*



1. Prewarp
   - *align views*
2. Morph
   - *move camera*
3. Postwarp
   - *point camera*



Face Recognition



Corrected Photographs

View Morphing Summary

Additional Features

- Automatic correspondence
- N-view morphing
- Real-time implementation

Pros

- Don’t need camera positions
- Don’t need shape/Z
- Don’t need dense correspondence

Cons

- Limited to *interpolation* (although not with three views)
- Problems with occlusions (leads to *ghosting*)
- Prewarp can be unstable (need good F-estimator)

Some Applications of Projective Geometry

Homogeneous coordinates in computer graphics

Metrology

- Single view
- Multi-view

Stereo correspondence

View interpolation and transformation

Camera calibration

Invariants

- Object recognition
- Pose estimation

Others...