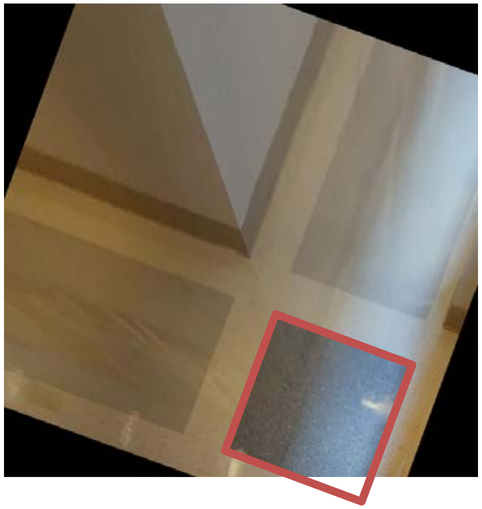
Hierarchy of Transformations



Euclidean (3 dof)

- Length
- Angle
- Area

$$\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ & & 1 \end{bmatrix}$$

Similarity (4 dof)

- Length ratio
- Angle

$$\begin{bmatrix} \alpha\cos\theta & -\alpha\sin\theta & t_x \\ \alpha\sin\theta & \alpha\cos\theta & t_y \\ & & 1 \end{bmatrix}$$

Affine (6 dof)

- Parallelism
- Ratio of area
- Ratio of length

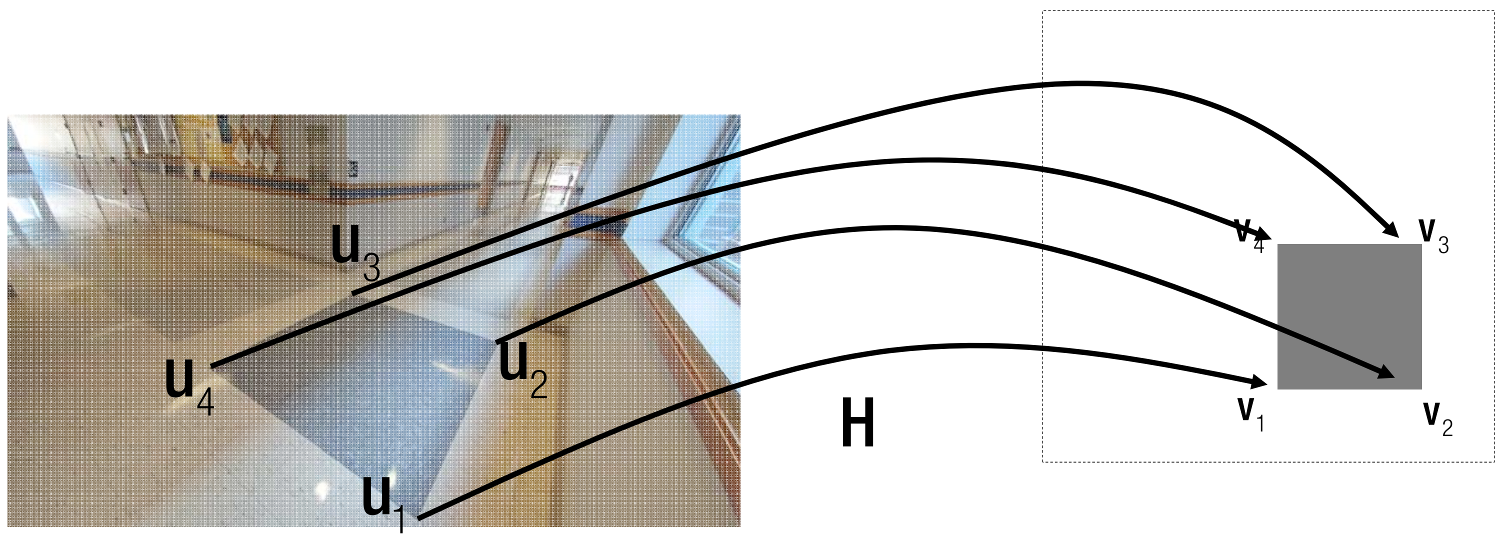
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

Projective (8 dof)

- Cross ratio
- Concurrency
- Colinearity

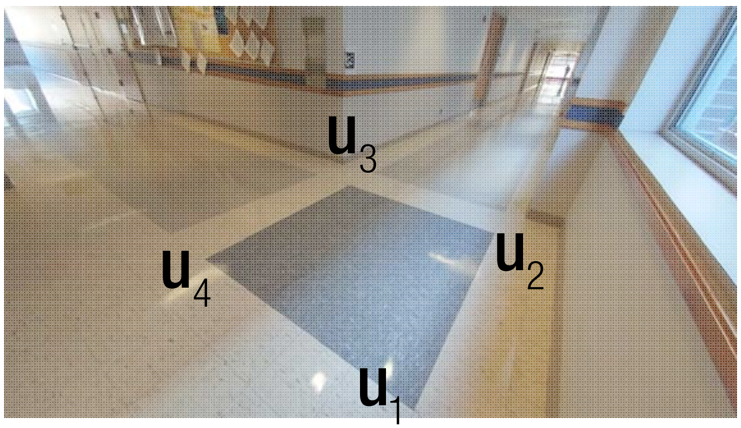
$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$$

Fun with Homography



The image can be rectified as if it is seen from top view.

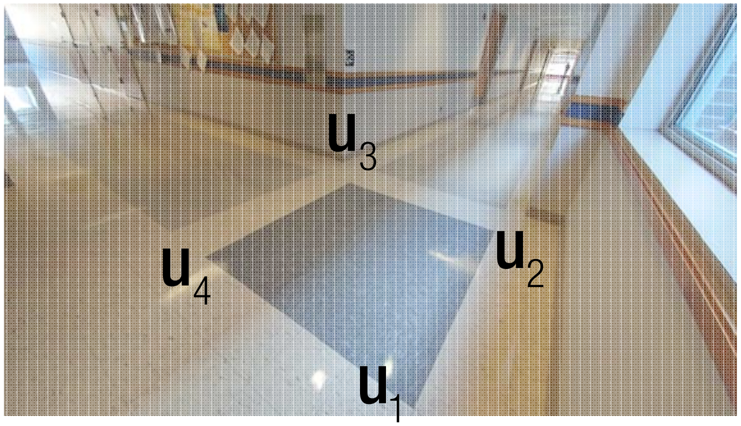
Fun with Homography



RectificationViaHomography.m

```
u = [u1'; u2'; u3'; u4'];
v = [v1'; v2'; v3'; v4'];

% Need at least non-colinear four points
H = ComputeHomography(v, u);

im_warped = ImageWarping(im, H);
```

Fun with Homography



Cf) ImageWarpingEuclidean.m

```
u_x = H(1,1)*v_x + H(1,2)*v_y + H(1,3);
u_y = H(2,1)*v_x + H(2,2)*v_y + H(2,3);
```

RectificationViaHomography.m

```
u = [u1'; u2'; u3'; u4'];
v = [v1'; v2'; v3'; v4'];

% Need at least non-colinear four points
H = ComputeHomography(v, u);

im_warped = ImageWarping(im, H);
```

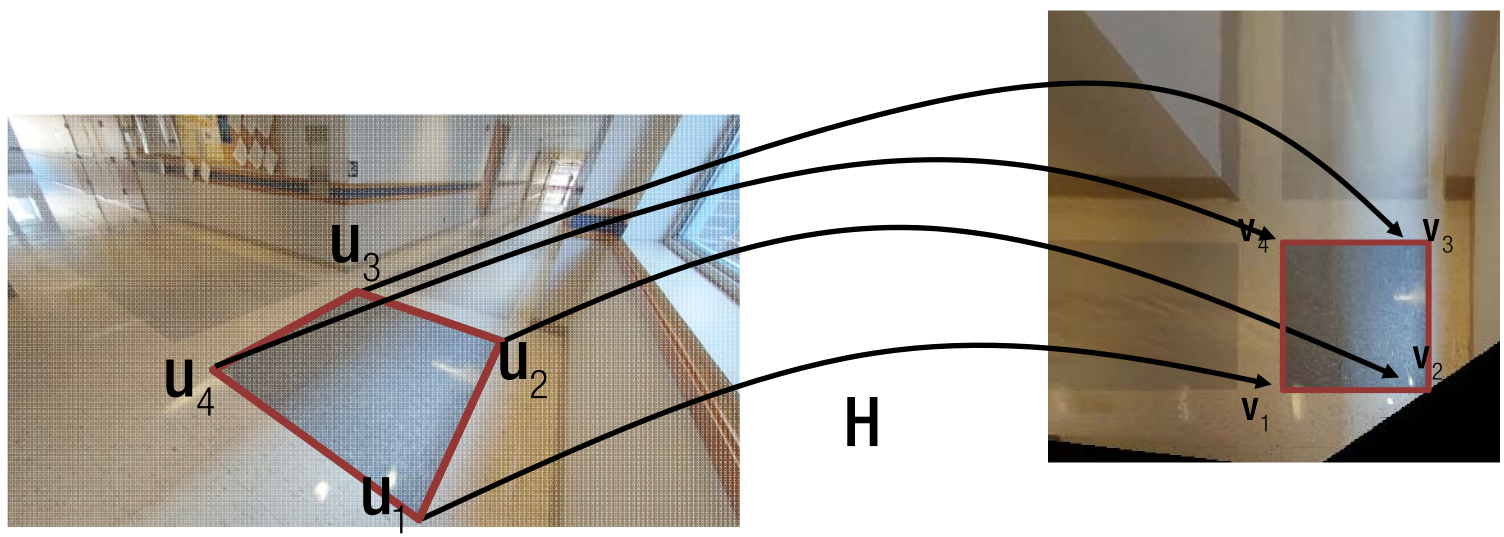
ImageWarping.m

```
u_x = H(1,1)*v_x + H(1,2)*v_y + H(1,3);
u_y = H(2,1)*v_x + H(2,2)*v_y + H(2,3);
u_z = H(3,1)*v_x + H(3,2)*v_y + H(3,3);

u_x = u_x./u_z;
u_y = u_y./u_z;
```

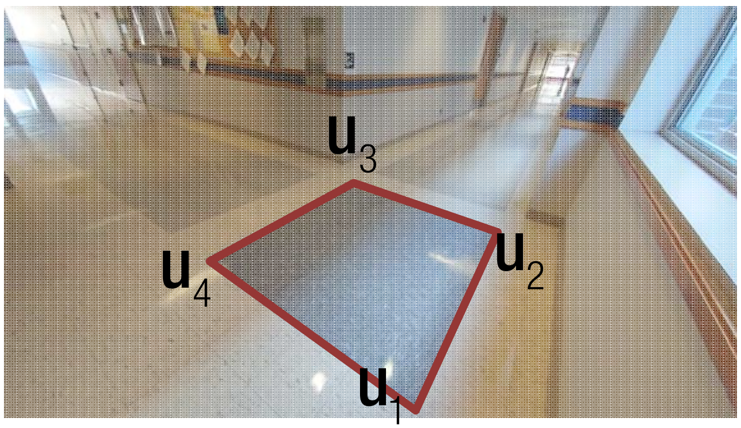
$$\lambda \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

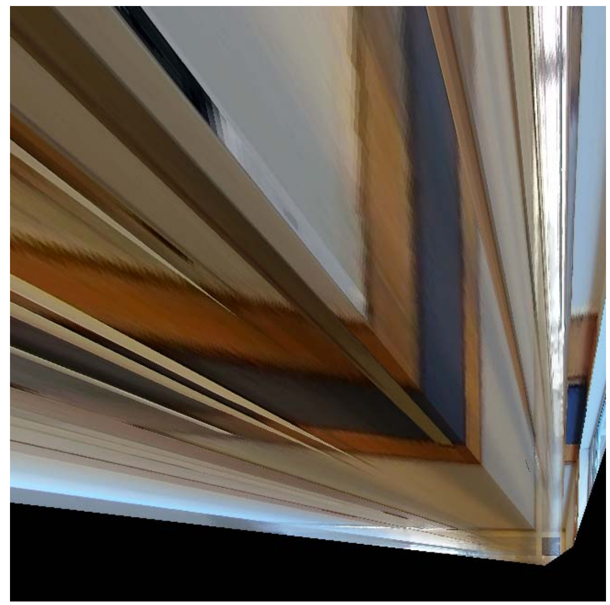
```
im_warped(:,:,1) = reshape(interp2(im(:,:,1), u_x(:), u_y(:)), [h, w]);
im_warped(:,:,2) = reshape(interp2(im(:,:,2), u_x(:), u_y(:)), [h, w]);
im_warped(:,:,3) = reshape(interp2(im(:,:,3), u_x(:), u_y(:)), [h, w]);

im_warped = uint8(im_warped);
```

Fun with Homography

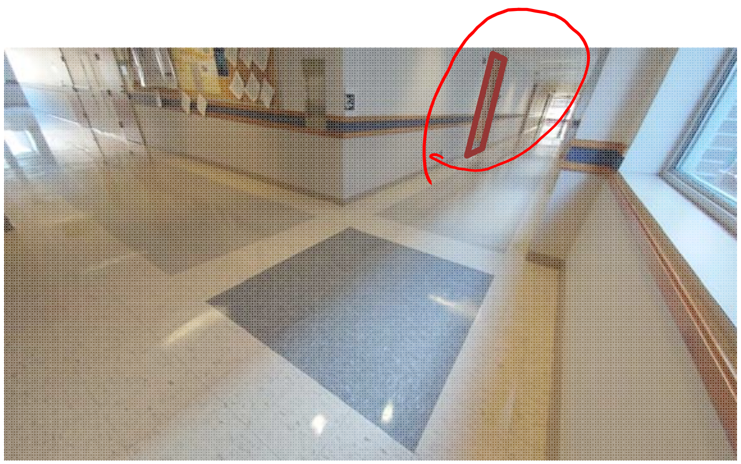


Fun with Homography

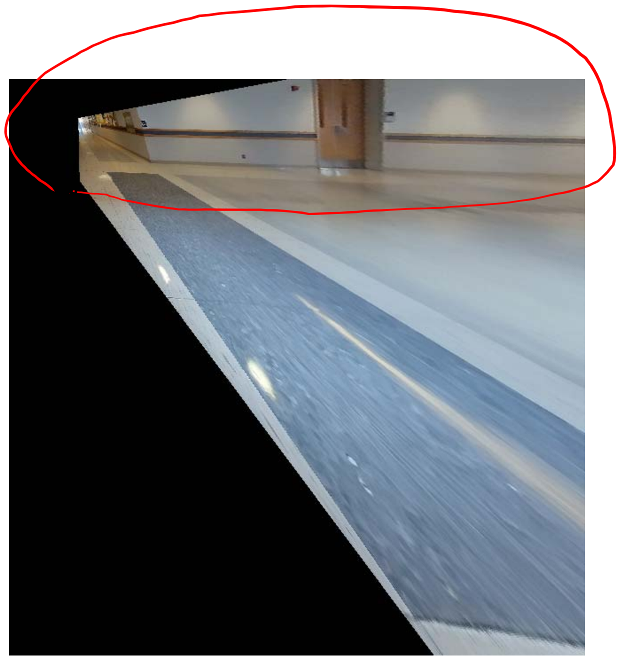




Fun with Homography



Fun with Homography



Morphing = Object Averaging

“an average” between two objects

Not an average of two *images of objects*…

…but an image of the *average object*!

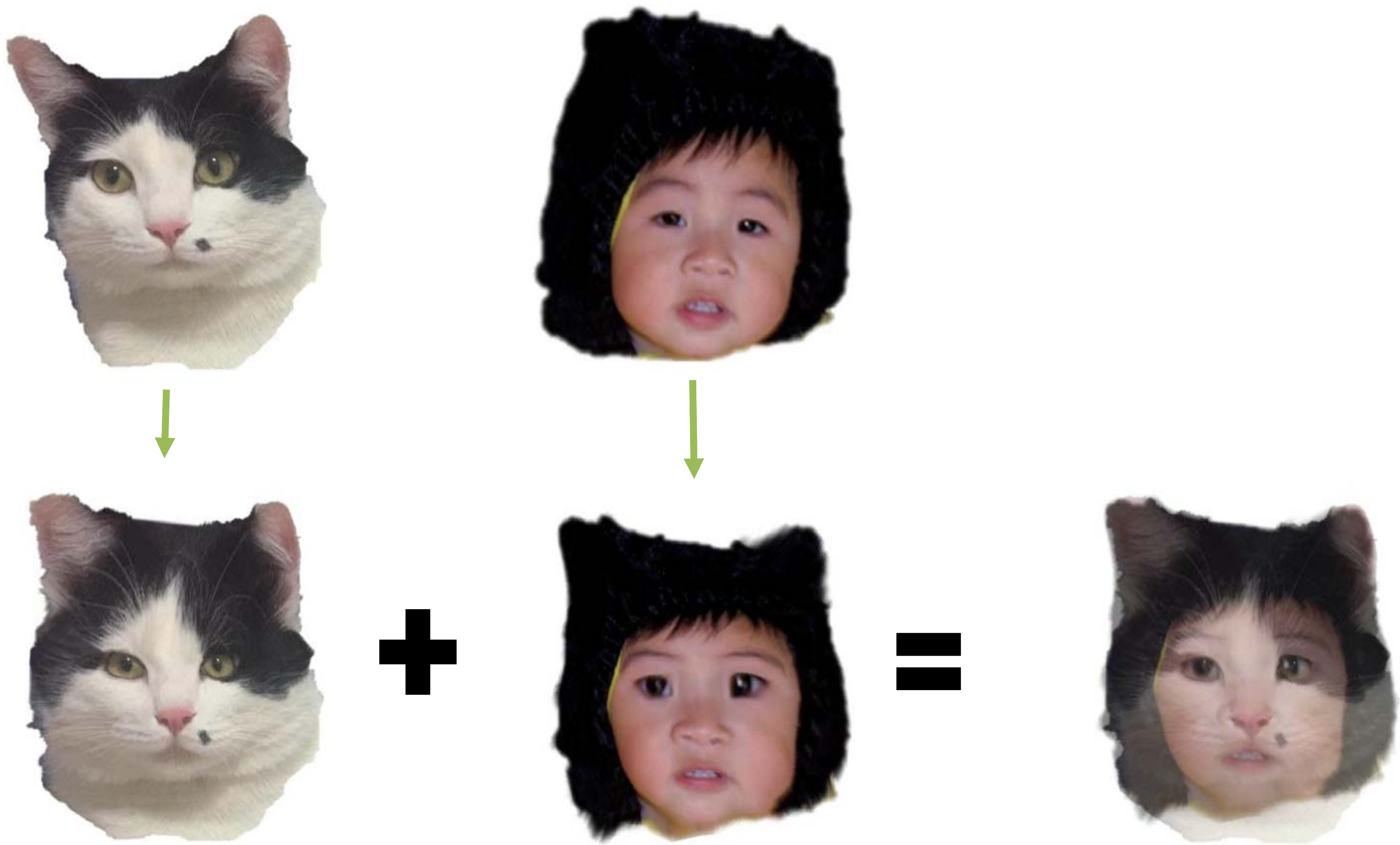
Morphing = Object Averaging

How do we know what the average object looks like?

– We haven’t a clue!

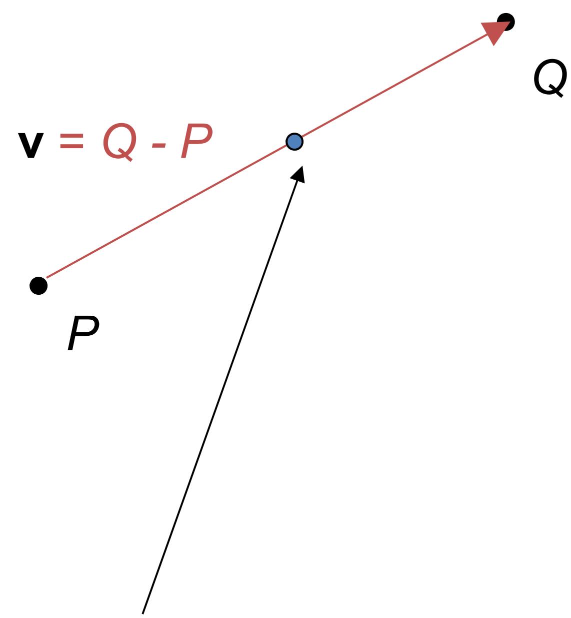
– But we can often fake something reasonable

Morphing = Warping + Cross Dissolving



Averaging Points

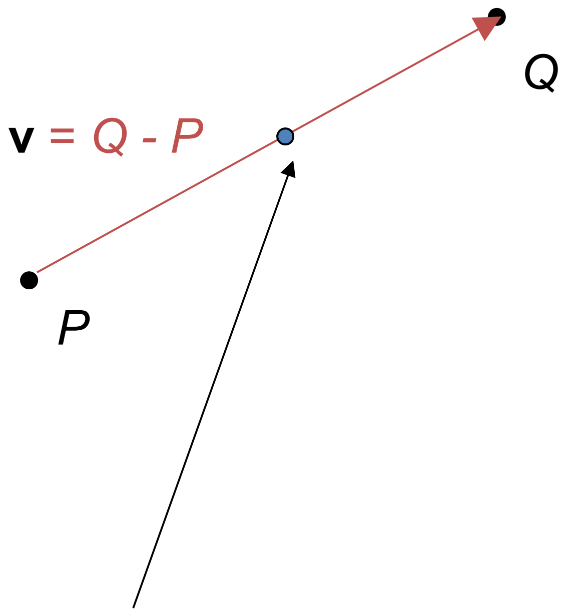
What's the average of P and Q?



