Report 2

Project 2: Image Homography and Warping

TJ DiMeola
Course: Computer Vision CSCI 581
Instructor: Dr. Hawk Wang
Date: October 19, 2025

Project Objective

In this project, geometric transformation concepts will be applied to carry out image rectification and explore creative applications of image homography. The goal is to understand how 2D transformations can recover fronto-parallel views of planar surfaces and be used in real-world applications such as document scanning or panoramic stitching.

Part 1: Image Homography Estimation

Implement homography estimation and image warping to rectify an input image containing a planar surface (e.g., a document, a poster, or a road sign) captured at an angle.

Tasks:

1. Manually select 4+ point correspondences between the original image and a target rectangular region.

2. Estimate the homography matrix H (e.g., using the DLT algorithm). The Direct Linear Transform (DLT) Algorithm estimates H from point correspondences between two images. A minimum of 4 point correspondences is selected. **Note** that the more points the better, making an overdetermined system, solved with least squares.

Steps:

- Set up the equation:

```
x1 = , y1 = , x2 = , y2 = ,x3 = , y3 = , x4 = , y4 =
x1* = , y1* = , x2* = , y2* = ,x3* = , y3* = , x4* = , y4* =
```

- For each correspondence (x, y) → (x', y'), we can write:

```
x' = (h11*x + h12*y + h13) / (h31*x + h32*y + h33)
y' = (h21*x + h22*y + h23) / (h31*x + h32*y + h33)
```

- Then cross-multiplying to eliminate the denominator:

```
h11*x + h12*y + h13 - h31*x*x' - h32*y*x' - h33*x' = 0
h21*x + h22*y + h23 - h31*x*y' - h32*y*y' - h33*y' = 0
```

- Build matrix A: For n point correspondences, create a 2n × 9 matrix A where each point gives 2 rows:

```
[-x  -y  -1   0   0   0  x*x'  y*x'  x']
[ 0   0   0  -x  -y  -1  x*y'  y*y'  y']
```

The homography vector h = [h11, h12, h13, h21, h22, h23, h31, h32, h33]^T satisfies: Ah = 0

- Solve using SVD: Since there are more equations than unknowns (overdetermined), use Singular Value Decomposition (SVD):

$$A = U \Sigma V^T$$

  The solution h is the last row of V^T (or equivalently, the last column of V) (corresponding to the smallest singular value).

- Reshape the 9-element vector h into the 3×3 homography matrix H.

- Normalize before computing H (optional but better stability):

