Recap: Resampling

\[ W(\tau) = S + \text{warp} \]

\[ \text{scale} \]

\[ \text{target position} \]

\[ \text{source position} \]

in units of whichever is larger

for each target position

for each source position (in range)

convert source pos \rightarrow target pos

lookup filter value

sum

note: the filter width might need to be scaled.

What about arbitrary warps?

\[ W(R) \Rightarrow IR \]

issues: kernel not constant

more generally \[ TR^2 \Rightarrow IR^2 \]

linear

affine

projective

nonlinear

arbitrary functions

grid warps

scattered data interpolation

other controls
What about this weird resampling

Circle/squares map to odd shapes!

Pre-filter may have a weird shape (circle/square in target)

Algorithm - forward

for each `source` pixel `(x,y)`

find destination position `w(x,y)`

splat in 2D

proble:\s hades

hard to get filter shapes small

need to accumulate

Kernel normalization at end (how many splats hit target)

Algorithm - reverse

for each `target` pixel `(x',y')`

find source position `x, y = w^{-1}(x', y')`

sample source at `x, y` need inverse

- reconstruct (interpolate)

- use larger reconstruction to account for pre-filter

hard: account for pre-filter (if funny shape), need inverse
Useful Case: Morph

Warp to align correspondences then blend works

Image Cloning

1. Just paste = hard edge, easily visible
2. Seam-cuts - find cut where images are the same
3. Feather - fade opacity as get near edges
   But: doesn't adjust to fit (white solid on solid blue background)
4. Match boundaries + interpolate