$P + 0.5\mathrm{v}$
$= \ P + 0.5(Q - P)$
$= \ 0.5P + 0.5\,Q$

Averaging Points

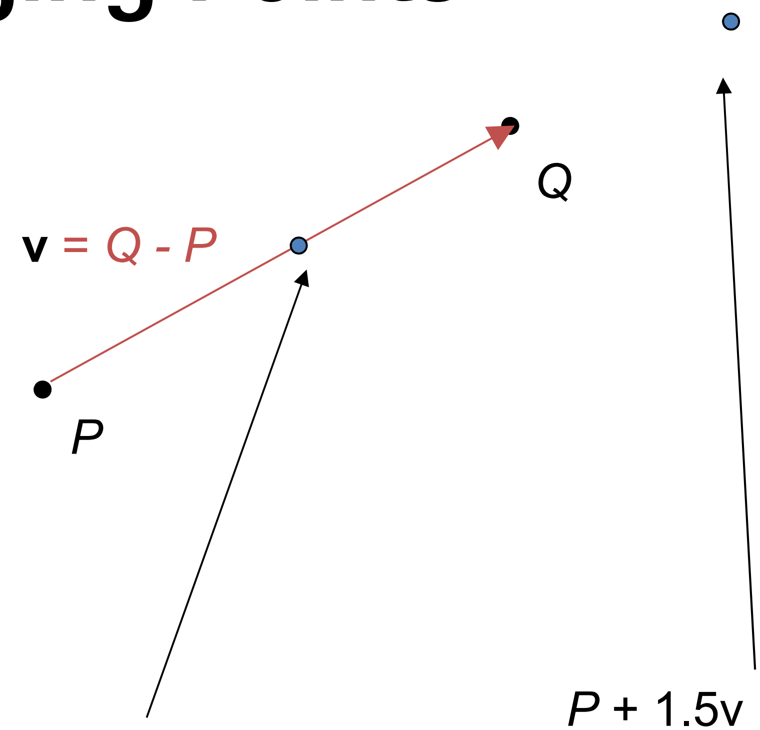
What' s the average of P and Q?



P + 0.5v
= P + 0.5(Q – P)
= 0.5P + 0.5 Q

Linear Interpolation
(Affine Combination):
New point $aP + bQ$,
defined only when $a+b = 1$
So $aP+bQ = aP+(1-a)Q$

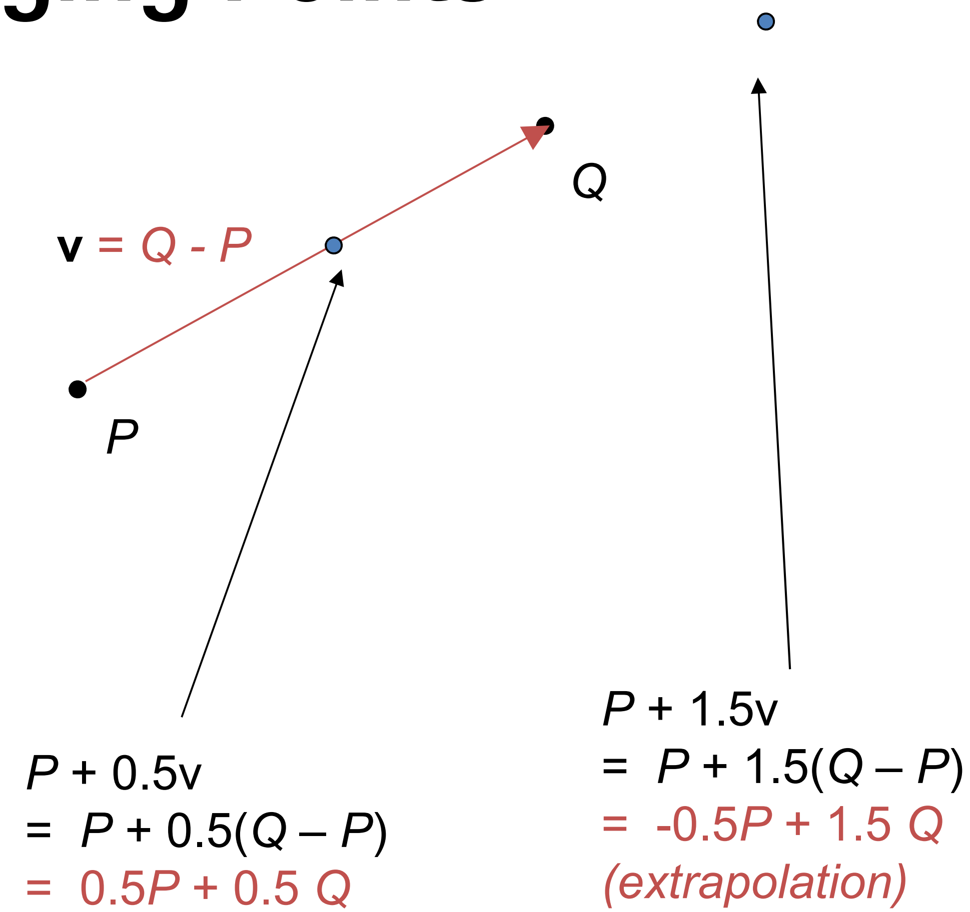
Averaging Points



$P + 0.5\mathrm{v}$
$= \; P + 0.5(Q - P)$
$= \; 0.5P + 0.5\,Q$

$P + 1.5\mathrm{v}$
$= \; P + 1.5(Q - P)$
$= \; -0.5P + 1.5\,Q$
(extrapolation)

Averaging Points



- P and Q can be anything:
  - points on a plane (2D) or in space (3D)
  - Colors in RGB or HSV (3D)
  - Whole images (m-by-n D)… etc.

Averaging Images: Cross-Dissolve

Interpolate whole images:

$$\text{Image}_{\text{halfway}} = (1\text{-}t)^*\text{Image}_1 + t^*\text{image}_2$$

This is called **cross-dissolve** in film industry

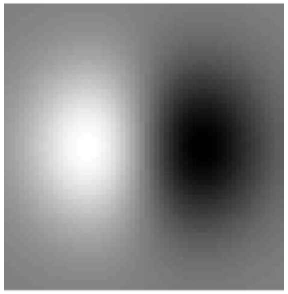
Averaging Images: Cross-Dissolve

Interpolate whole images:

$$\text{Image}_{\text{halfway}} = (1\text{-}t)*\text{Image}_1 + t*\text{image}_2$$

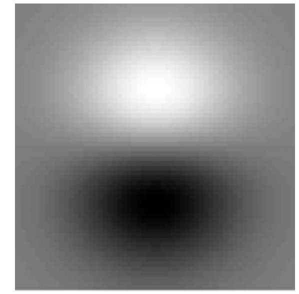
This is called **cross-dissolve** in film industry

Averaging Images



Image$_1$

Averaging Images = Rotating Objects

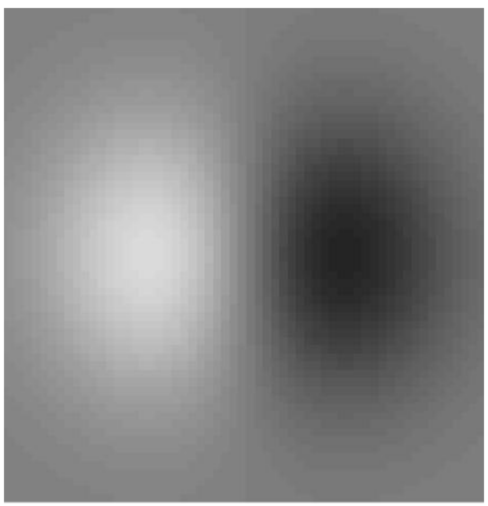


Image$_2$

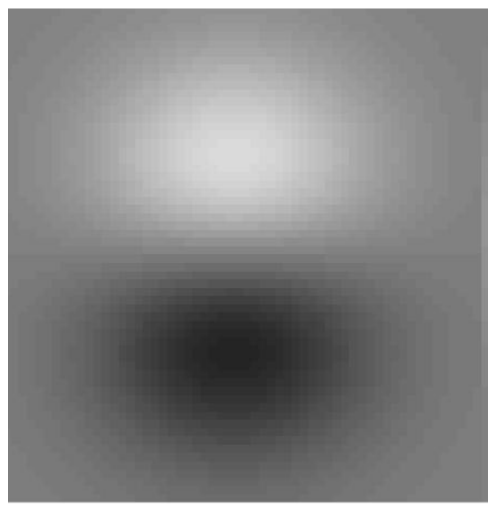
t = 0

t = 0.5

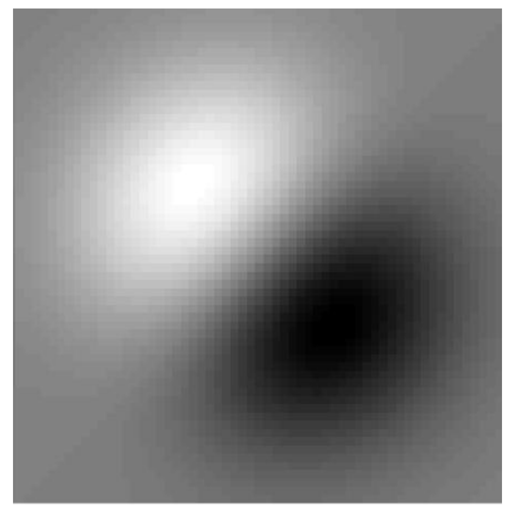
t = 1



+



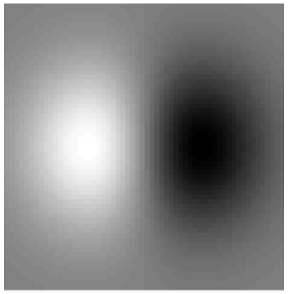
=



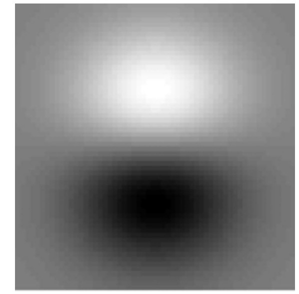
(1-t)*Image$_1$ + t*Image$_2$ = Image$_{halfway}$

Averaging Images

Averaging Images = Rotating Objects

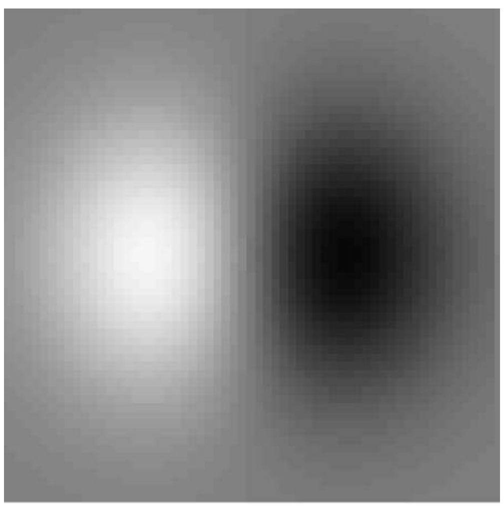


Image$_1$

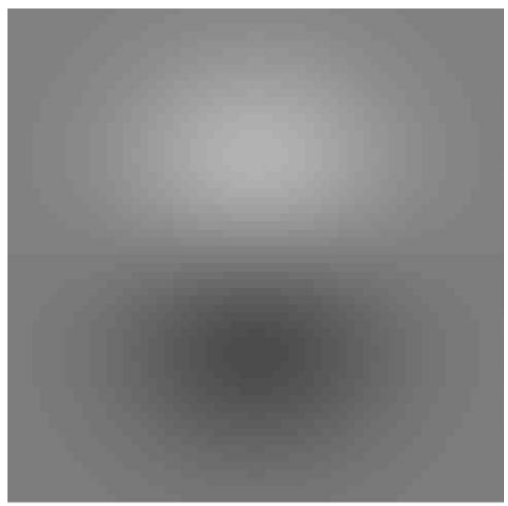


Image$_2$

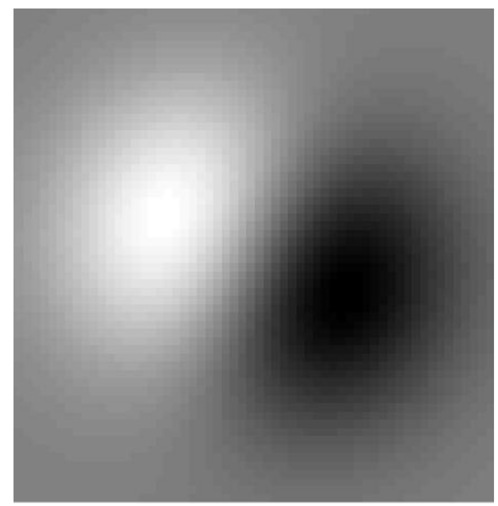
t = 0   t = 0.3   t = 1



+

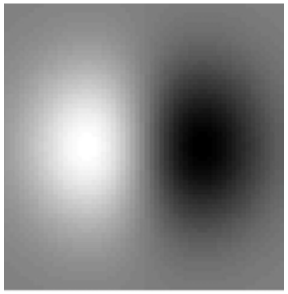


=



(1-t)*Image$_1$ + t*Image$_2$ = Image$_{halfway}$

Averaging Images



Averaging Images = Rotating Objects

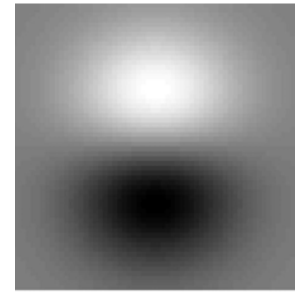
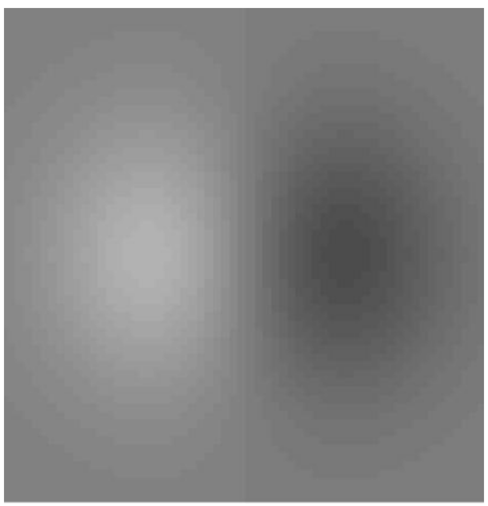
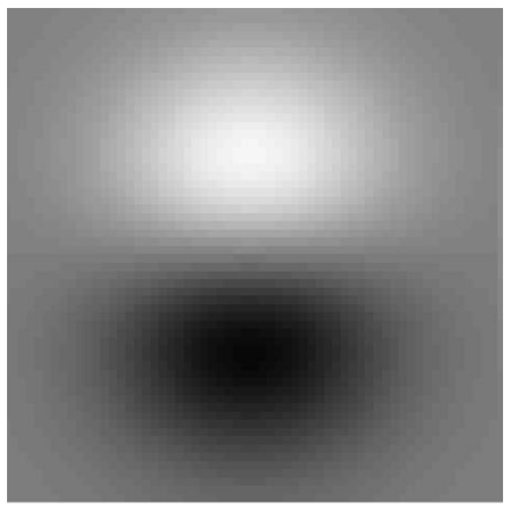


Image_1 Image_2

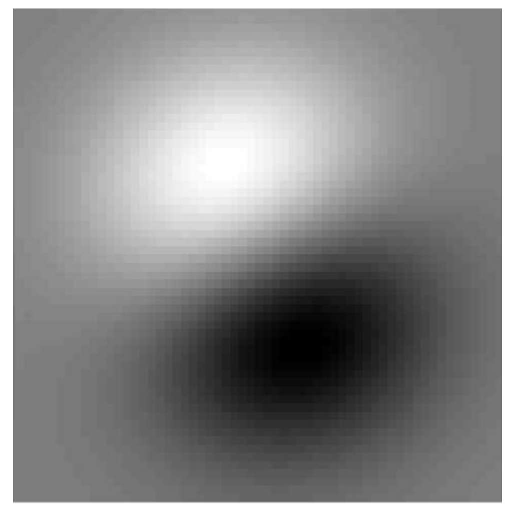
$t = 0$ $t = 0.7$ $t = 1$



+



=



$(1-t)*\text{Image}_1 + t*\text{Image}_2 = \text{Image}_{\text{halfway}}$

Averaging Images

Image$_1$ Image$_2$

+

= ?

Averaging Images

Image_1 + Image_2 ? =

$\text{Image}_1 + \text{Image}_2 \stackrel{?}{=}$

Averaging Images

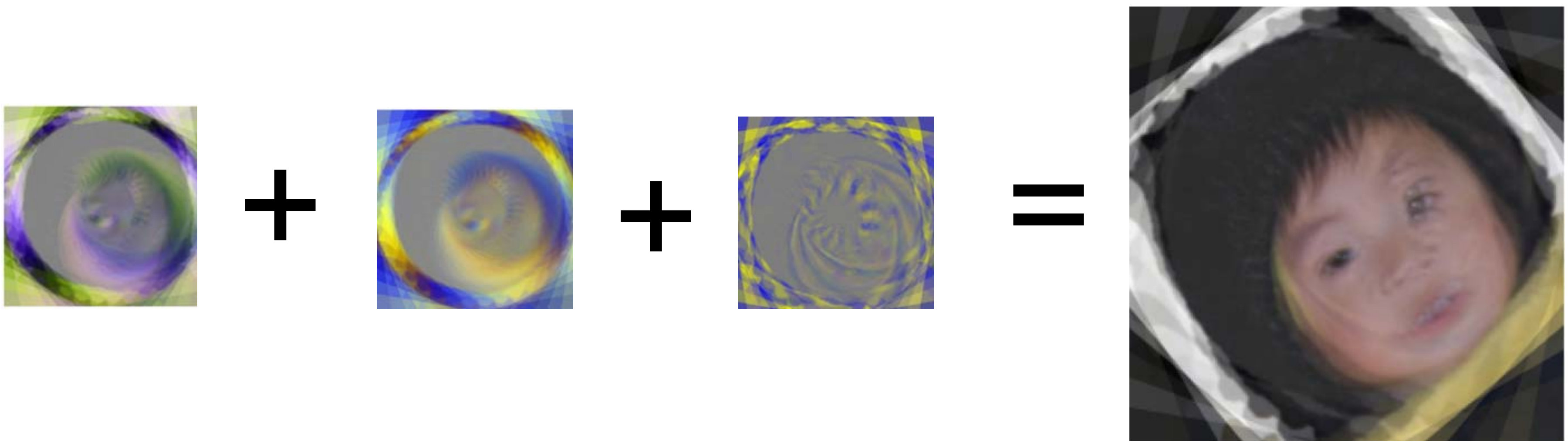
Image_1 Image_2

+

=

Averaging Images != Rotating Complex Objects

Averaging Images



Averaging 'Eigen' Images = Rotating Objects

Cat-Baby Averaging

Object Averaging with feature matching!

- Nose to nose, eye to eye, mouth to mouth, etc.
- This is a non-parametric **warp**

Cat-Baby Averaging

Object Averaging with feature matching (warping)!