  - Translate points so their centroid is at origin

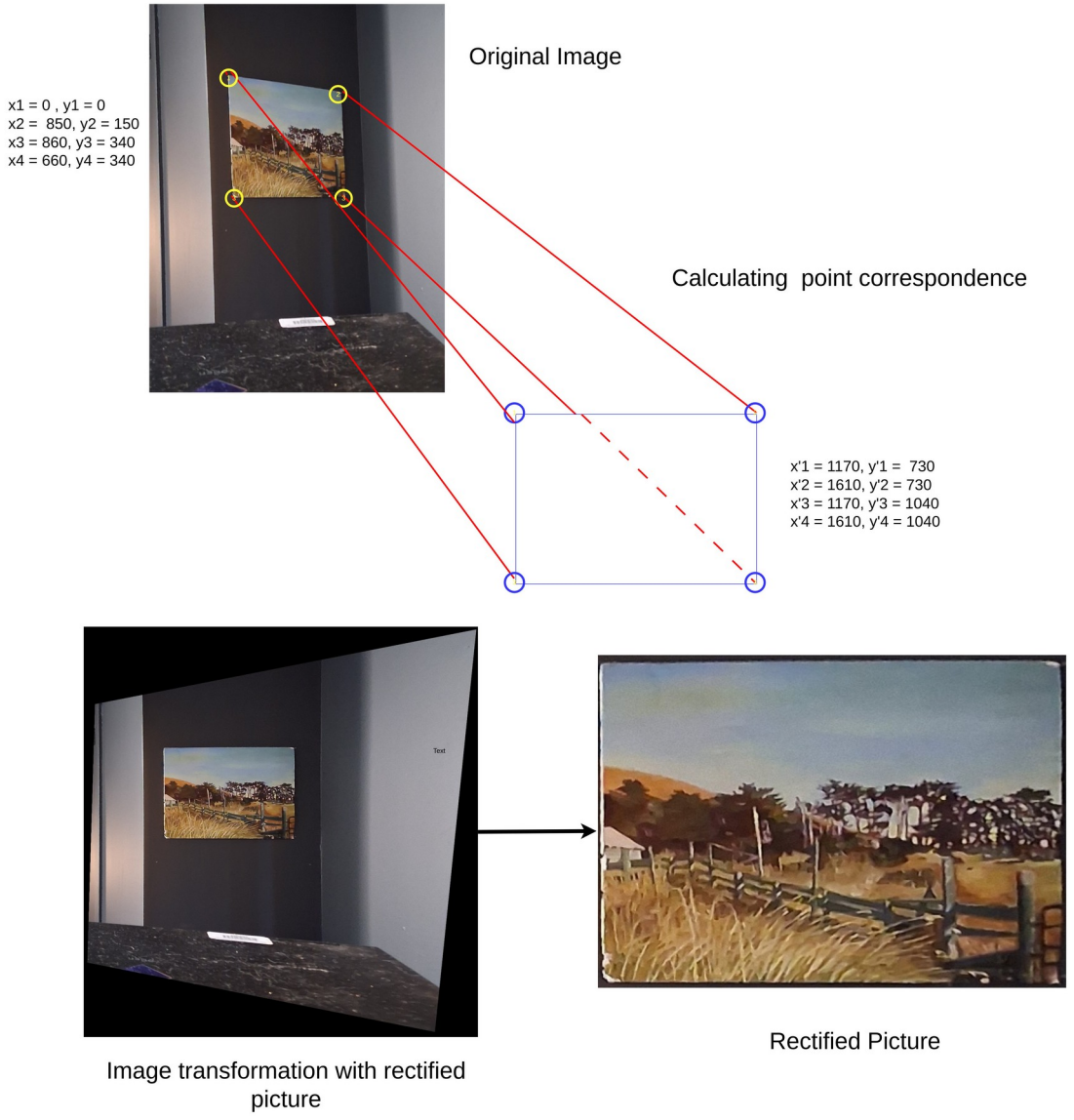
- Scale so average distance from origin is √2
- Compute H normalized
- Denormalize: H = T_dest^(-1) × H_normalized × T_src

- Warp the original image using inverse mapping to obtain a rectified (fronto-parallel) view.
- Visualize before/after results.

Deliverables:

(a) Visualization of correspondences.

(b) Original image and rectified output.



(c) Brief discussion describing how homography enables rectification.

In some ways, homography is intuitive: If you select four (or more) points in one coordinate system in which the points are not orthogonal, and then translate them to another coordinate system in which they *are* orthogonal, you could probably figure out the math (figure out the relationships in space between the points, calculate the angles, etc.). Homography provides a clear “recipe” for this translation, and in particular, the DLT algorithm.

Part 2: Creative Application (Exploration and Extension)

Use your homography implementation creatively to demonstrate a real-world or artistic application.

Requirements:

- Use homography code from Part 1.
- Describing your idea, what worked well, and any challenges.
- You are encouraged to be creative and propose your own ideas.

Experiment: Cover the walls and doors of a hallway with colorful wallpaper.

Tasks:

- Create a utility that would help select points on any section of the wall (sort of like a lasso selection tool in GIMP or Photoshop).
- Take a reasonable photo of the hallway
- For each selected wall section, warp a texture using homography to fit the wall's perspective.
- Repeat for all wall/ceiling/door sections of interest.

Discussion

My original idea was to take photos of each wall of a hallway and stitch them together using homographic control points. However, I could not, within the confined space, obtain sufficient overlap to build the ~50% overlap needed to fully unwrap the walls.

My fallback then was to take a single photo of the hall (top left above) and figure out a way to texture each wall.

I definitely did not want to hand-calculate each control point for every part of the wall, so I used a utility to select as many points as were required to capture the surfaces of walls, ceiling, doors, and floor.

For each surface and set of points, the homography was then calculated and the texture (found online) applied.

This was very interesting because although the points in the hall for each wall were obvious (corners, etc.), the texture points had to be selected blindly. But just as long as they matched the same number of control points as on the walls, this worked pretty well and made really interesting warpage of the textures. I did have to add one section where the wall just got too complex. For fun this was fine. For real, this might have been tough.

Part 3: Warping Comparison

Compare triangular mesh warping and thin-plate spline (TPS) warping.

Tasks:

Implement or use available routines for both warping methods.

Apply both to the same input (e.g., facial landmarks, grid deformation, or a planar surface).

Compare visually and discuss differences in smoothness, continuity, and computational cost.

