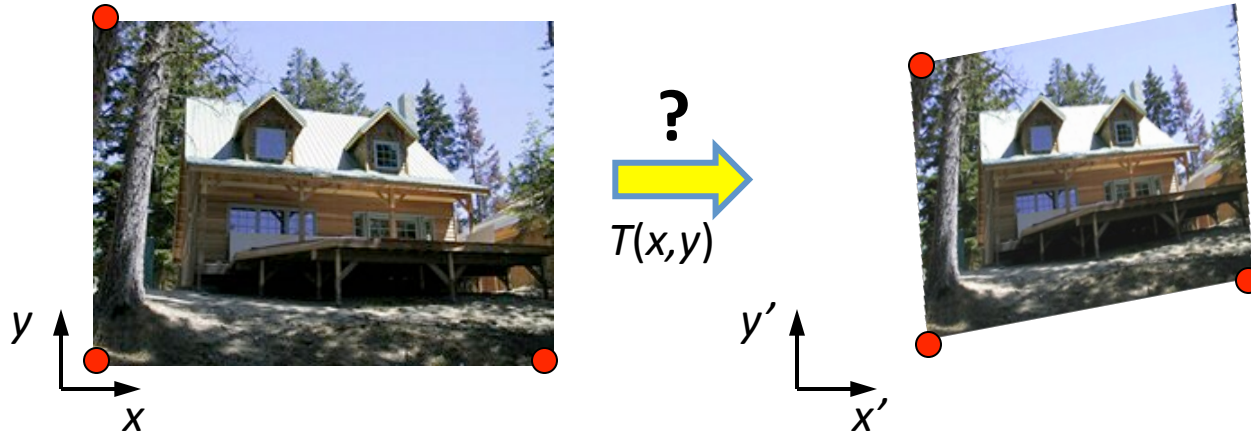


Warping

slides from Alyosha Efros

Affine transformation



Homography transform

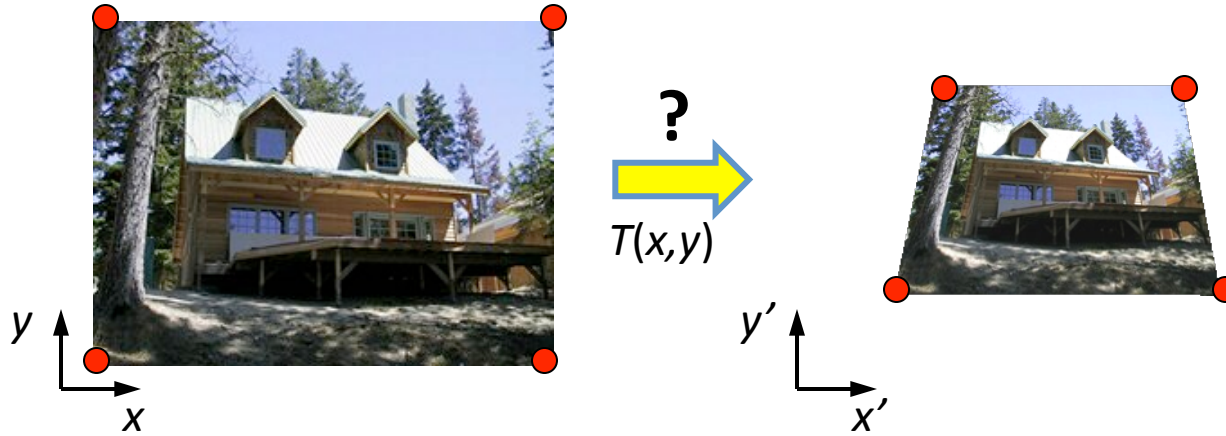
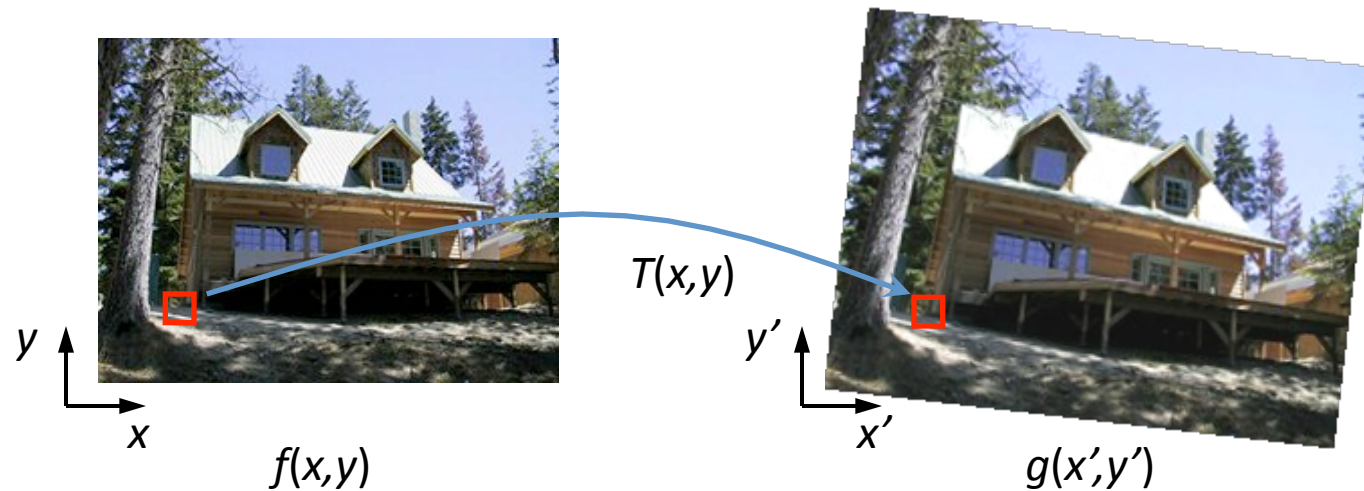
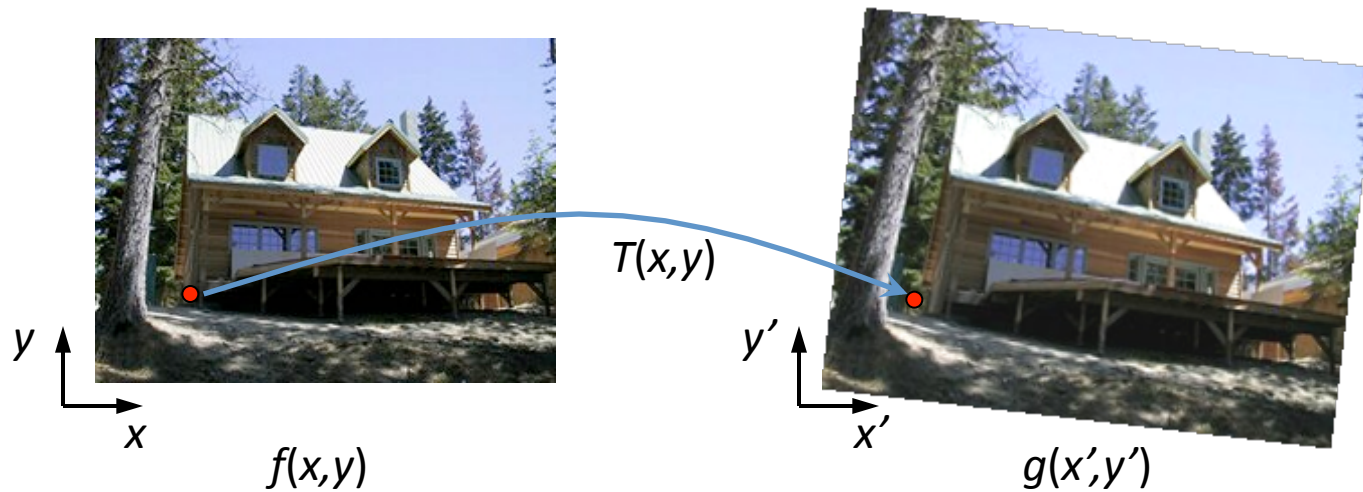


Image warping