- Nose to nose, eye to eye, mouth to mouth, etc.
- This is a non-parametric warp

Warping, then cross-dissolve

Morphing procedure:

1. Find the average shape
2. Non-parametric warping
3. Find the average color
   – Cross-dissolve the warped images

Image warping – non-parametric

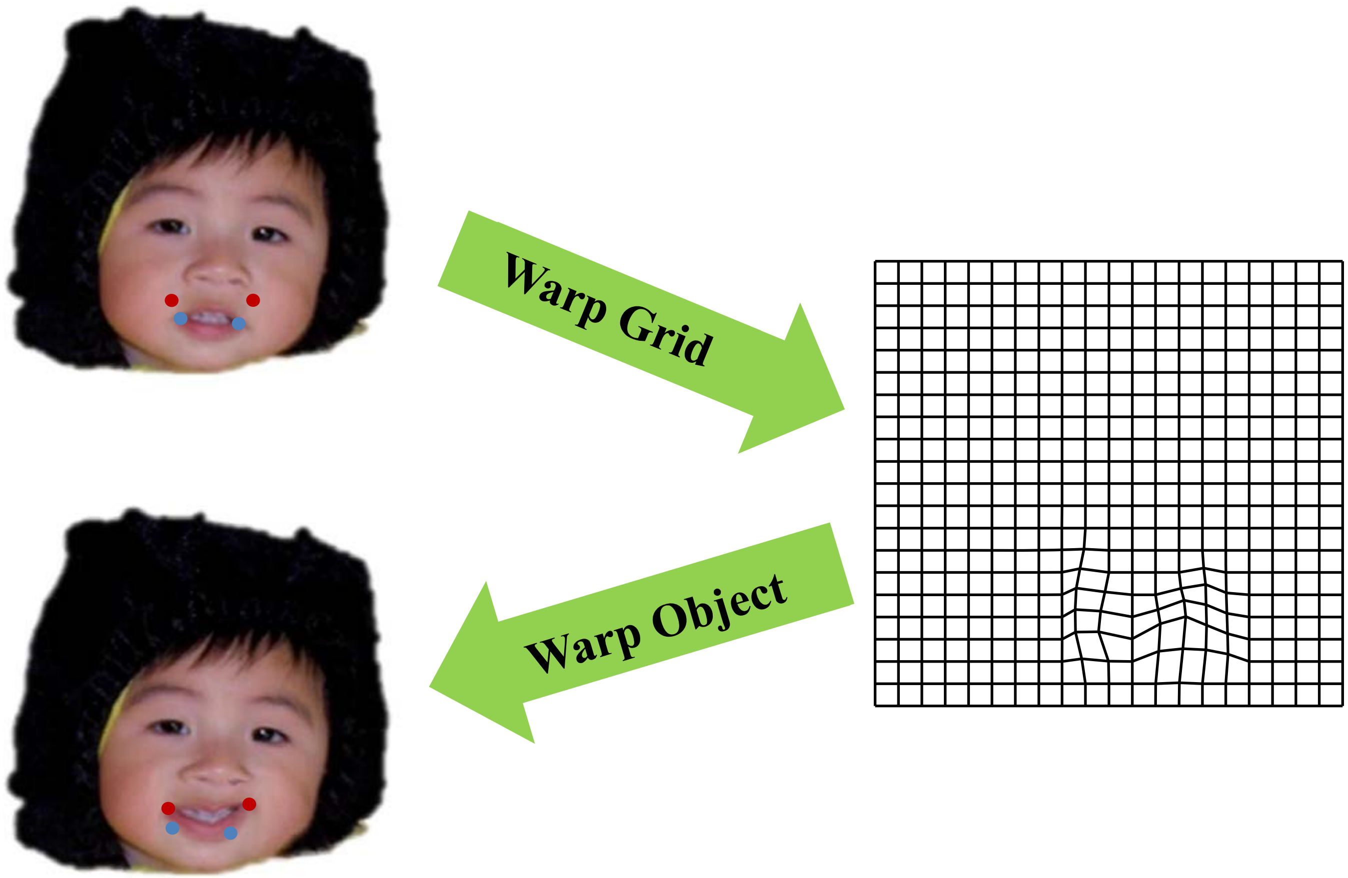
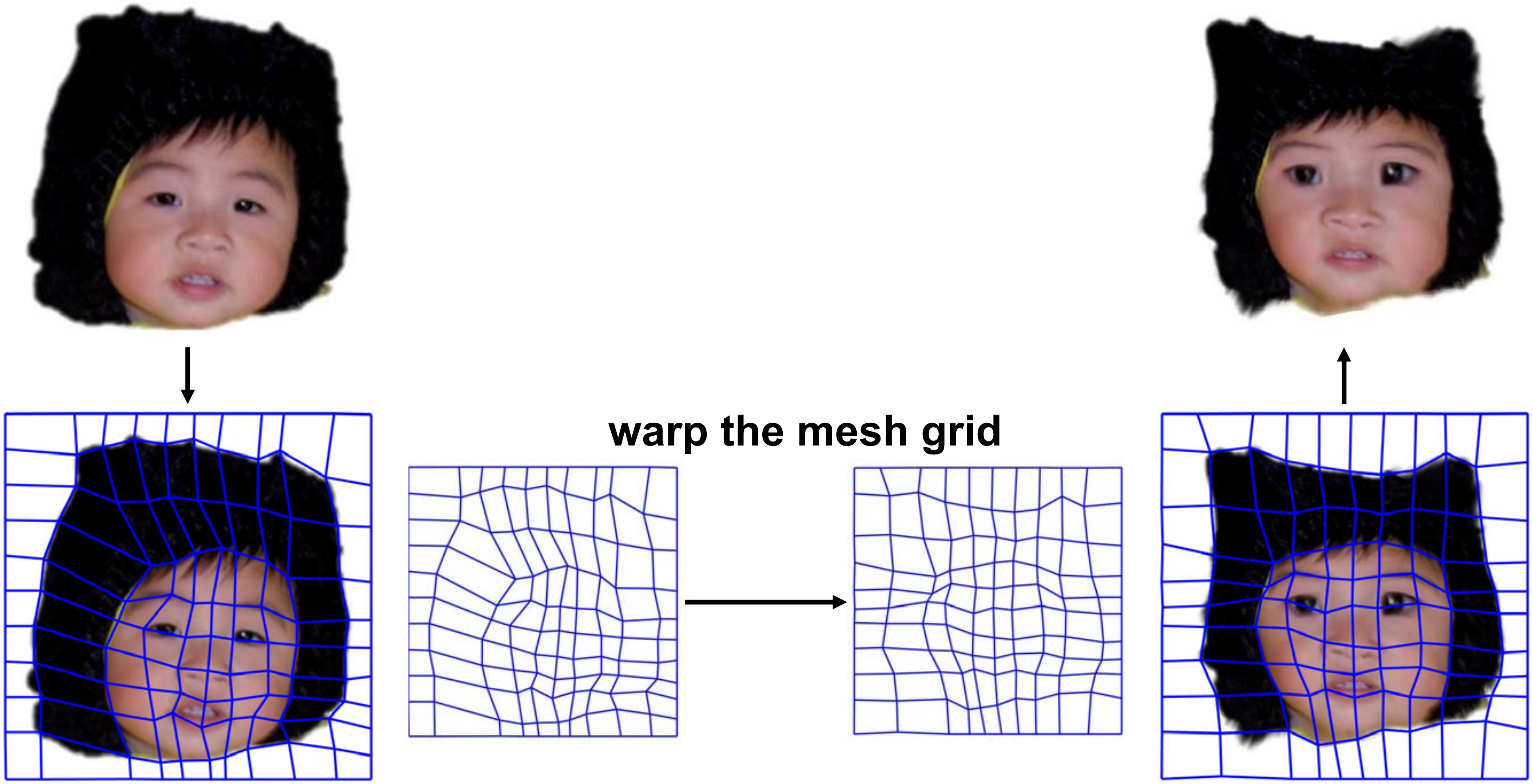


Image warping idea 1: dense flow

Displacement vector (u,v) for each pixel.

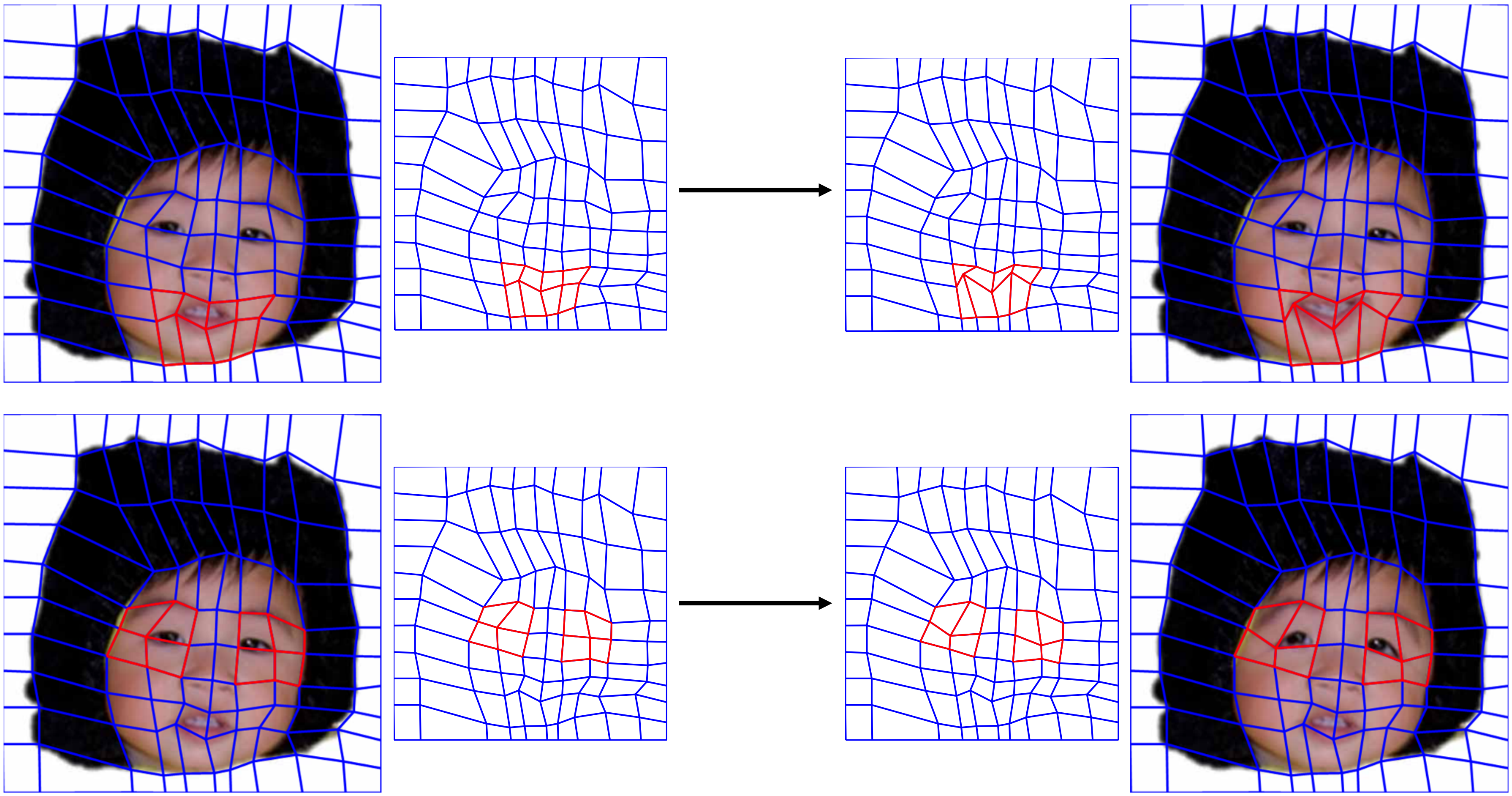
Great details… but too much work, let's simply it to mesh grid

Image warping idea 2 : dense grid



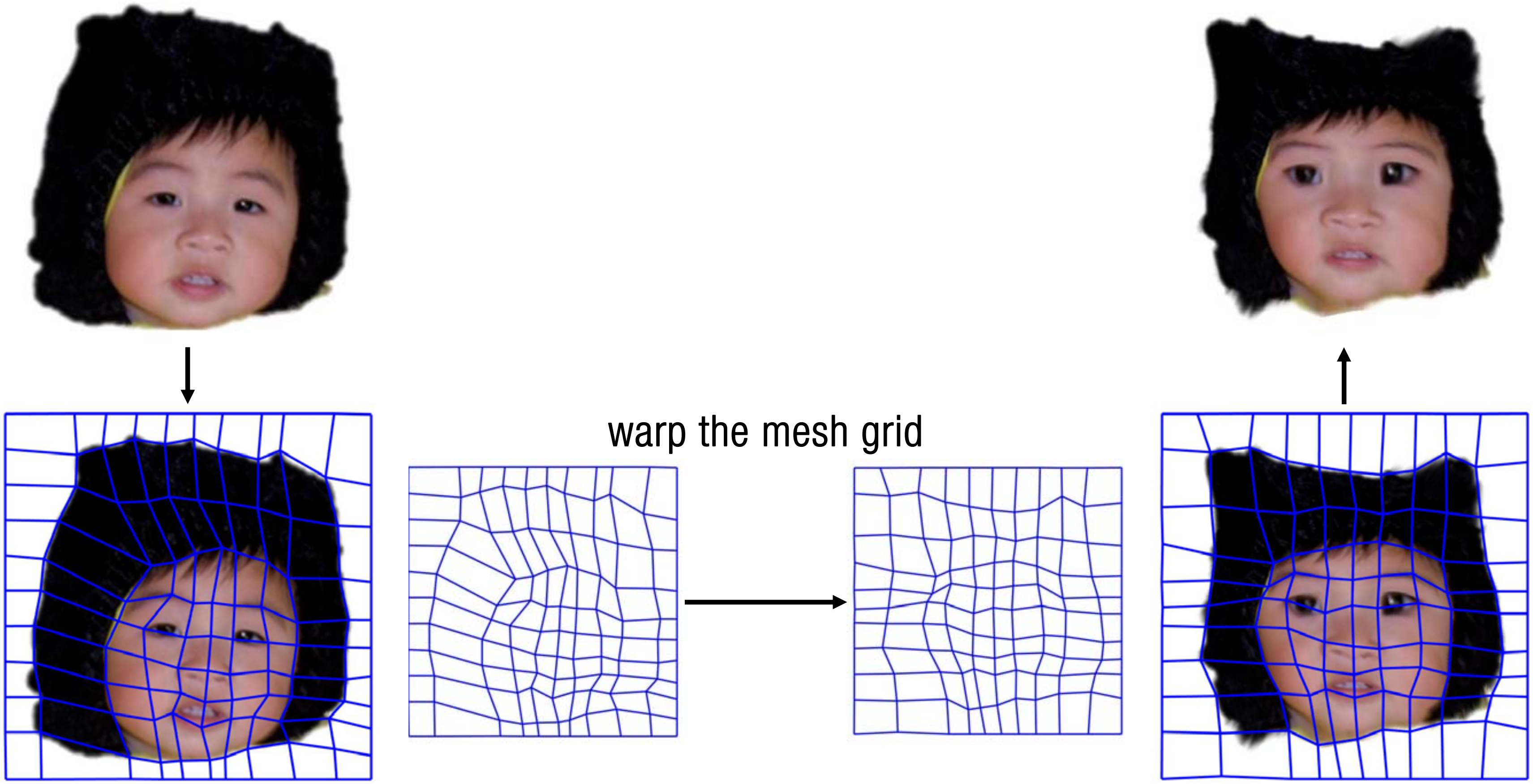
Define and manipulate the mesh grid

Image warping idea 2 : dense grid



Grid deformation generates expression change

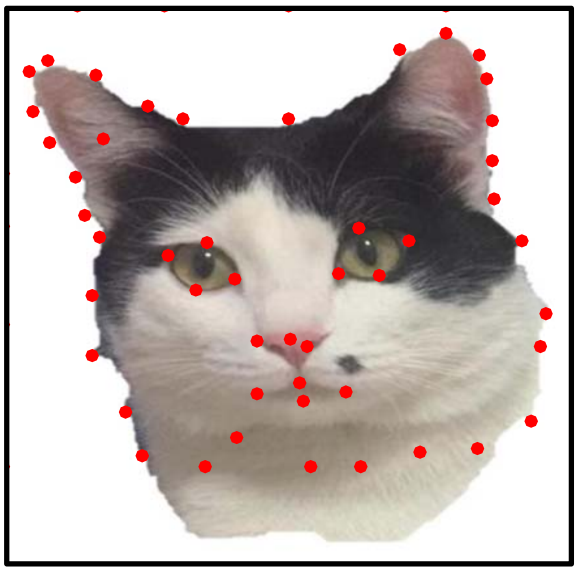
Image warping idea 2 : dense grid

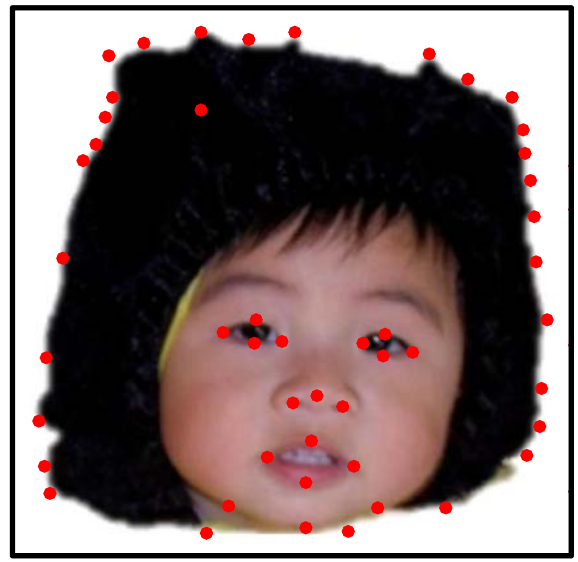


Still too much work…

simplify it to sparse control points and triangles

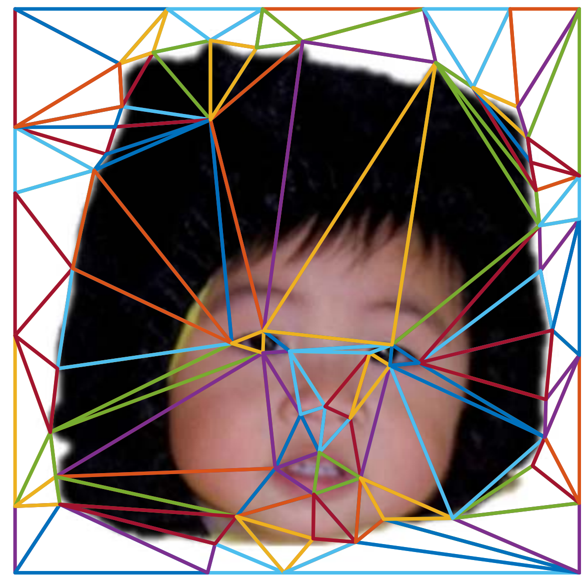
Image warping idea 3 : sparse points





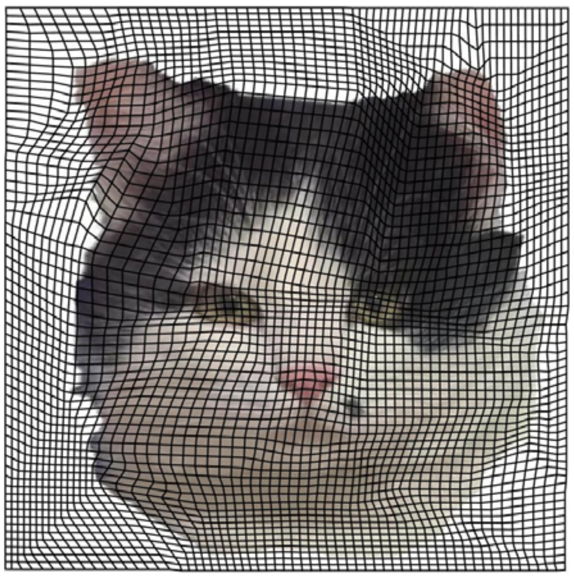
Specify sparse points and their correspondence

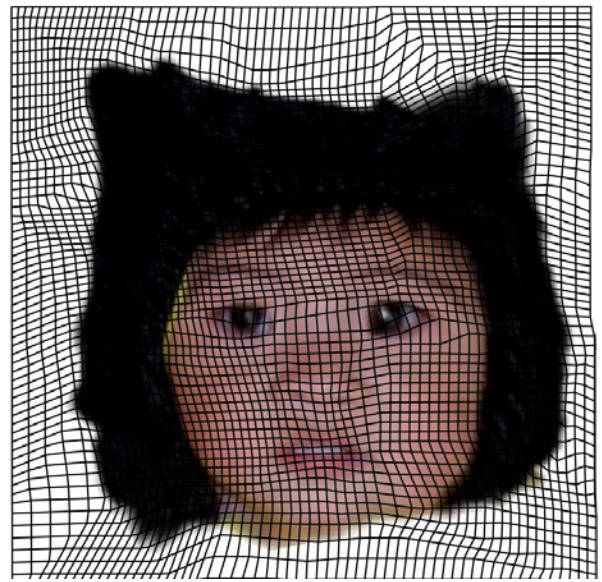
Image warping idea 3 : sparse points



- Define a triangular mesh over the feature points
- Triangle-to-triangle correspondences
- Warp each triangle separately from source to destination