Deliverables:

Visual comparison (side-by-side outputs).

Discussion: Mesh warping vs TPS warping

- **Mesh Warping** uses Delaunay triangulation to partition the image into non-overlapping triangles, then warps each triangle independently using affine transformations. The triangles are created by connecting control points such that no point lies inside the circumcircle of any triangle - this maximizes the minimum angle of all triangles, avoiding long, skinny triangles that would create artifacts. For each triangle, an affine transformation is carried out:

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

  These 6 degrees of freedom ensure that:

  - Straight lines remain straight
  - Parallel lines remain parallel (within each triangle)
  - Ratios of distances along lines

  The transformation is computed from the three vertex correspondences by:

  - Mapping source triangle vertices to destination triangle vertices
  - Solving for the 2×3 affine matrix
  - Applying the transformation only within that triangle's region

  The computed triangular mesh is overlaid on the image and each triangle acts as a local coordinate system - moving one control point only affects the triangles connected to it, not the entire image. Piecewise the mesh is smooth but there is potential for discontinuities at triangle edges where different affine transforms meet - this is called C^0 continuity. Since each triangle is warped independently huge deformations are unlikely.
- **TPS Warping** is a radial basis function (RBF):

$$\varphi(r) = r^2 \log(r)$$

where r is the distance from a control point. For 2D image warping, the transformation for each coordinate (x or y) is:

$$f(x, y) = a_1 + a_2x + a_3y + \Sigma\, w_i \cdot \varphi(||p_i - p||)$$

where:

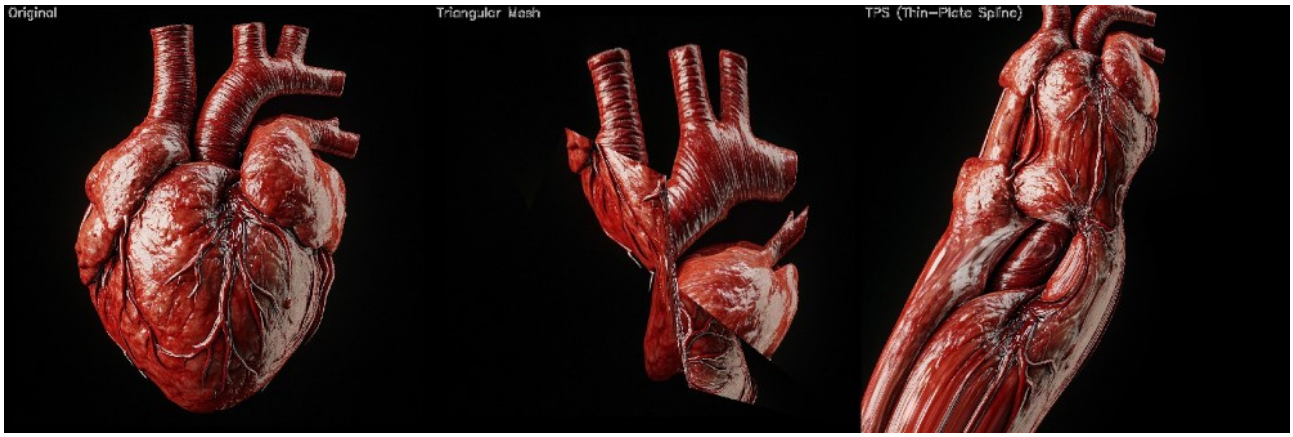
- The first part ($a_1 + a_2x + a_3y$) is an affine transformation (linear/polynomial part)
- The 2nd part is a weighted sum of radial basis functions centered at each control point p_i
- $||p_i - p||$ is the Euclidean distance from point p to control point p_i
- w_i are weights determined by solving a system of equations to satisfy the control point constraints.

Unlike mesh warping, TPS warping, $r^2 \log(r)$, grows slowly, creating smooth deformations, but since they are global, potentially subject to massive deformation.

