Image Warping:
- Restore an image that is geometrically distorted (image does not match the scene view)
- Construct a new image with a new geometry (rotate an image, tilt to a new perspective, etc.)

Basic process
- Map each pixel of the new image to the original image, and resample at the mapped point to estimate the original image value at that point, to store in the new image pixel.
- Both digital images (original and new) are taken as pixel arrays of samples from a single, real image of a scene. Since the real image is not available, the new image must be constructed by resampling the original digital image.

Example
- Create "new" from "original" by scanning over pixel space of "new"
* Resampling - estimates image value at point \((x', y')\) in original image space \(I\) (input image)

- Quality of value depends on type of resampling and support (area of original image involved)
- Nearest neighbor: fast; only if no scale change smaller [minification], may introduce jaggies
- Bilinear (bilinear interpolation): good, if scale change is not minification (needs larger than \(8 \times 8\) support)
- Sinc \(\left( \frac{\sin(x)}{x} \right)\): best, even perfect restoration, but needs a larger support area; size of support basis must increase for minification

Polynomial warp functions - "rubber sheet" warp

\[
x' = f(x, y) = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 + \cdots + a_n x^n y^2
\]

\[
y' = g(x, y) = b_0 + b_1 x + b_2 y + b_3 x^2 + b_4 xy + b_5 y^2 + \cdots + b_n x^n y^2
\]

Take \(f(x, y)\) and \(g(x, y)\) to have the same form.
- This is not required for this approach but simplifies the presentation here.

To define the warp functions it is necessary to

- Establish the terms to be used in \(f\) & \(g\)
  (a full polynomial in 2 variables is not needed, and may be undesirable)

- Collect some observational data on \(x', y'\) and \(x, y\) to be used to estimate the unknown coefficients \(a_k\) and \(b_k\), \(k=0, 1, \ldots, n\)
Control points
- pairs of observational points \( (x_i', y_i') \) and \( (x_i, y_i) \)
  - \( (x_i', y_i') \) are known locations in the new image (fiducial points)
  - \( (x_i, y_i) \) are measured locations in the original image (measured points)
- Objective is to determine coefficients \( a_k \) and \( b_k \) so that
  \[ x_i' = f(x_i, y_i) \] for all \( i = 0, \ldots, p \)
  \[ y_i' = g(x_i, y_i) \] for all \( i = 0, \ldots, p \)
and so that the errors
  \[ x_i' - f(x_i, y_i) \] and \( y_i' - g(x_i, y_i) \)
are minimal under some criterion, such as least squares. \((p+1)\)
- The number of pairs depends on the number of unknown coefficients.

- Separating \( f \) and \( g \) gives 2 sets of observations:
  one for \( f \) and one for \( g \):
  \( \{ x_i', (x_i, y_i) \} \) for linear regression to find the \( a_k \)
  \( \{ y_i', (x_i, y_i) \} \) to find the \( b_k \) coefficients
  for \( i = 0, \ldots, p \) and \( k = 0, \ldots, n \)

Apply linear regression from statistics to determine the coefficient sets \( \{ a_k \} \) \( k = 0, \ldots, n \) and \( \{ b_k \} \) \( k = 0, \ldots, n \):
- both \( f(x, y) \) and \( g(x, y) \) are simple linear equations in the coefficients \( a_k \) and \( b_k \)
- the control points give observations \( (x_i', y_i') \) for the given (fiducial) points \( (x_i, y_i) \)
- all the terms \( a_k x^k y^k \) and \( b_k x^k y^k \) have a constant value \( x^k y^k \) applied to each unknown coefficient
- thus we separate the observational data \( x_i' \) and \( (x_i, y_i) \)
  and \( y_i' \) and \( (x_i, y_i) \) from the unknowns \( a_k \) and \( b_k \)

- See Wikipedia "Linear regression" and "Non-linear regression.
  for in-depth discussions, or review statistics texts on the topic.
Adjustment to determine coefficient values

Define initial values for the two coefficient vectors, for example

\[ \bar{a} = (a_0, a_1, \ldots, a_n)^T = (0, 1, 0, \ldots, 0)^T \] (the identity X transform)

\[ \bar{b} = (b_0, b_1, \ldots, b_n)^T = (0, 0, 1, \ldots, 0)^T \] (the identity Y transform)

Then, for each point \( i = 0, \ldots, p \) define a vector of partial derivatives

\[ \bar{d}_i = (d_i)_j = \left( \frac{\partial f}{\partial a_k}(x_i, y_i), \frac{\partial f}{\partial b_k}(x_i, y_i) \right) \]

Note: for all entries \( g_k, k = 1, \ldots, n \), the partial derivative is evaluated for point \( (x_i, y_i) \).