From sparse points to dense grid

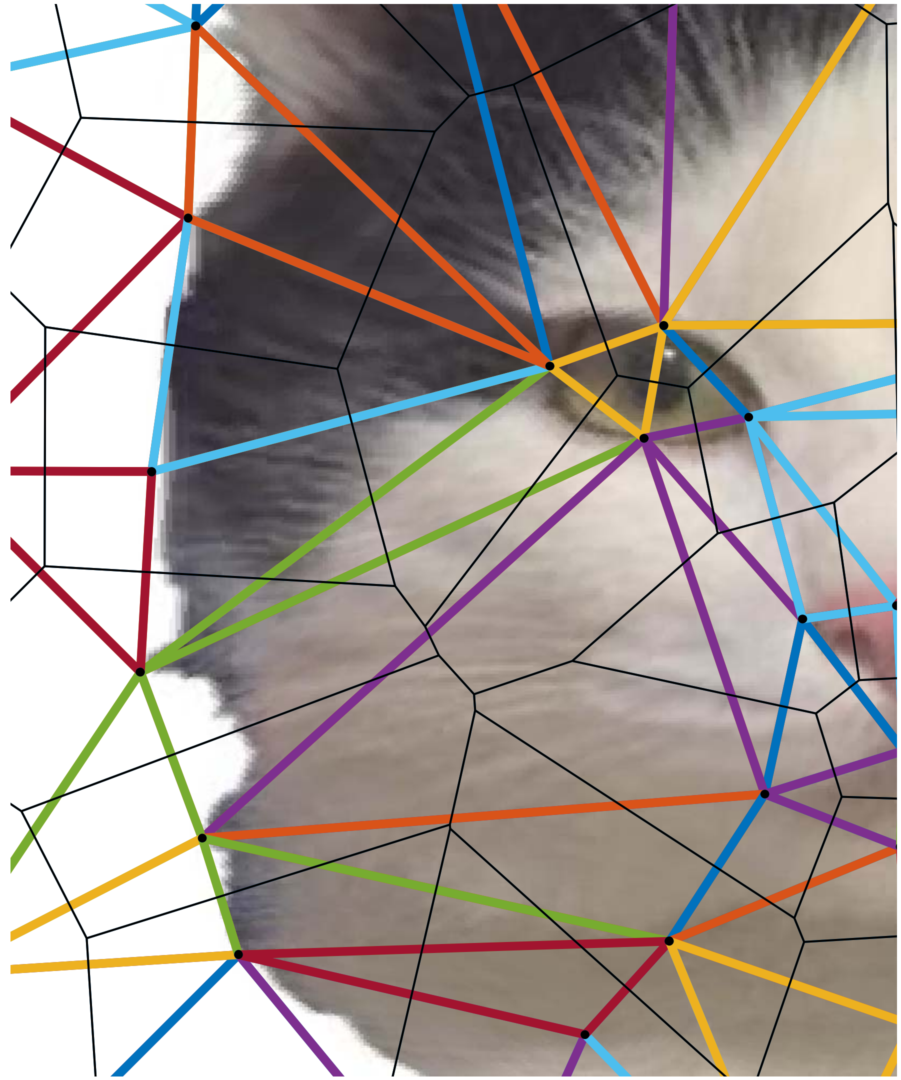




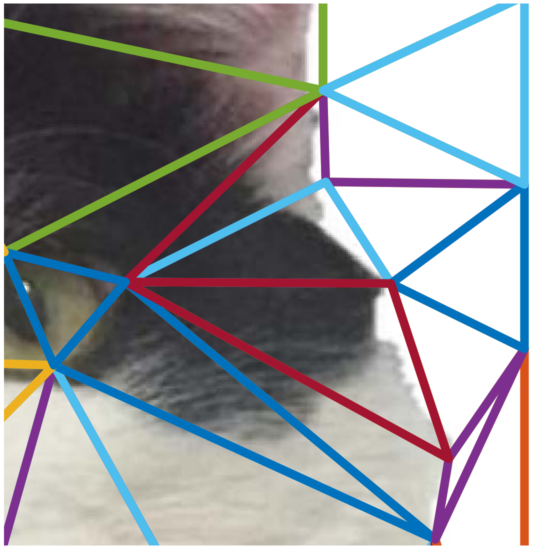
- Warping on triangulation corresponds to warping on dense grid, and dense pixel flow

Delaunay Triangulation

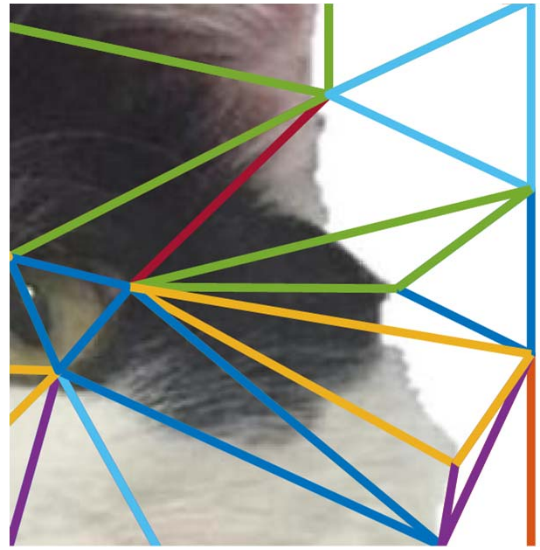
- Draw the dual to the Voronoi diagram by connecting each two neighboring sites in the Voronoi diagram.
- The DT may be constructed in O(nlogn) time.
- This is what Matlab's delaunay function uses.



What is good feature points



Good

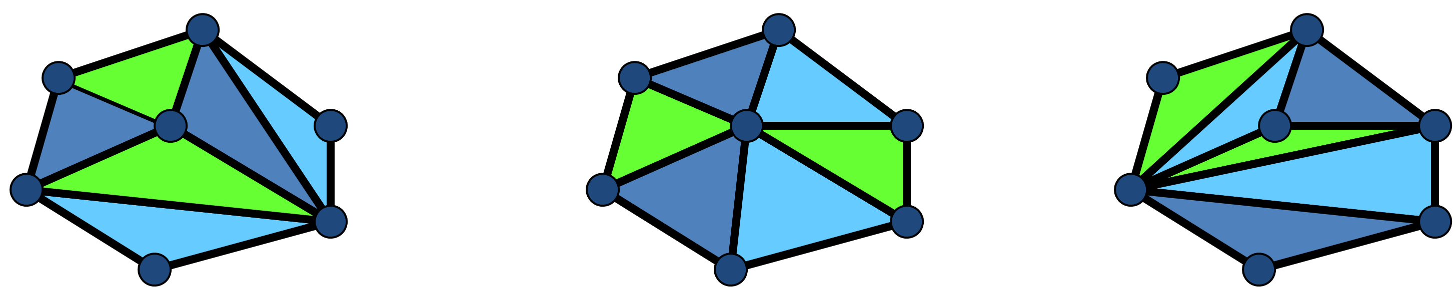


Bad

- The triangulation is consistent with image boundary
  - Texture regions won't fade into the background when morphing
- Maintain the relationship between parts

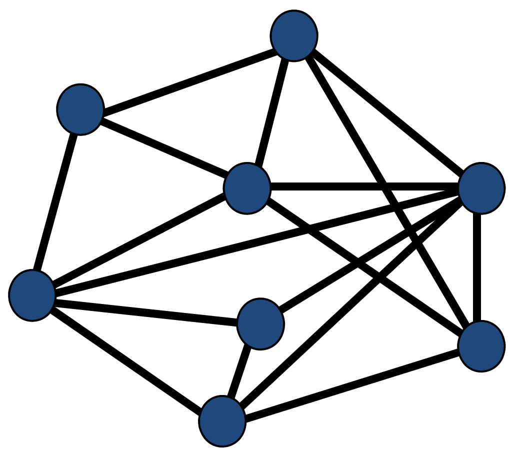
Triangulations

- A *triangulation* of set of points in the plane is a *partition* of the convex hull to triangles whose vertices are the points, and do not contain other points.
- There are an exponential number of triangulations of a point set.



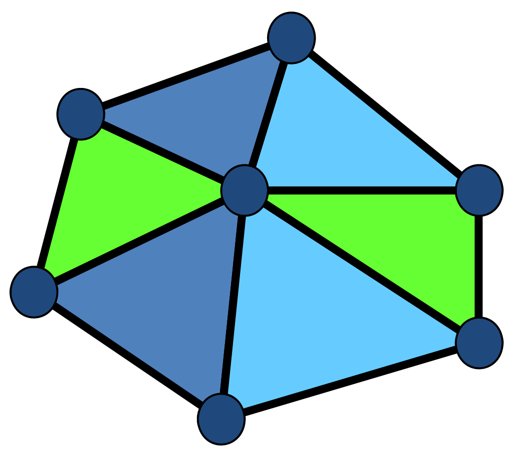
An O(n^3) Triangulation Algorithm

- Repeat until impossible:
  - Select two sites.
  - If the edge connecting them does not intersect previous edges, keep it.

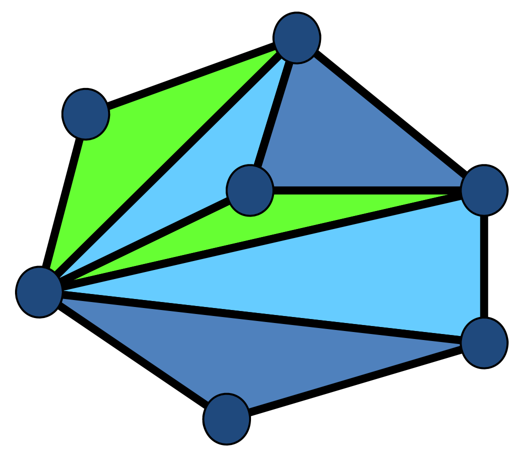


"Quality" Triangulations

- Let $\alpha(T) = (\alpha_1, \alpha_2 ,.., \alpha_{3t})$ be the vector of angles in the triangulation T in increasing order.
- A triangulation T_1 will be "better" than T_2 if $\alpha(T_1) > \alpha(T_2)$ lexicographically.
- The Delaunay triangulation is the "best"
  - Maximizes smallest angles

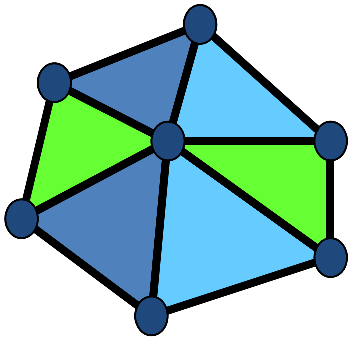


good

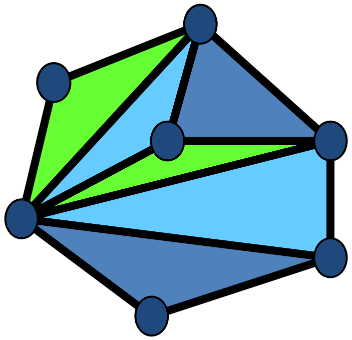


bad

Boris Nikolaevich Delaunay (March 15, 1890 – July 17, 1980)



Delaunay

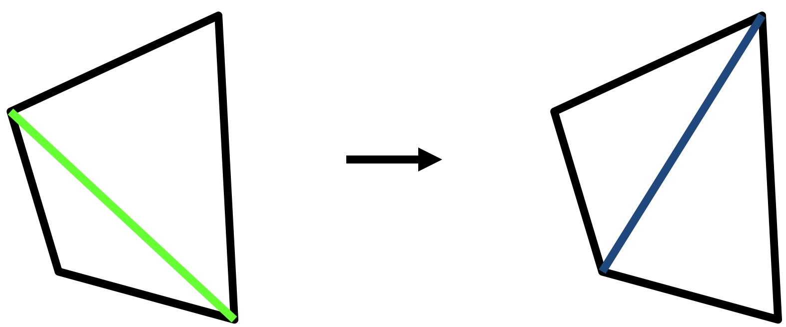


bad

http://higeom.math.msu.su/history/delone_r.html

Improving a Triangulation

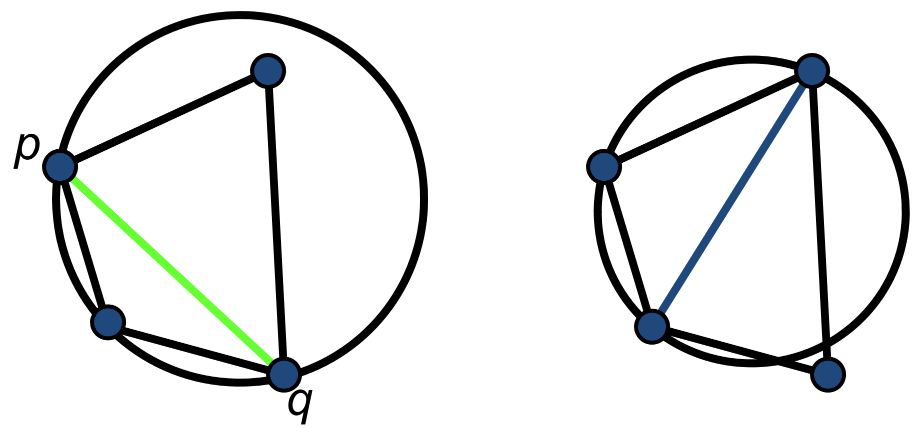
- In any convex quadrangle, an *edge flip* is possible. If this flip *improves* the triangulation locally, it also improves the global triangulation.



If an edge flip improves the triangulation, the first edge is called *illegal*.

Illegal Edges

- **Lemma:** An edge *pq* is illegal iff one of its opposite vertices is inside the circle defined by the other three vertices.
- **Proof:** By Thales' theorem.



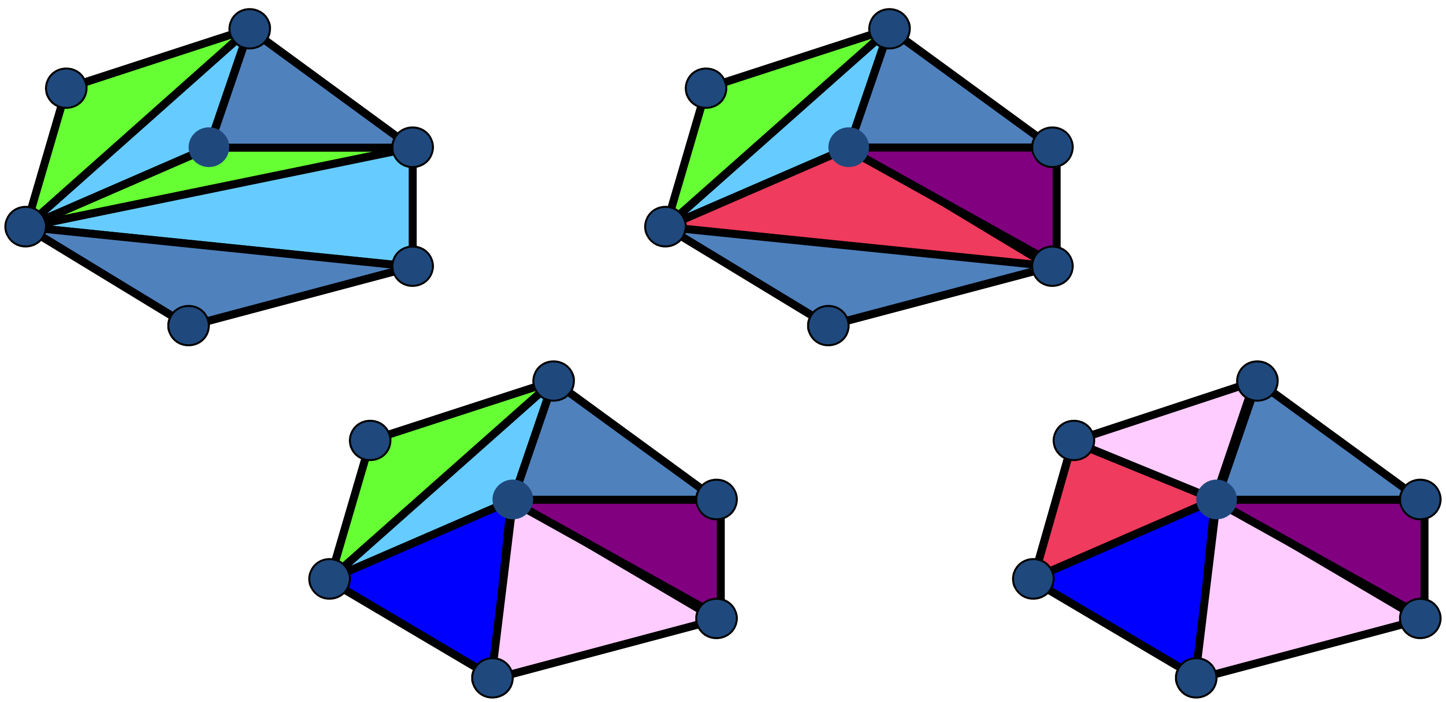
Theorem: A Delaunay triangulation does not contain illegal edges.

Corollary: A triangle is Delaunay iff the circle through its vertices is empty of other sites.

Corollary: The Delaunay triangulation is not unique if more than three sites are co-circular.

Naïve Delaunay Algorithm

- Start with an arbitrary triangulation. Flip any illegal edge until no more exist.
- Could take a long time to terminate.

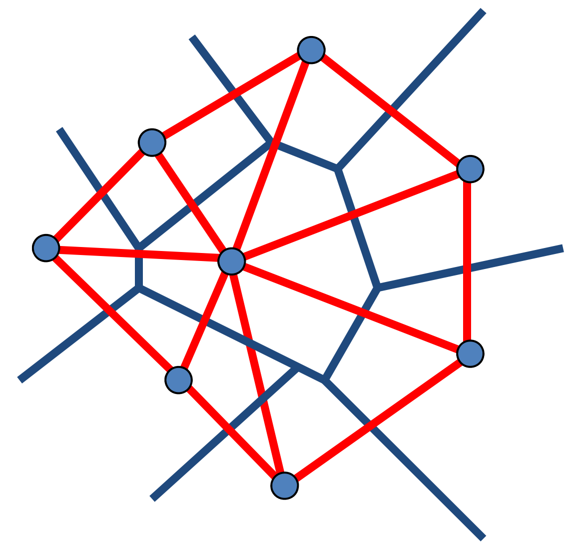


Delaunay Triangulation by Duality

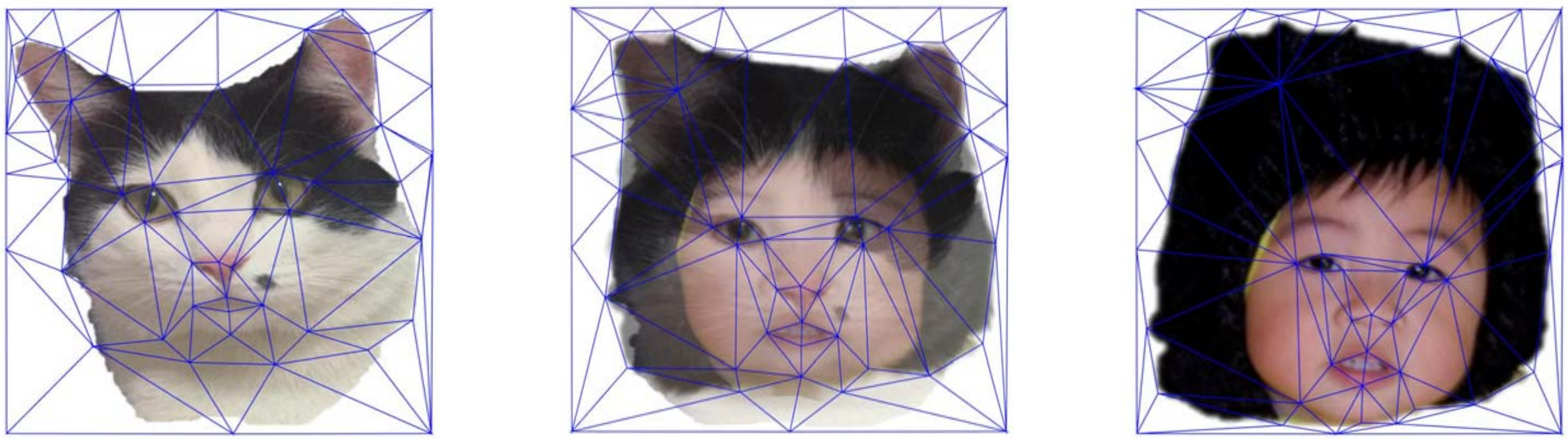
- General position assumption: There are no four co-circular points.
- Draw the dual to the Voronoi diagram by connecting each two neighboring sites in the Voronoi diagram.

- **Corollary:** The DT may be constructed in O($n$log$n$) time.

- This is what Matlab's `delaunay` function uses.