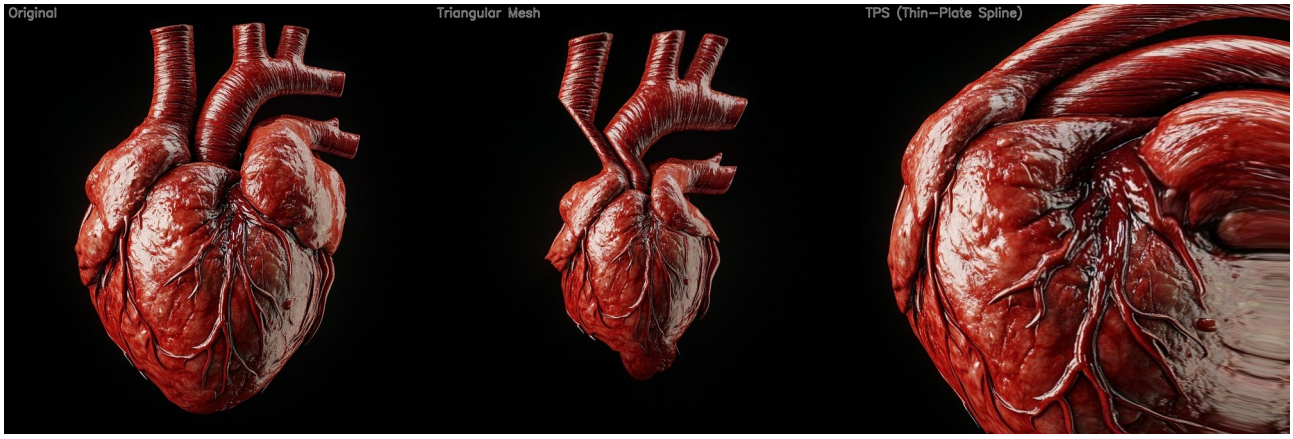
Experimentation: To test both mesh and TPS warping, I decided to see how adding specific control points to an image of a heart, could cause deformation such that the image would be contracted. The overall goal was to see if the heart could look like it was beating.

The first attempt (below), 6 control points were placed around the heart in a irregular circle. In triangular mesh warping, this basically squished the entire heart to nothingness, while TPS turned the heart into a ring.

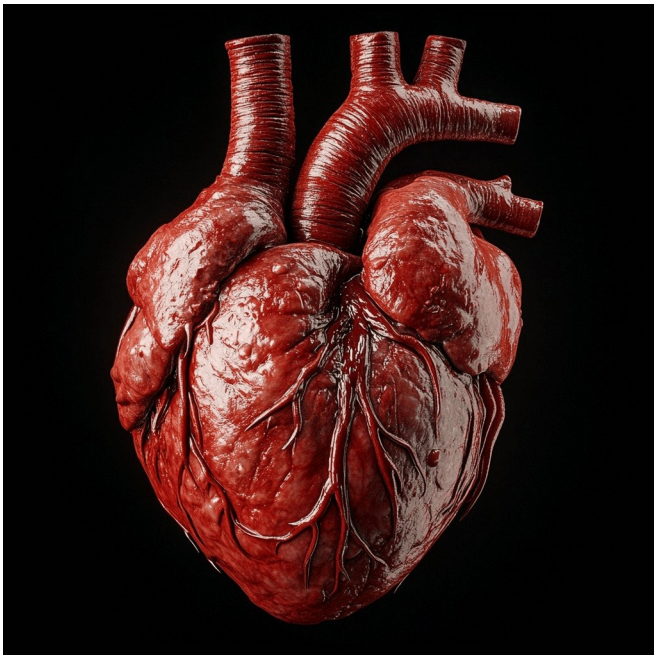
For the second attempt (below), more points (12) to form two rings, but in a zig zag fashion so that every second point would “pull” the part in. This sort of worked, but in both methods the heart was deformed. The point placement needed more precision to avoid extreme deformation.



The third attempt (below), retained 12 points, but care was taken to make the rings accurate, and follow the anatomy of the heart.



The mesh warp method resulted in a heart image that was sufficiently deformed such that it might work well as a “beat” of the heart. By putting the two images together to create a gif, the original idea to mimic a beating heart did work.



Analysis:

For smoothness and continuity, TPS is better. It is fully smooth and globally continuous whereas, triangular mesh leaves visible seams at triangle boundaries. Meshes are piecewise continuous (within triangles), but not across them.

TPS’ global approach means that a pattern that is global (like my ring of points), could end up completely redefining the structure (but smoothly). This global sensitivity extends to outlier control points - a single poorly-placed point can affect the entire image, as demonstrated by the ring effect in attempt 1, above. The mesh approach maintains more local structure so that it is less likely to result in a sudden weird shape.

With respect to computational cost, mesh is faster because each triangle is processed independently, while TPS takes more time in order to carry out global optimization.

The choice depends on the application: triangular mesh is preferable for fast, local deformations with acceptable discontinuities (e.g., real-time applications), while TPS is ideal when smooth, natural-looking deformations are critical (e.g., medical imaging, facial morphing).