Note: the vector is identical for both \( f \) and \( g \) as long as \( f \) and \( g \) have the same form.

\[ \frac{\partial f}{\partial a_k}(x_i, y_i) = \frac{\partial g}{\partial b_k}(x_i, y_i) \] for all \( k \) and all \( i \)

Note: since the functions are linear in the parameters, the partial derivatives eliminate all parameters \( (a_k = b_k) \) from the vectors \( \bar{d}_i \). Since the \( d_k \) entries are evaluated at fixed points \( (x_i, y_i) \), the vectors \( \bar{d}_i \) are constant, and need only be evaluated once.

Use the vectors of partial derivatives to define the \((p+1) \times (p+1)\) normal matrix \( S \)

\[ S = (S_{jk}) = \sum_{i=0}^{p} \left( \frac{\partial f}{\partial a_j}(x_i, y_i), \frac{\partial f}{\partial b_k}(x_i, y_i) \right) \bar{d}_i = \sum_{i=0}^{p} \bar{d}_i \bar{d}_i^T \]

These vectors \( \bar{d}_i, i = 0, \ldots, p \), and the normal matrix \( S \) are constants, and need only be calculated once, to set up the adjustment of coefficients \( a_k \) and \( b_k \), \( k = 0, \ldots, n \). These constants are used for both sets of coefficients, for \( f(x, y) \) and \( g(x, y) \).

Note also that the dot products \( \bar{a} \cdot \bar{d}_i \) and \( \bar{b} \cdot \bar{d}_i \) are the function evaluations \( f(x_i, y_i) \) and \( g(x_i, y_i) \), respectively, \( i = 0, \ldots, p \).
Now, iterate to adjust coefficient vectors \( \bar{a} \) and \( \bar{b} \) to improve fit of \( \bar{a} \cdot \bar{d}_i \) to \( x'_i \) and \( \bar{b} \cdot \bar{d}_i \) to \( y'_i \) and converge to minimal (or sufficiently small) error.

Calculate residuals for each point \( i = 0, \ldots, p \):
- In \( x \):
  \[
  r_{x_i} = x'_i - \bar{a} \cdot \bar{d}_i, \quad i = 0, \ldots, p
  \]
- In \( y \):
  \[
  r_{y_i} = y'_i - \bar{b} \cdot \bar{d}_i, \quad i = 0, \ldots, p
  \]

Use the partial derivatives (slope) vectors \( \bar{d}_i \) to find the error contributions of the residuals to the coefficients:
- In \( x \):
  \[
  \frac{\partial x}{\partial \bar{a}} = \begin{pmatrix} c_{x_1} \\ c_{x_2} \\ \vdots \\ c_{x_p} \end{pmatrix} = \sum_{i=0}^{p} r_{x_i} \bar{d}_i \\
  \text{\( \bar{c}_x \) is (n+1) col. vec.}
  \bar{d}_i \text{ is (n+1) col. vec. for each } i = 0, \ldots, p
  \]
- In \( y \):
  \[
  \frac{\partial y}{\partial \bar{b}} = \begin{pmatrix} c_{y_1} \\ c_{y_2} \\ \vdots \\ c_{y_p} \end{pmatrix} = \sum_{i=0}^{p} r_{y_i} \bar{d}_i \\
  \text{\( \bar{c}_y \) is (n+1) col. vec.}
  \]

Adjust the coefficient vectors for the error contributions:
- In \( x \):
  \[
  \bar{a}_{\text{new}} = \bar{a} + \bar{c}_x \bar{c}_x^{-1}
  \]
- In \( y \):
  \[
  \bar{b}_{\text{new}} = \bar{b} + \bar{c}_y \bar{c}_y^{-1}
  \]

Continue the iteration until the residuals \( r_{x_i} \) and \( r_{y_i} \), \( i = 0, \ldots, p \), are sufficiently small by an appropriate measure:
- mean square error in \( x, y \) separately
- mean square error in \( x, y \) together (length of \( \sqrt{(rx, ry)} \))
- etc.

- Fast convergence may happen, but is not guaranteed (guard against "infinite" iteration)
- Better initial values for \( \bar{a} \) and \( \bar{b} \) can speed up convergence.
Problems with "rubber sheet" warps - causes & results

1. Poorly distributed control
   - lack of identifiable point pairs
   - point pairs where one or both can not be measured accurately

2. Insufficient control at or near image boundaries
   - severe distortion near boundary
   - extrapolating control outside boundary usually invalid

Examples of problems
control points

boundary

distortions produced for a horizontal line

- very bad accuracy, but residuals small
- large error at image boundary
- good fit at control points
- not good between control points

Saving characteristics

Polynomials. warps can be much faster to perform
- a warp that a complex, rigorous distortion model
  - use the rigorous model to produce control

Polynomials have a limited number of "wiggles"
- only \((n-1)\) at most for an order \(n\) polynomial
- use the lowest order polynomial possible, for sufficient accuracy
- drop polynomial terms that produce large residuals
Residuals indicate point-fit (but only at control points)

Error measurement
Residuals, but these are only for a small set of points
- eg sum of squares
Use a second set of control points, not used for adjustment