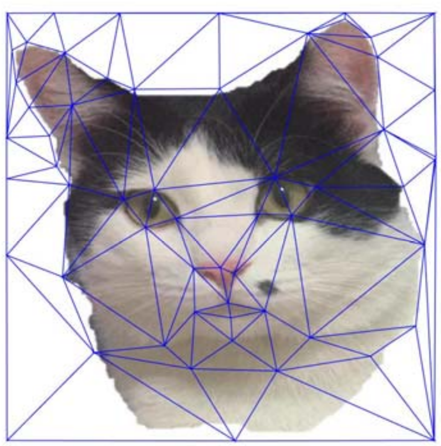
Triangular Mesh

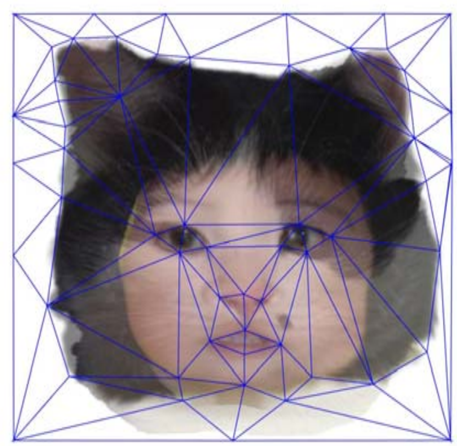


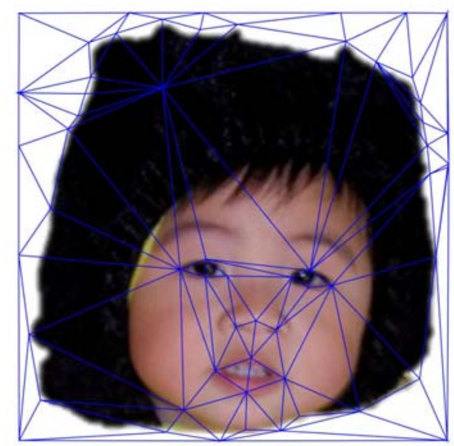
1. Input correspondences at key feature points
2. Define a triangular mesh over the points
   - Same mesh in both images!
   - Now we have triangle-to-triangle correspondences

Warp interpolation

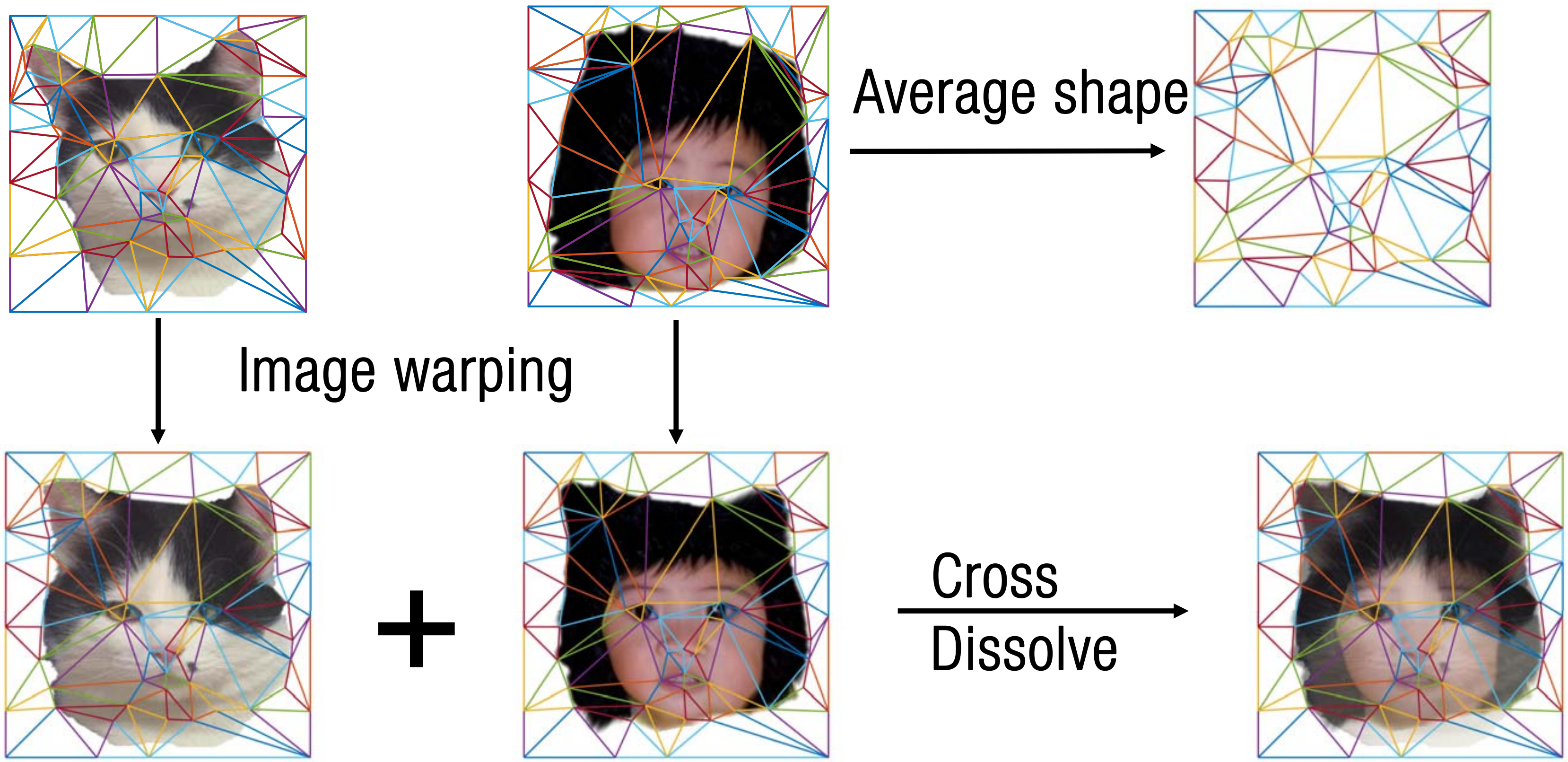
- How do we create an intermediate warp at time t?
  - Assume t = [0,1]
  - Simple linear interpolation of each feature pair
  - (1-t)*p0+t*p1 for corresponding features p0 and p1





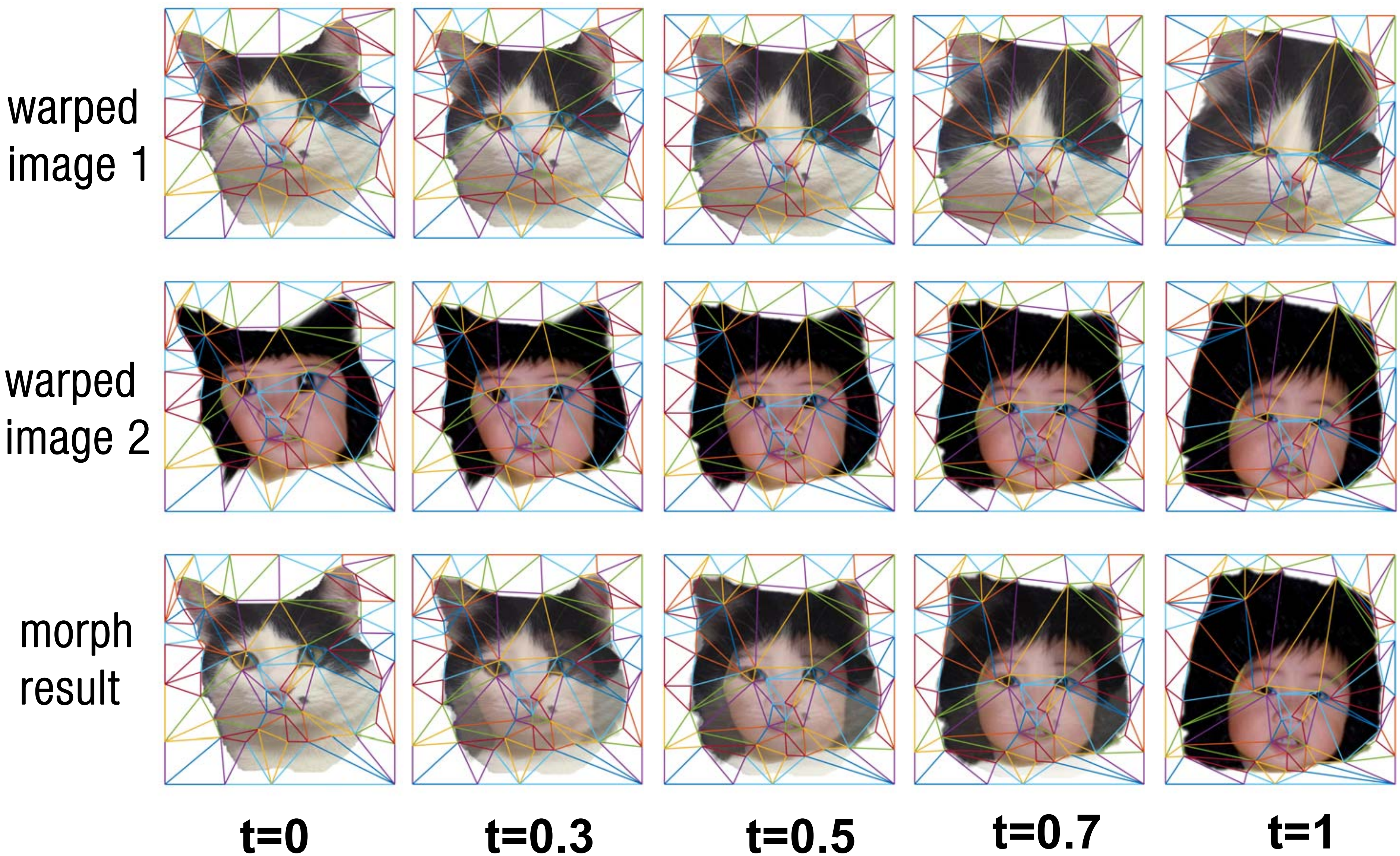


Morphing = Warping + Cross-dissolve

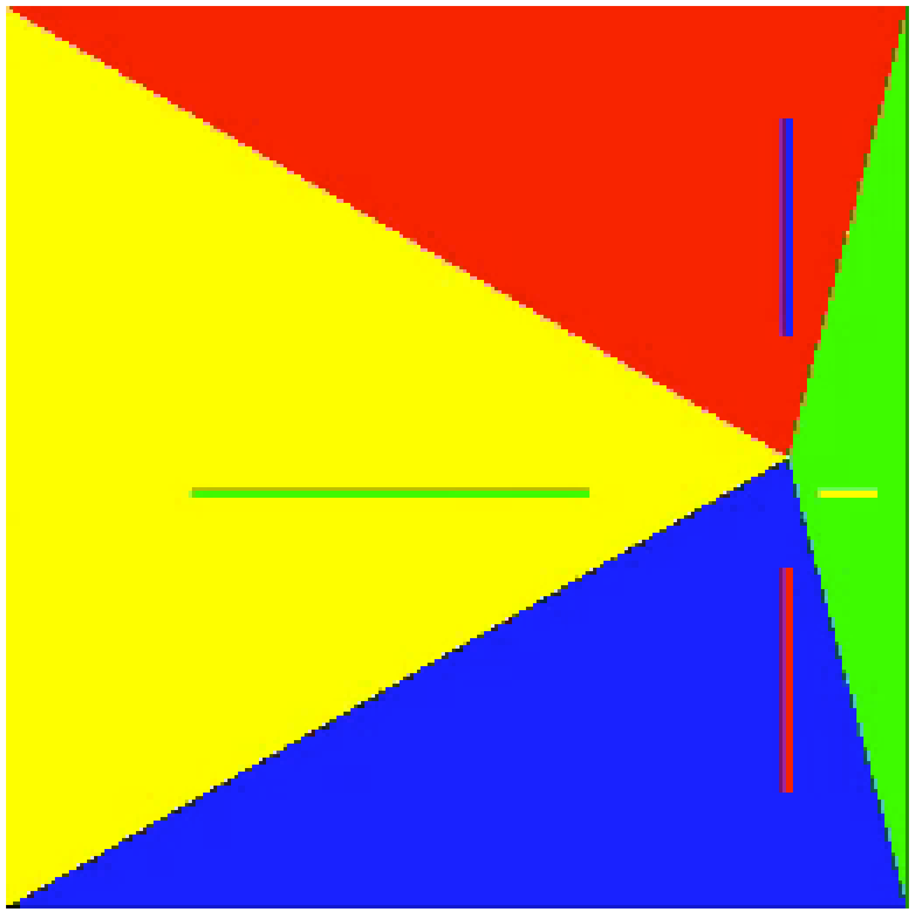


- For each time t, define the intermediate shape
  - $p_t = (1-t) \times p_1 + t \times p_2$
  - triangulation doesn't change
- Warp both image to the intermediate shape
- Dissolve image = $(1\text{-}t) \times \text{image}_1 + t \times \text{Image}_2$

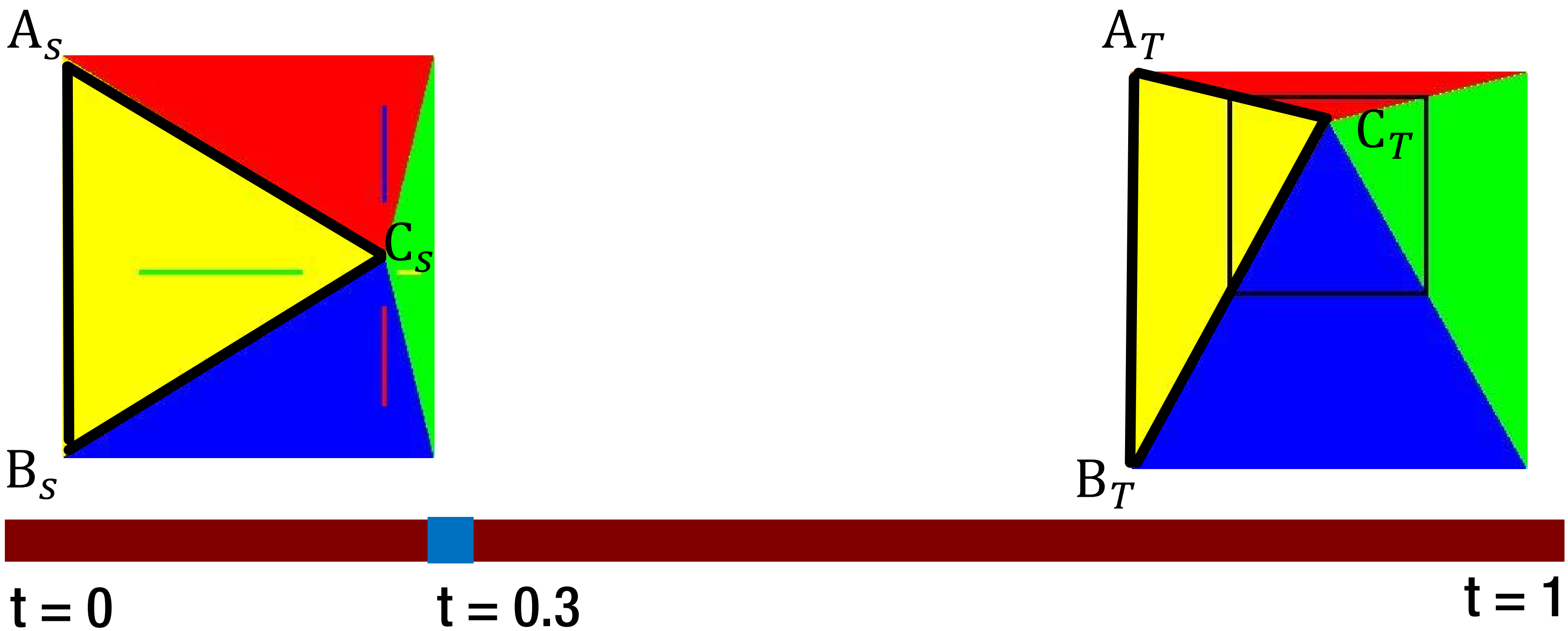
Morphing Sequence



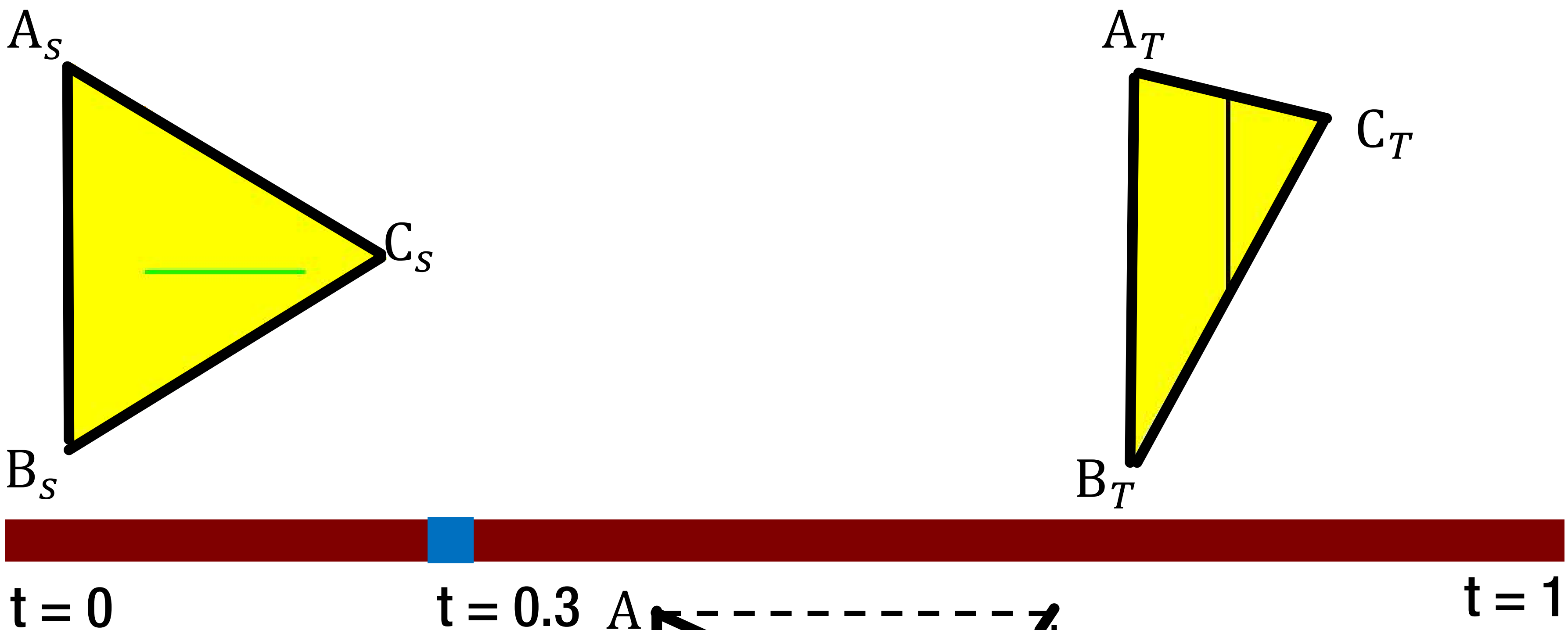
An Example

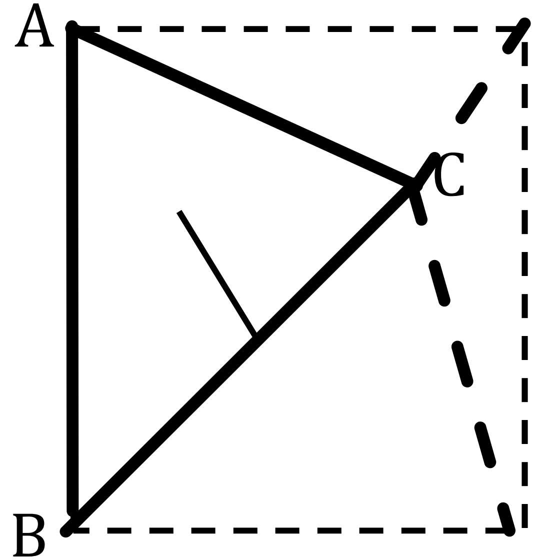


Morphing



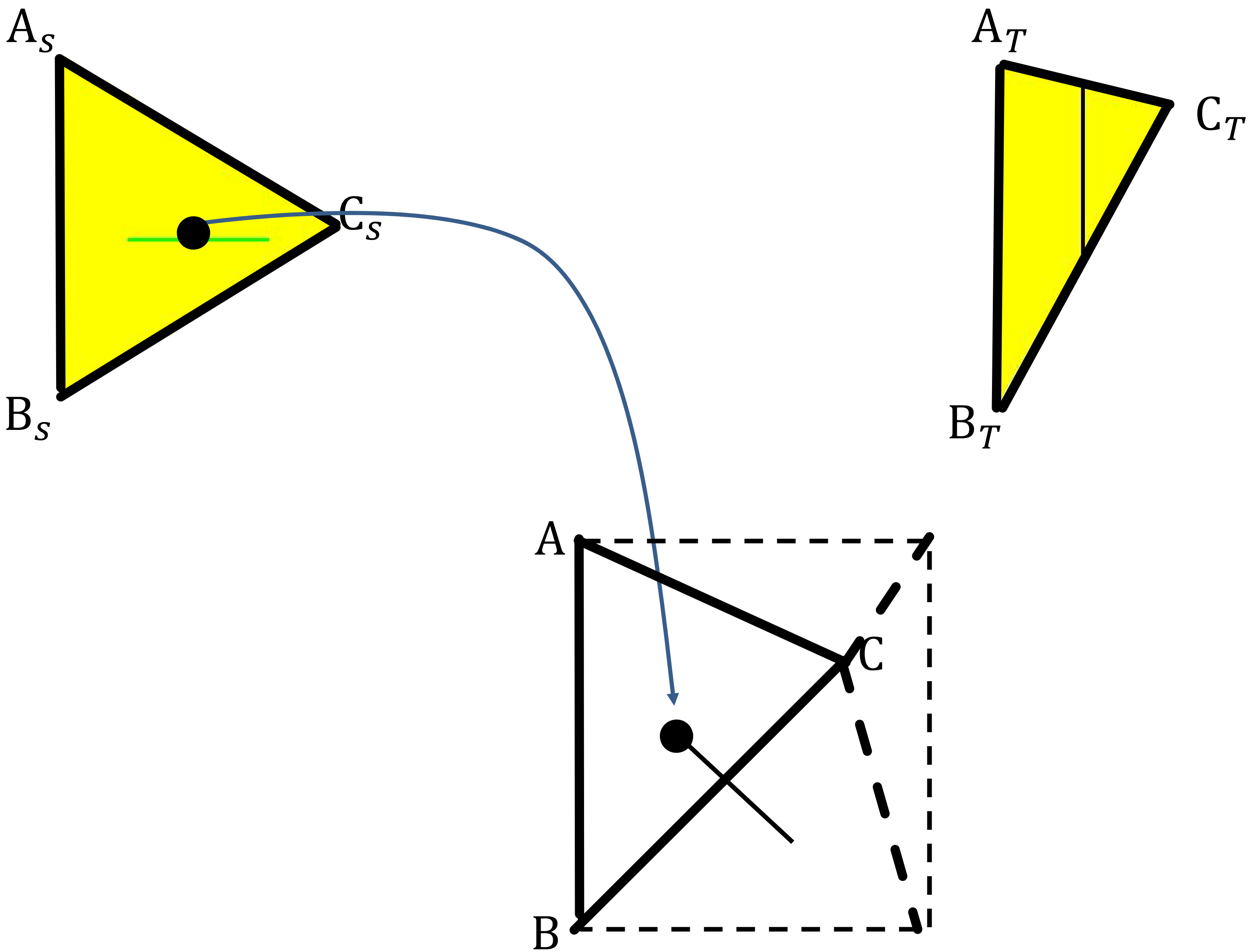
Step 1: Triangle interpolation





$$\mathbf{A}_t = (1-t)\mathbf{A}_S + t\mathbf{A}_T$$
$$\mathbf{B}_t = (1-t)\mathbf{B}_S + t\mathbf{B}_T$$
$$\mathbf{C}_t = (1-t)\mathbf{C}_S + t\mathbf{C}_T$$