- Given a coordinate transform $(x',y') = T(x,y)$ and a source image $f(x,y)$, how do we compute a transformed image $g(x',y') = f(T(x,y))$?

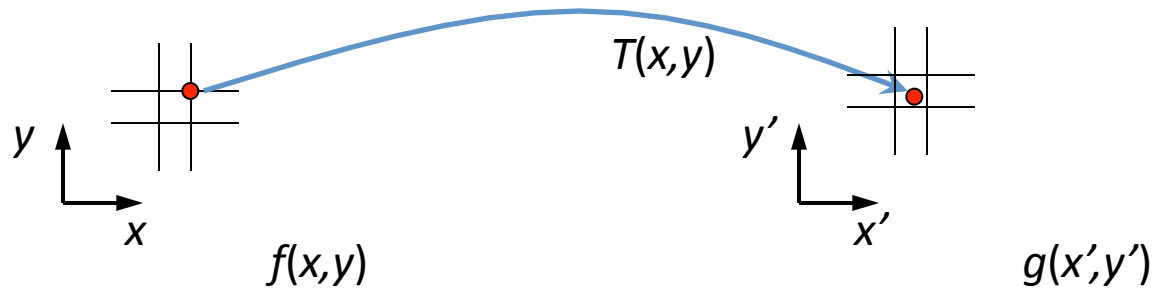
Forward warping



- Send each pixel $f(x,y)$ to its corresponding location $(x',y') = T(x,y)$ in the second image

Q: what if pixel lands “between” two pixels?