Step 2: Warping



Step 2: Warping

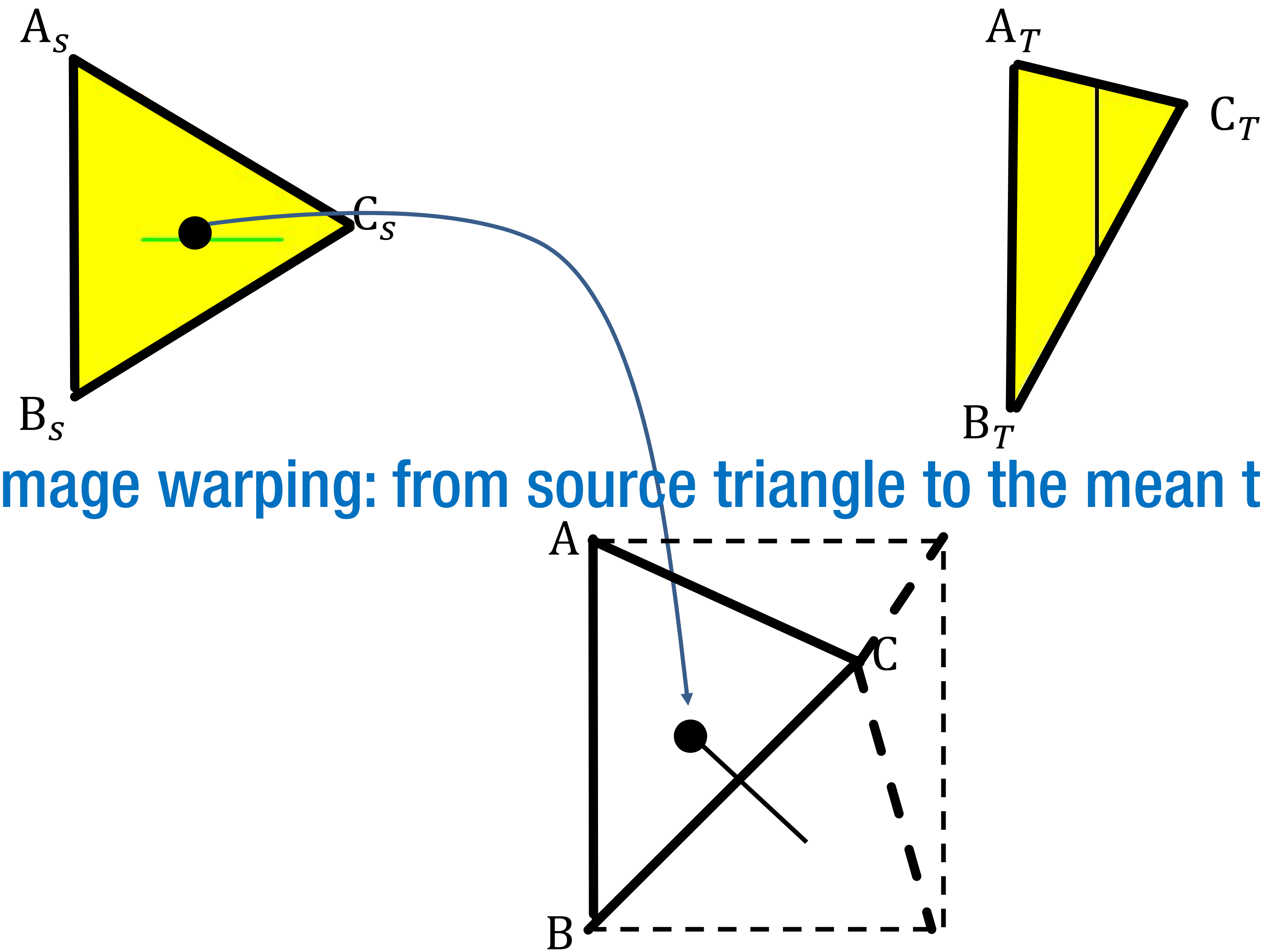
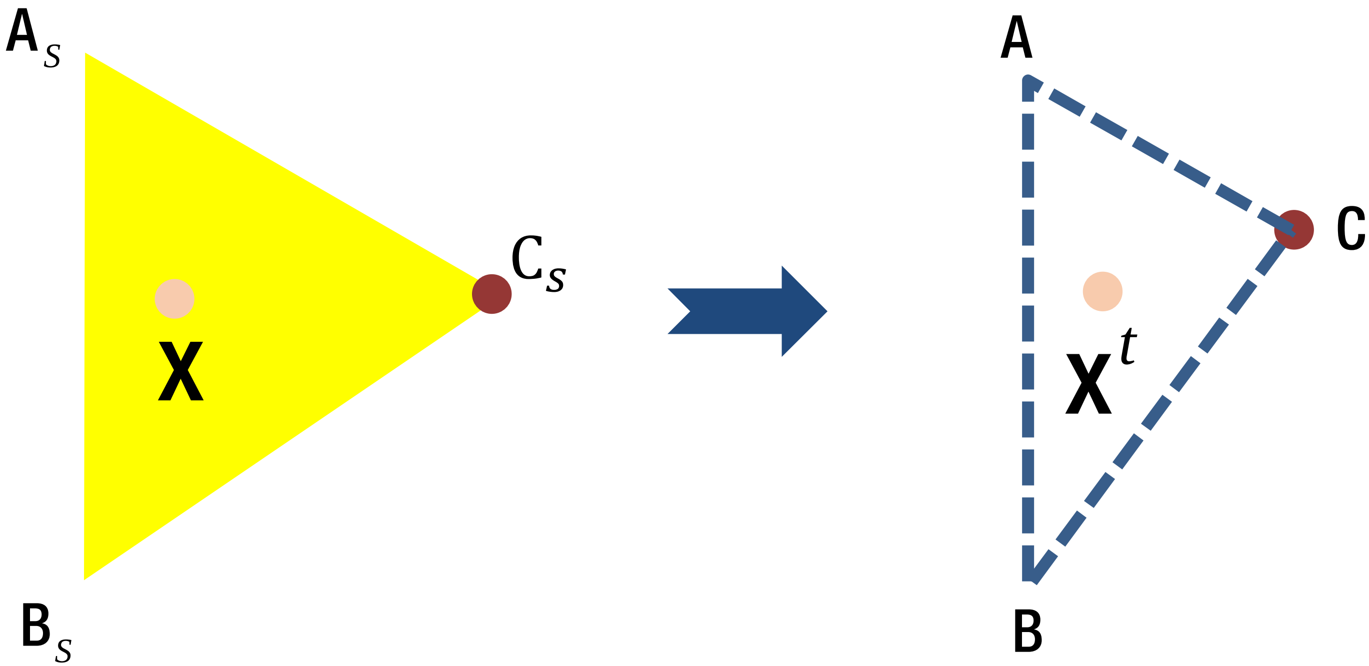


Image warping: from source triangle to the mean triangle

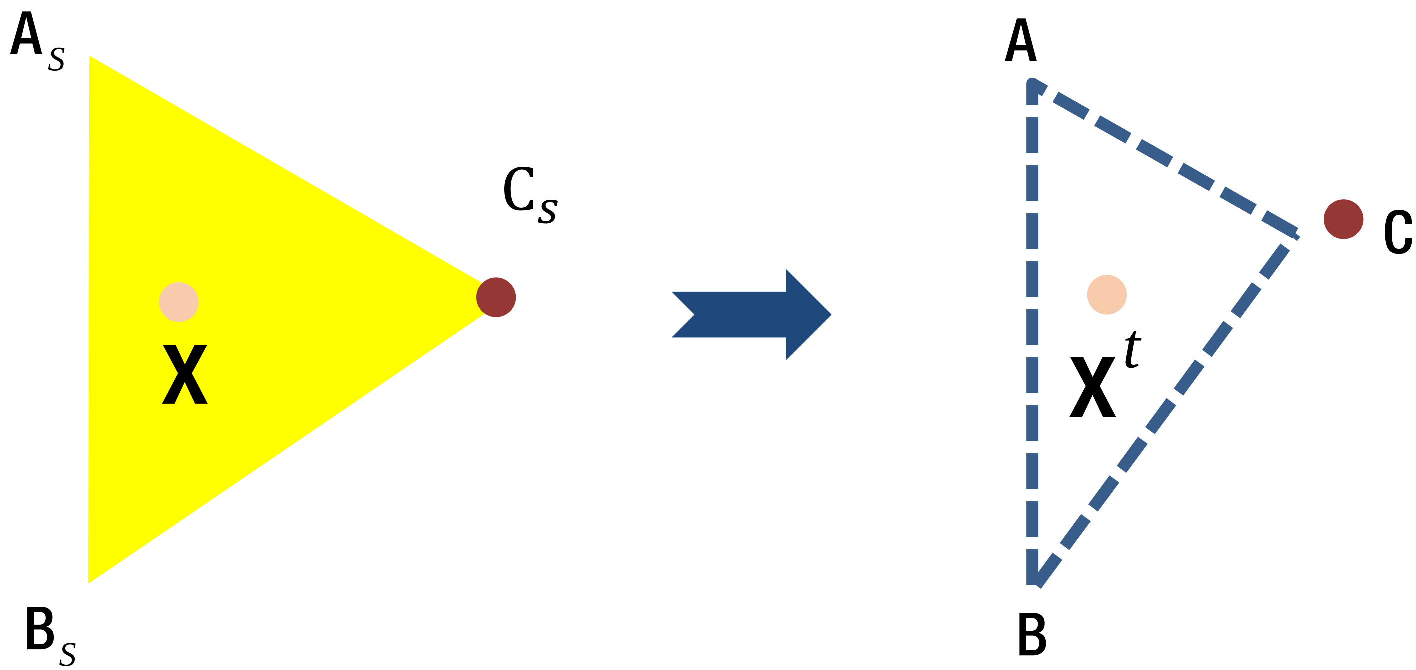
Triangle warping = Affine transform



Affine transform is a pixel transportation $X \rightarrow X^t$

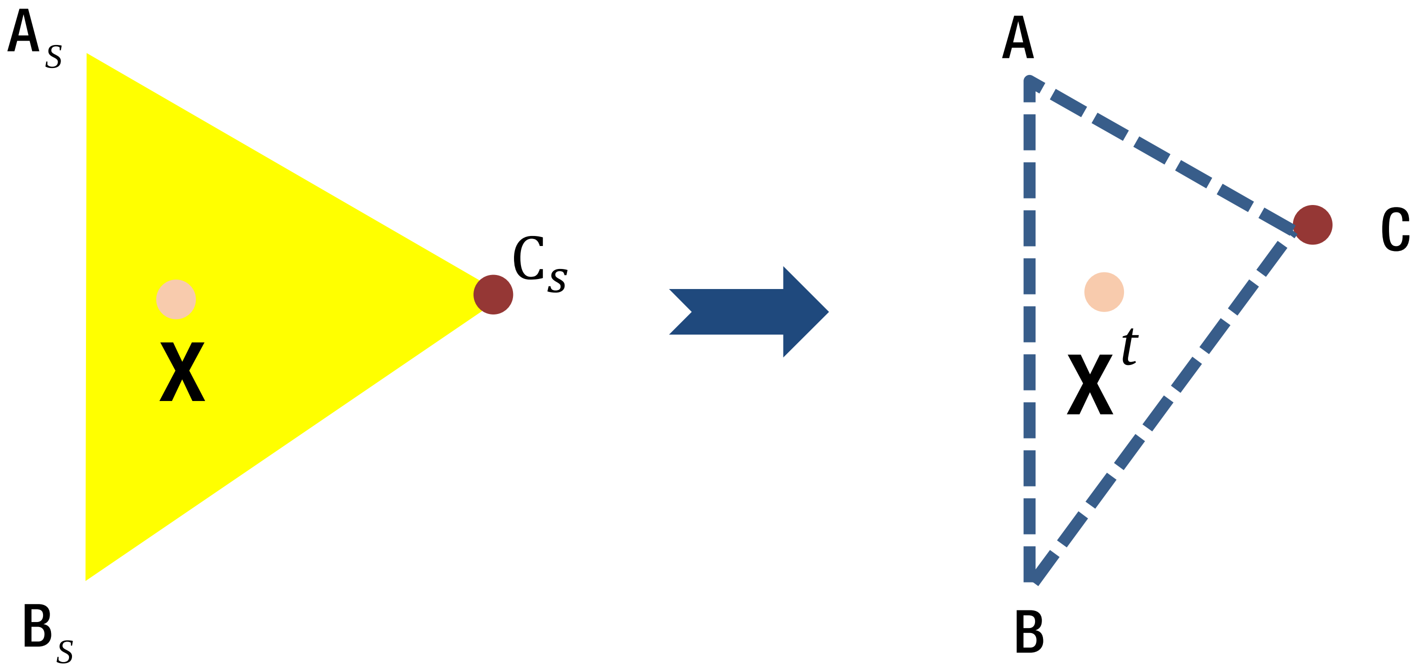
It is controlled by the movement of the three vertices of the triangle

Barycentric Coordinates



Each point **X** has an invariant representation with respect to the three vertices.

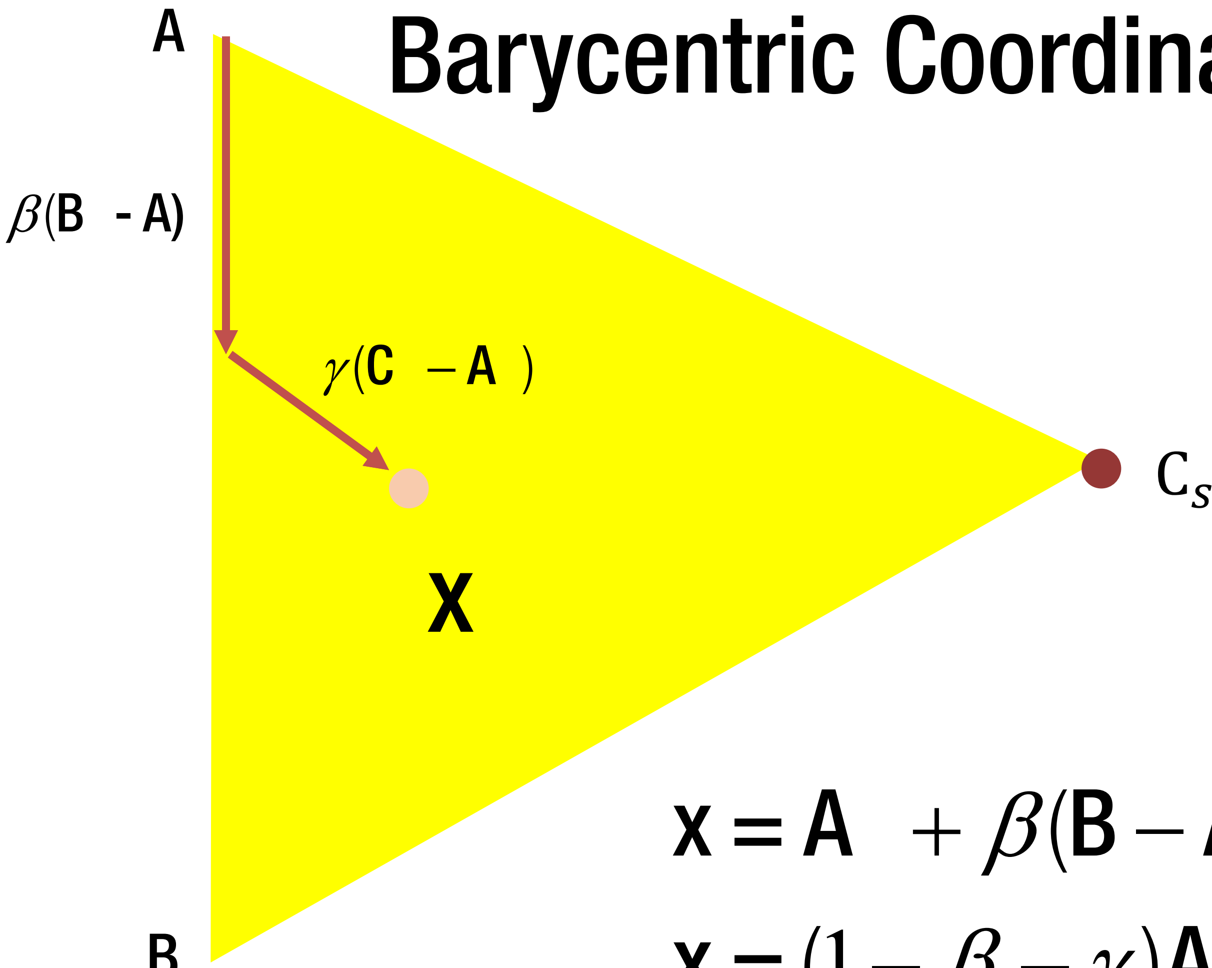
Barycentric Coordinates



$$\mathbf{X} = \alpha \mathbf{A}_s + \beta \mathbf{B}_s + \gamma \mathbf{C}_s \qquad \mathbf{X}^t = \alpha \mathbf{A}_t + \beta \mathbf{B}_t + \gamma \mathbf{C}_t$$

$$\alpha + \beta + \gamma = 1$$

Barycentric Coordinates

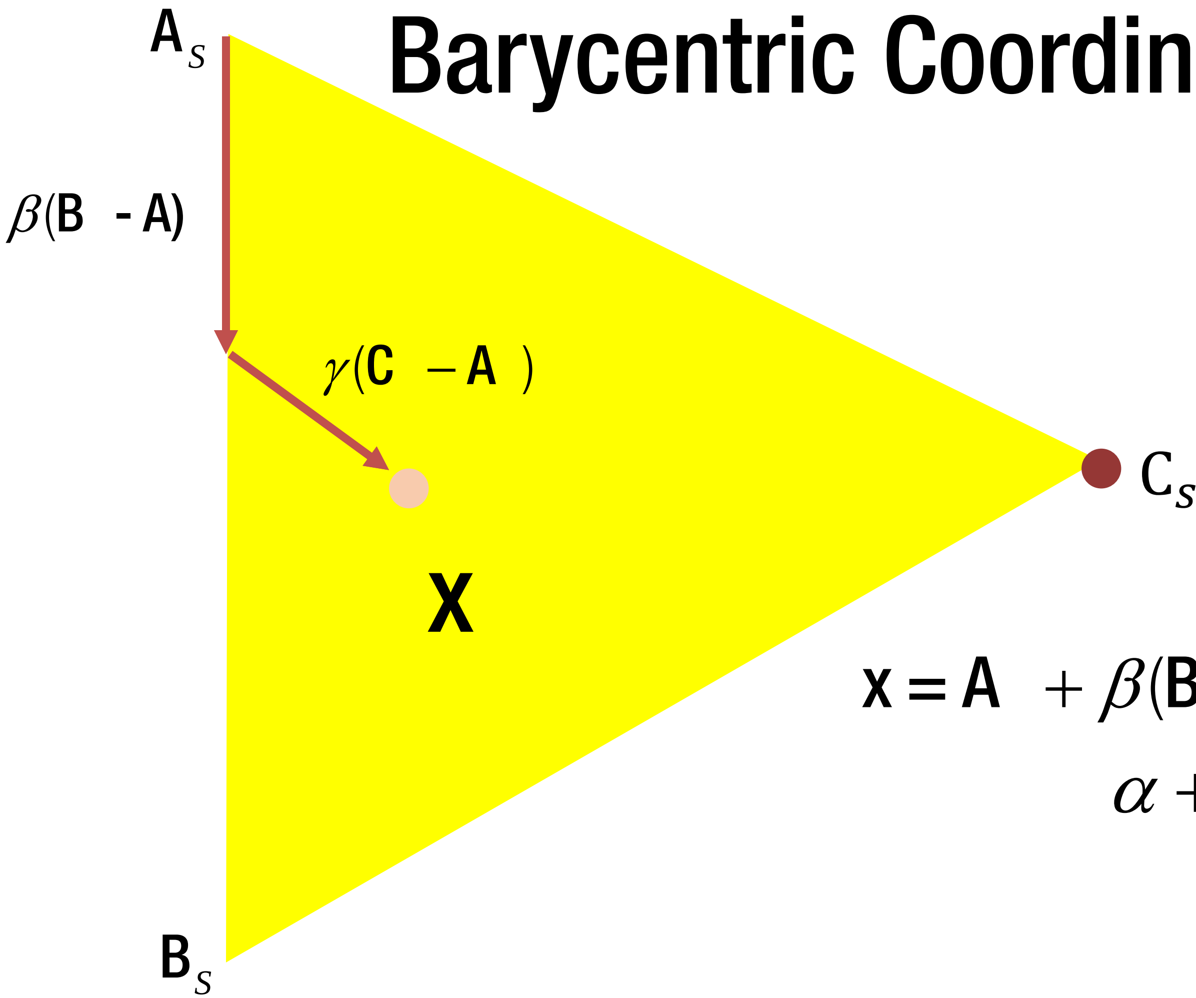


$$\mathbf{x} = \mathbf{A} + \beta(\mathbf{B} - \mathbf{A}) + \gamma(\mathbf{C} - \mathbf{A})$$

$$\mathbf{x} = (1 - \beta - \gamma)\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$$

$$\mathbf{x} = \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C} \qquad \alpha + \beta + \gamma = 1$$

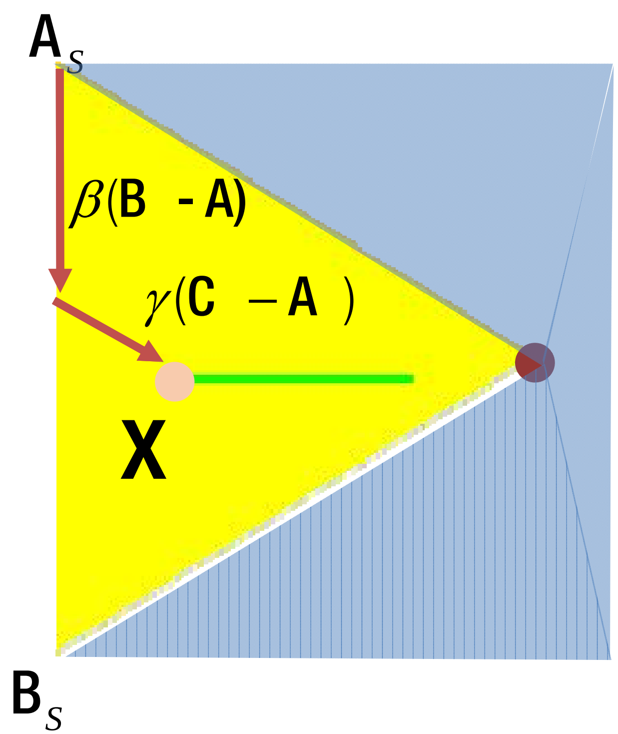
Barycentric Coordinates



$$\mathbf{x} = \mathbf{A} + \beta(\mathbf{B} - \mathbf{A}) + \gamma(\mathbf{C} - \mathbf{A})$$

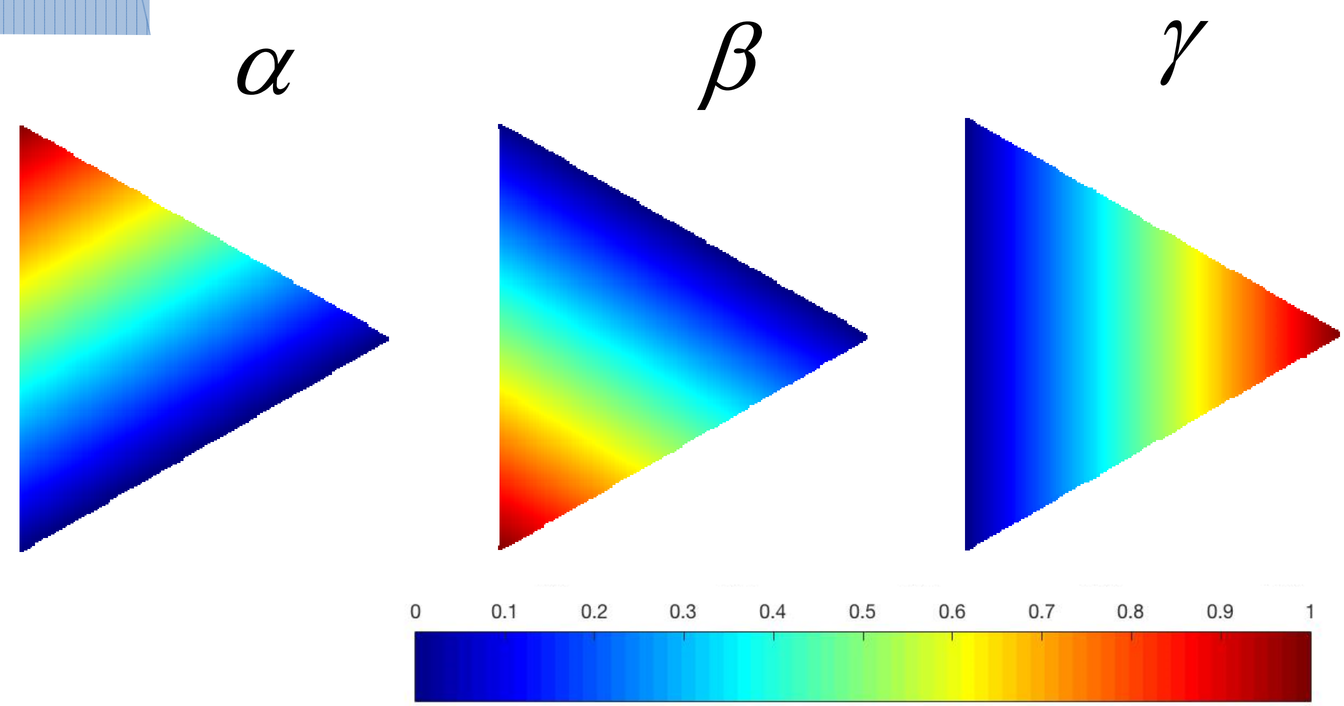
$$\alpha + \beta + \gamma = 1$$

$$\begin{bmatrix} \mathbf{A}_x & \mathbf{B}_x & \mathbf{C}_x \\ \mathbf{A}_y & \mathbf{B}_y & \mathbf{C}_y \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \lambda \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$ linear equations in 3 unknowns



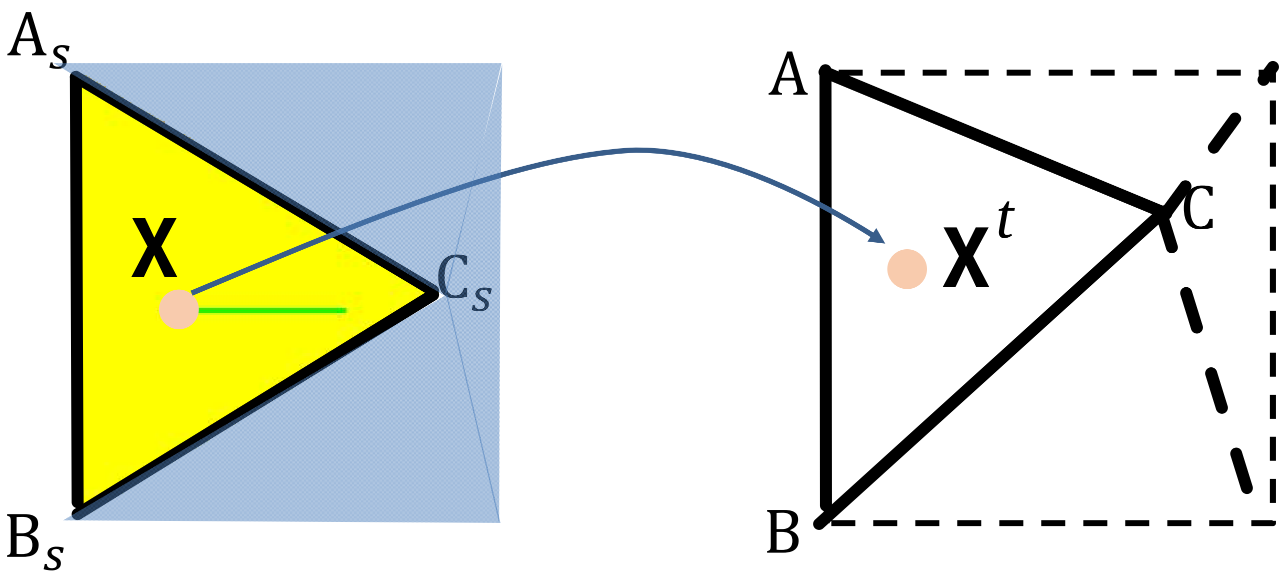
Barycentric coordinate

$$\begin{bmatrix} \mathbf{A}_x & \mathbf{B}_x & \mathbf{C}_x \\ \mathbf{A}_y & \mathbf{B}_y & \mathbf{C}_y \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \lambda \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

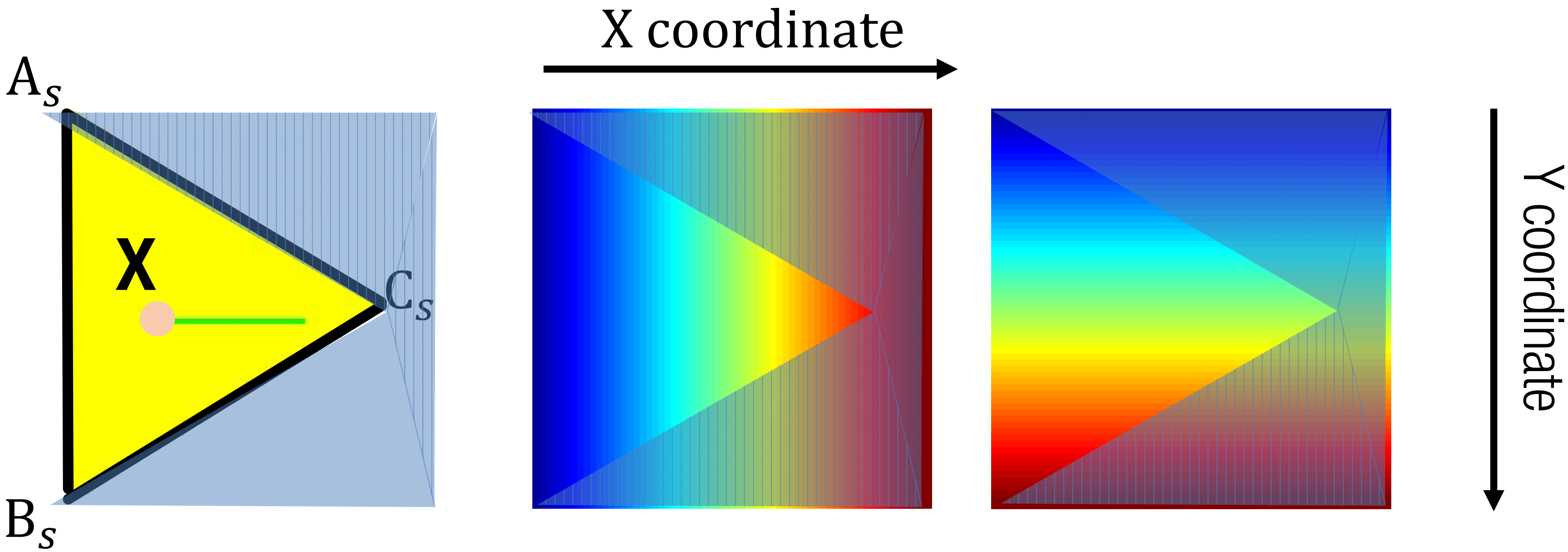


Warping with Barycentric Coordinate

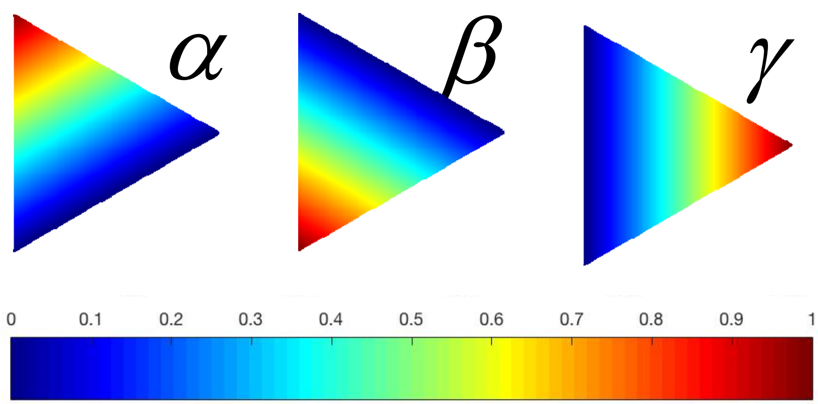
$$\mathbf{X} = \alpha \mathbf{A}_S + \beta \mathbf{B}_S + \gamma \boldsymbol{C}_S \qquad \mathbf{X}^t = \alpha \mathbf{A} + \beta \mathbf{B} + \gamma \mathbf{C}$$

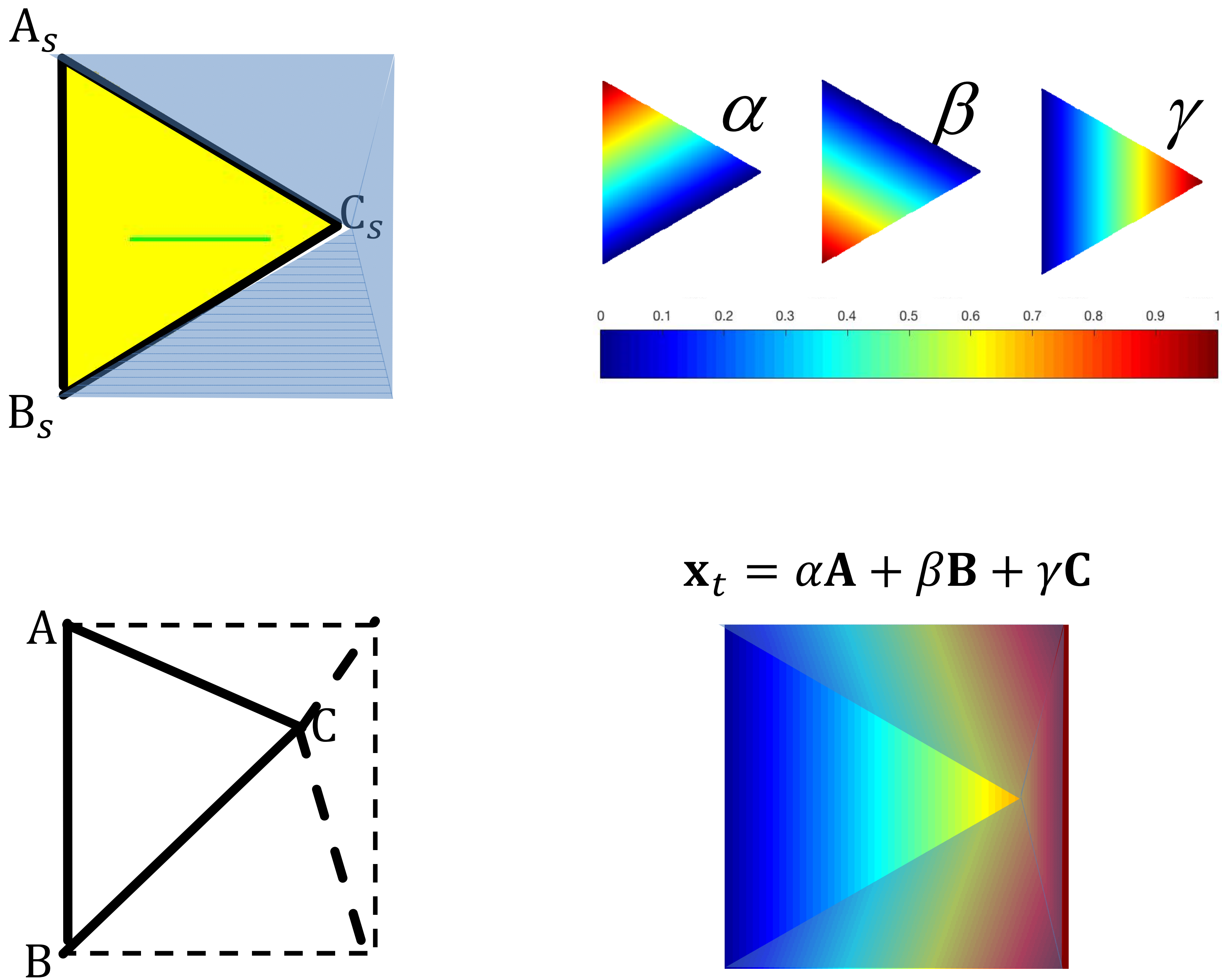


Warping with Barycentric Coordinate



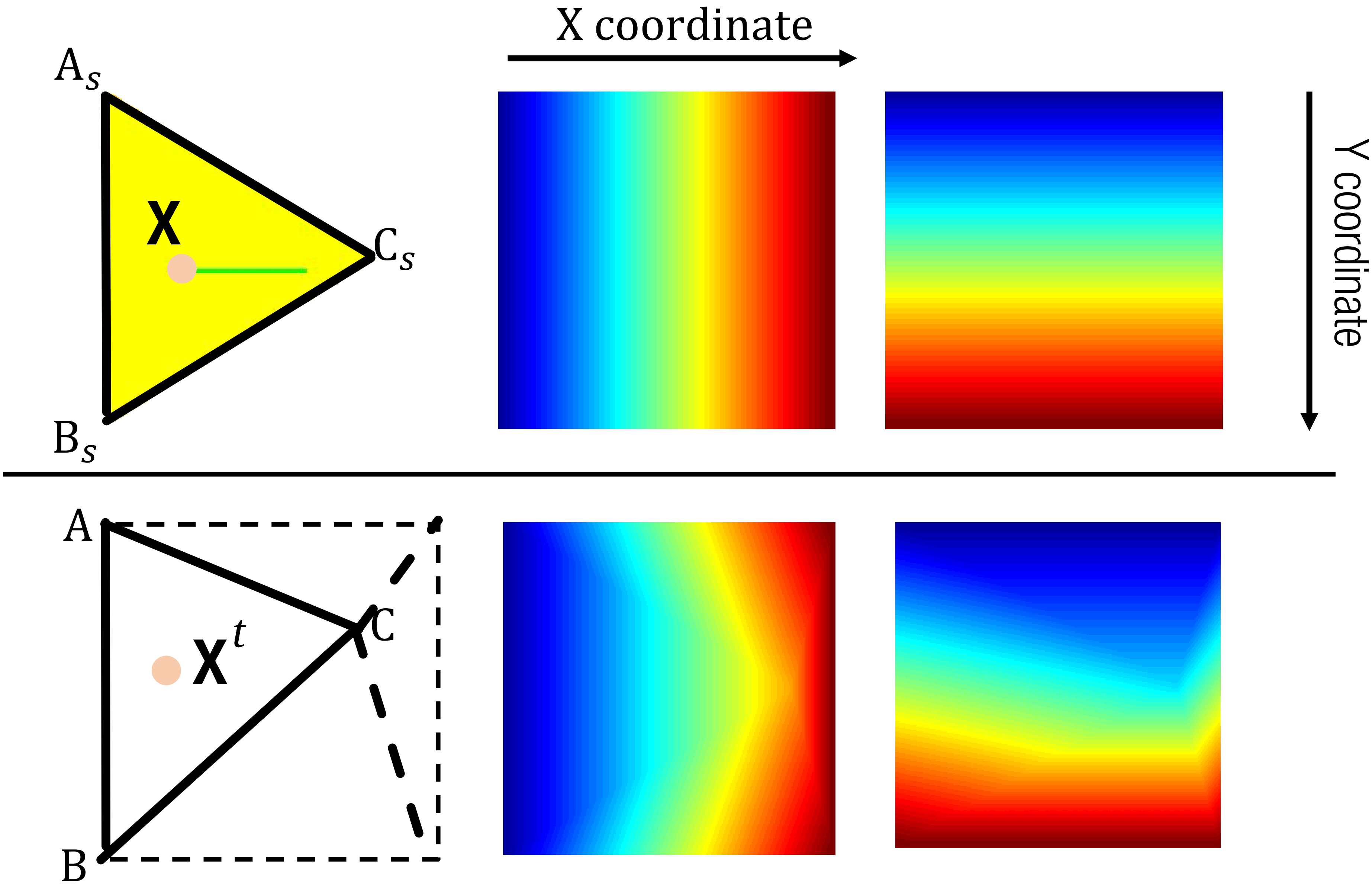
$$\begin{bmatrix} A_x & B_x & C_x \\ A_y & B_y & C_y \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \lambda \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