Forward warping



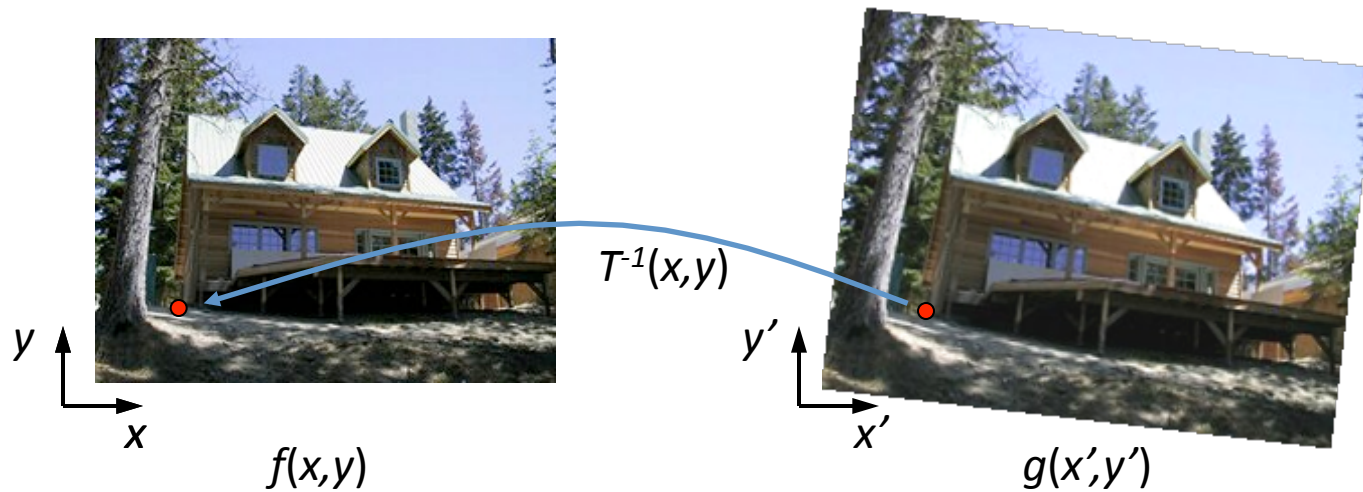
- Send each pixel $f(x, y)$ to its corresponding location $(x', y') = T(x, y)$ in the second image

Q: what if pixel lands “between” two pixels?

A: distribute color among neighboring pixels (x', y')

- Known as “splatting”
- Check out `griddata` in Matlab

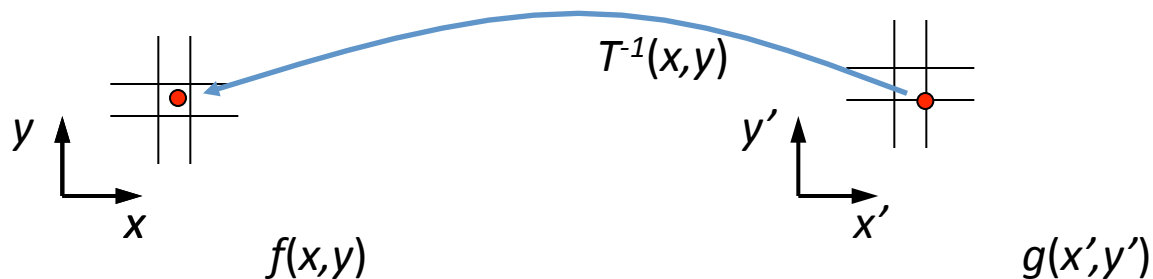
Inverse warping



- Get each pixel $g(x',y')$ from its corresponding location $(x,y) = T^{-1}(x',y')$ in the first image

Q: what if pixel comes from “between” two pixels?

Inverse warping



- Get each pixel $g(x', y')$ from its corresponding location $(x, y) = T^{-1}(x', y')$ in the first image

Q: what if pixel comes from “between” two pixels?

A: *Interpolate* color value from neighbors

- nearest neighbor, bilinear, Gaussian, bicubic
- Check out `interp2` in Matlab

Forward vs. inverse warping

- Q: which is better?
- A: usually inverse—eliminates holes
 - however, it requires an invertible warp function—not always possible...

For Proj 3, you'll use inverse warping
The inverse of a homography is H^{-1}
We'll use Matlab's `interp2` interpolation