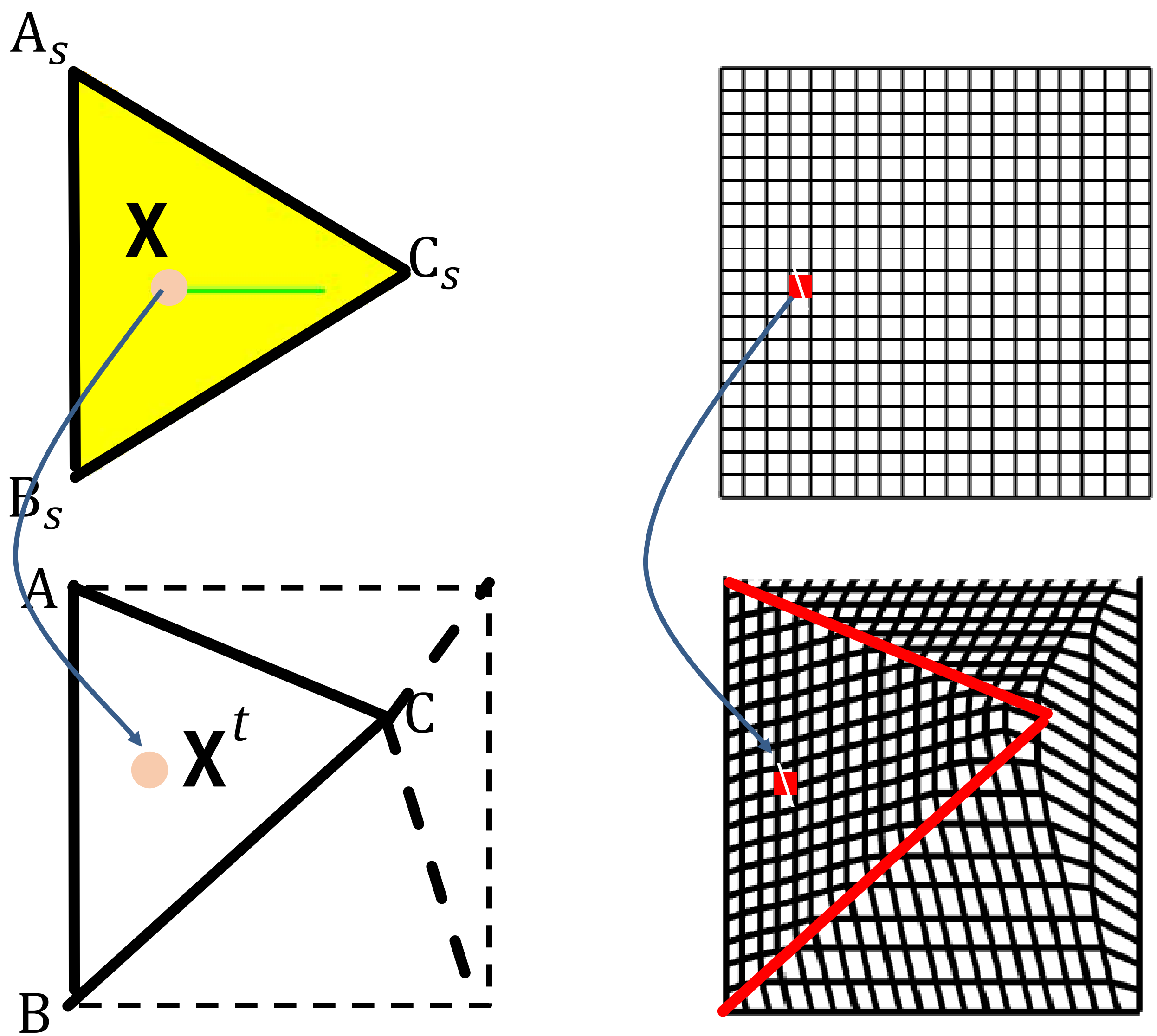


A_s
C_s
B_s
α
β
γ
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
A
C
B
$\mathbf{x}_t = \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$

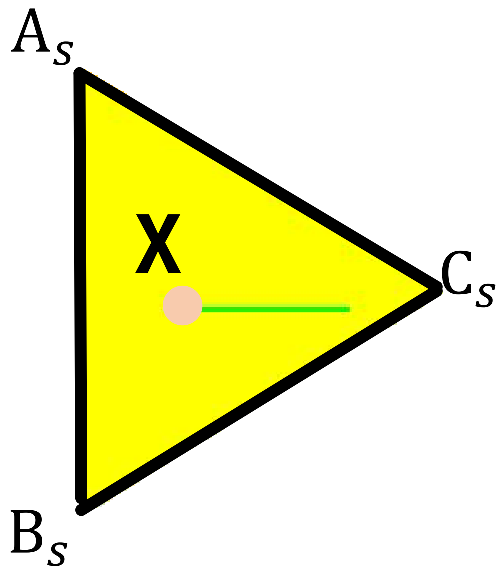
Warping with Barycentric Coordinate

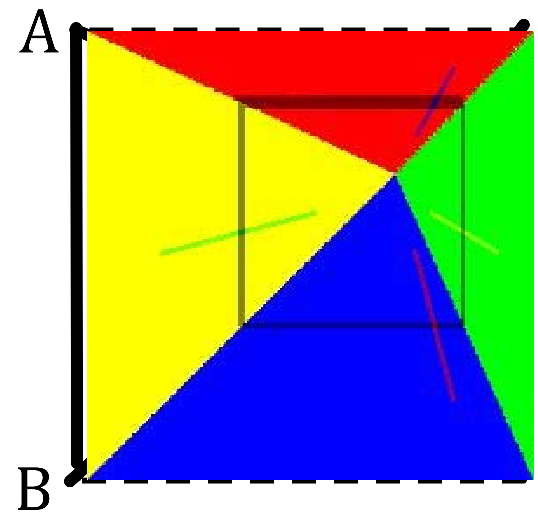


Grids before and after warping

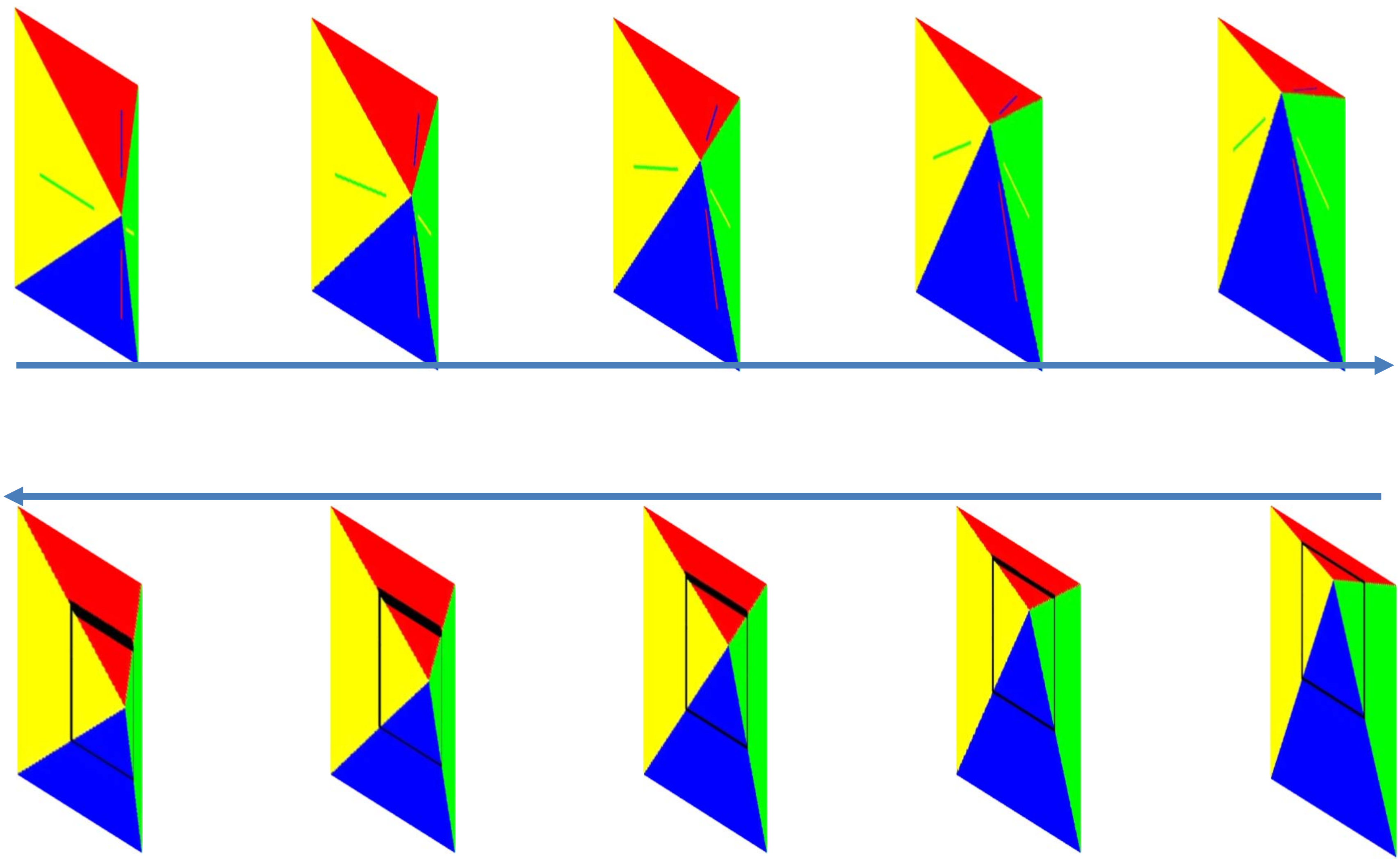


Step 3: Average warped image

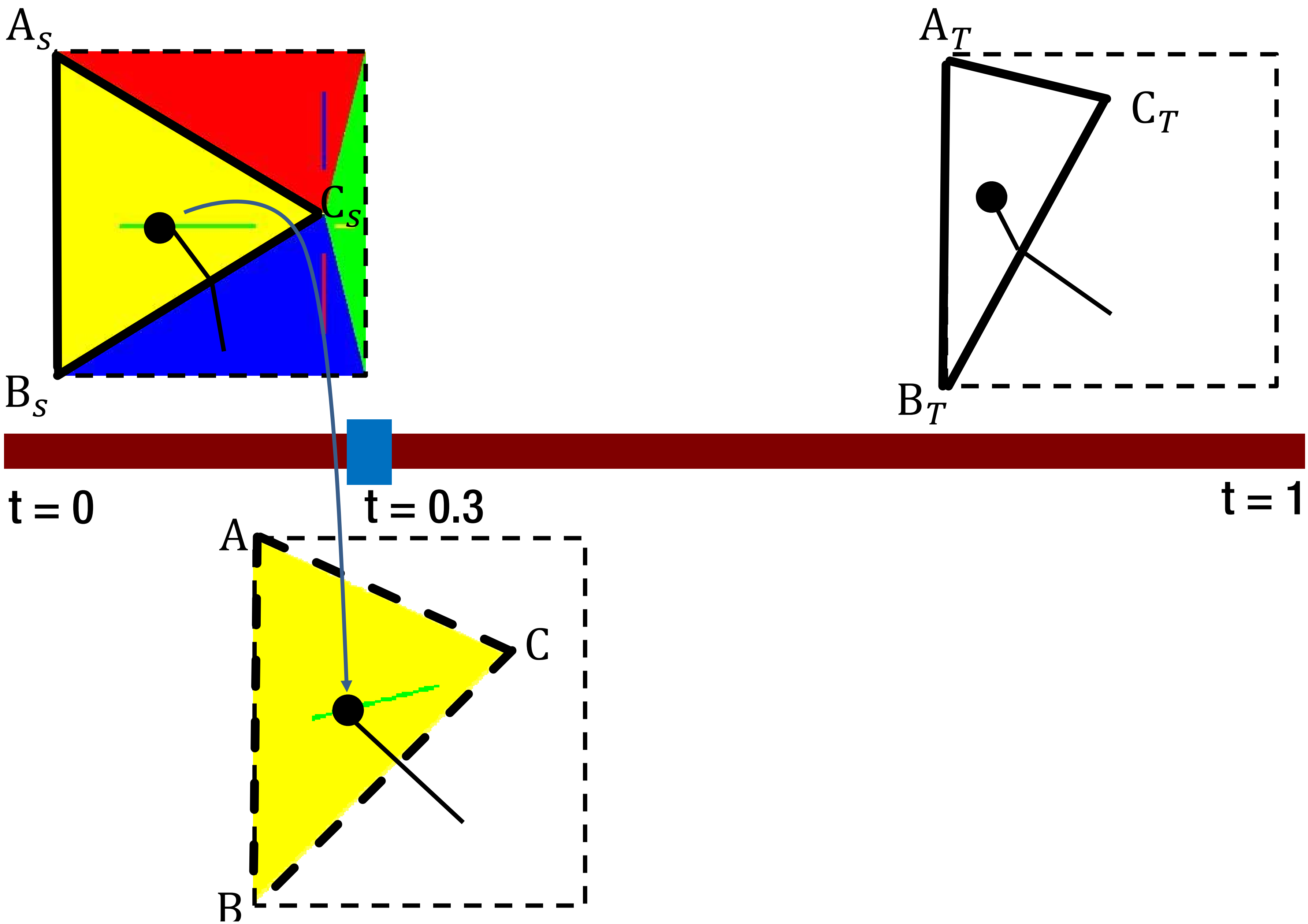




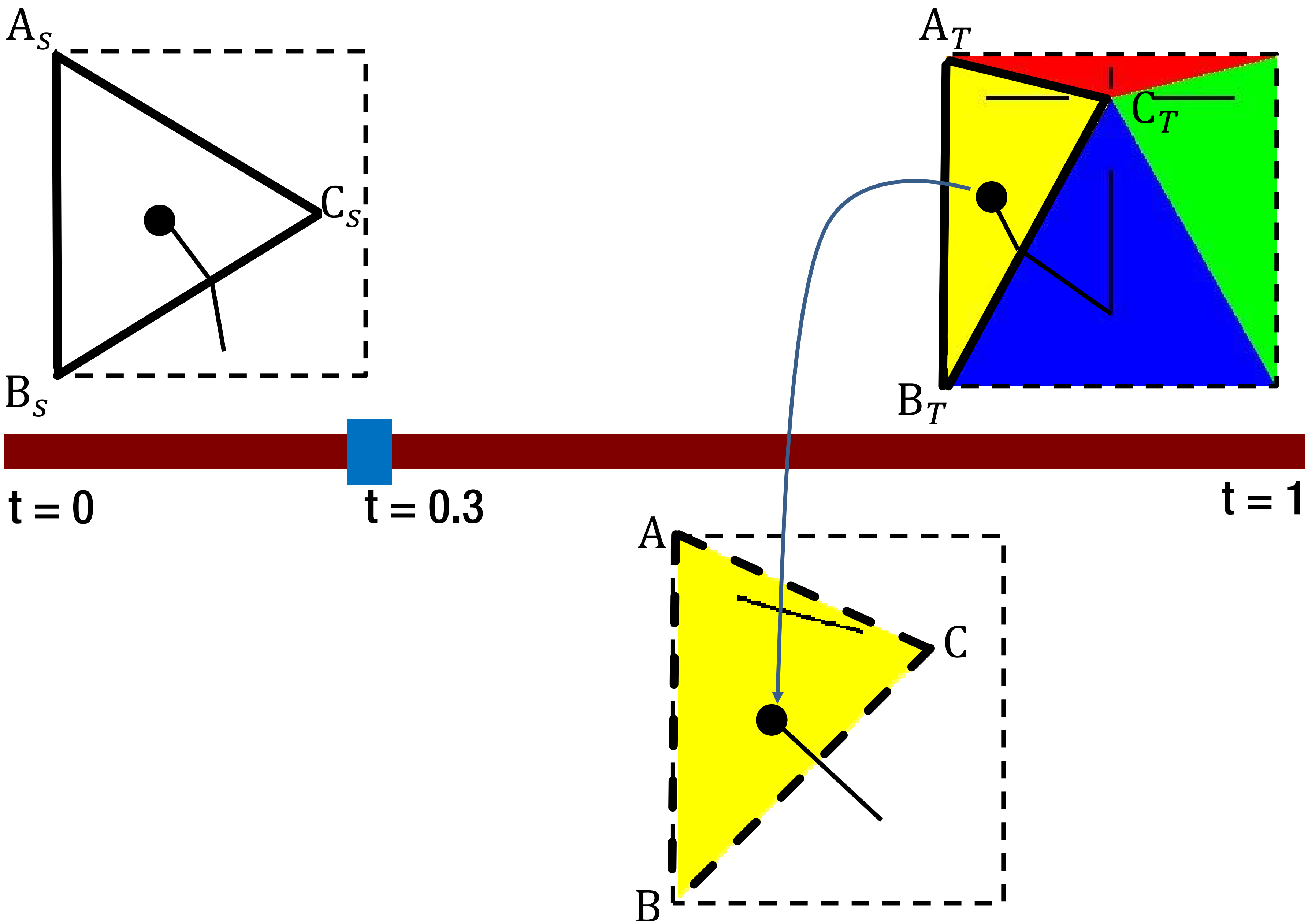
Warping and Cross Dissolve



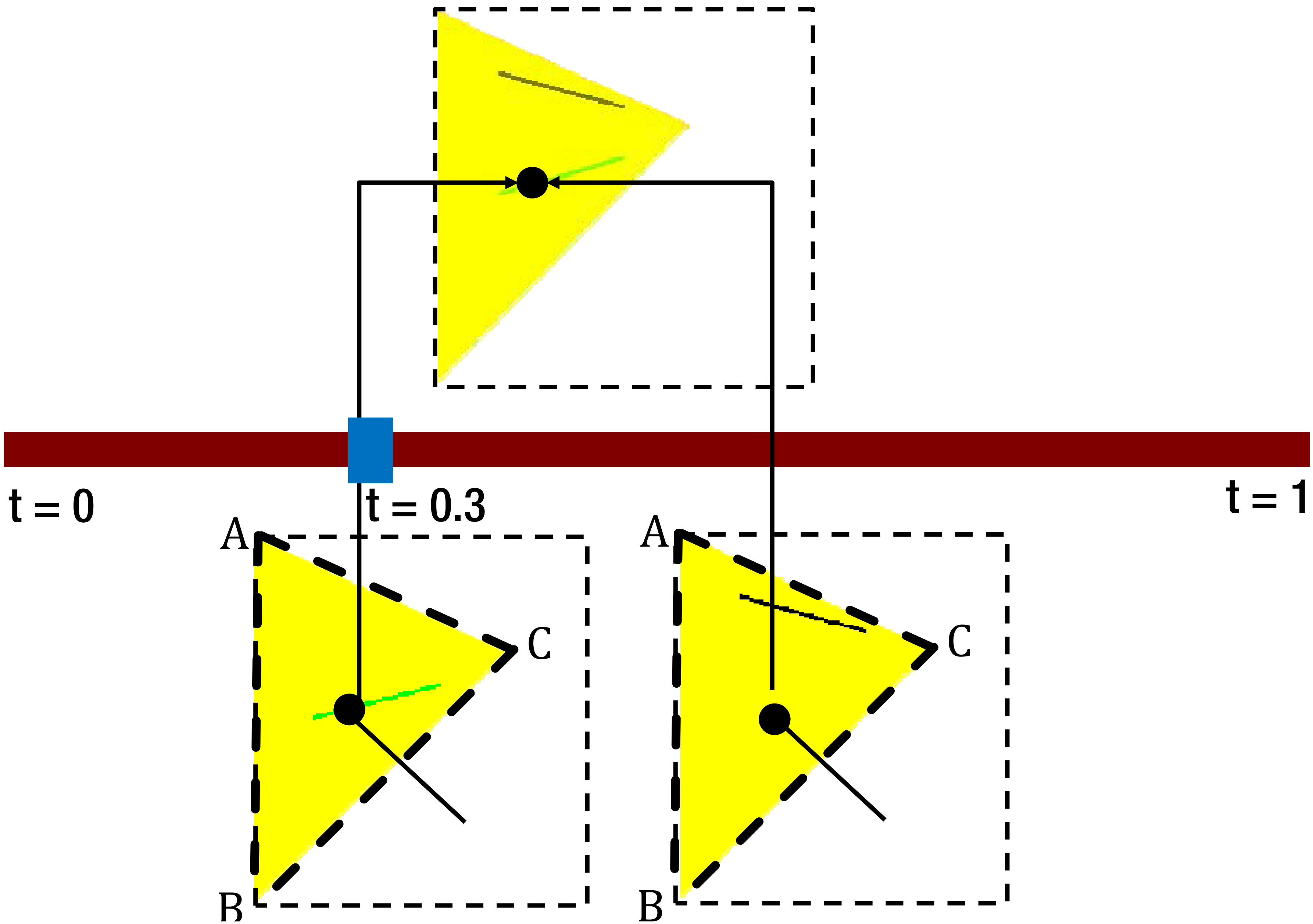
Inverse warping from the source image



Inverse warping from target image



Step 3: averaging warped image



Warping, then cross-dissolve

Morphing procedure:

for every t,

1. Find the average shape
2. Non-parametric warping
3. Find the average color
   - Cross-dissolve the warped images