

# Normal from single image

D.A. Forsyth,

University of Illinois at Urbana-Champaign

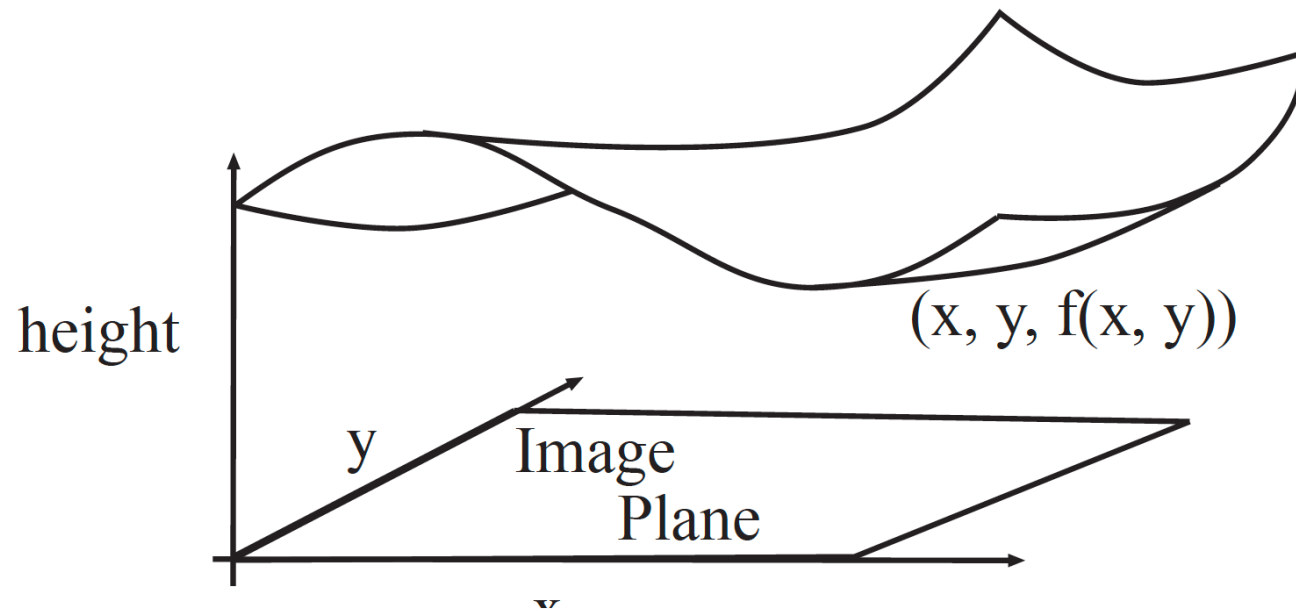
# New recipe, normal case

- Procedure:
  - find many training pairs (image, normal)
  - adjust filters so that
    - $\text{Decode}(\text{Encode}(\text{image}))$  is close to normal
    - on average, over pairs
  - hope that this generalizes to new images
- Result:
  - Single image normal predictor

# Normal from depth

- For training:
  - If you have depth data, fit a plane locally
  - use the plane's normal
- Alternative:
  - geometric reasoning

# Geometrical model (for now)



- At  $(x, y)$  in image, depth is  $f(x, y)$ 
  - We see a surface  $(x, y, f(x, y))$

# Normal from depth

illustrated in Figure 19.4 Now consider the vector from  $(x, y, f(x, y))$  to  $(x + \delta x, y, f(x + \delta x, y))$ ; as  $\delta x$  gets smaller, this vector will get shorter, but

$$\lim_{\delta x \rightarrow 0} \frac{1}{\delta x} (\delta x, 0, f(x + \delta x, y) - f(x, y))^T = \left(1, 0, \frac{\partial f}{\partial x}\right)^T$$

will be tangent to the surface if the derivative exists. Similarly,

$$\left(0, 1, \frac{\partial f}{\partial y}\right)^T$$

is tangent to the surface. Using the very widely established convention of writing

$$p = \frac{\partial f}{\partial x} \text{ and } q = \frac{\partial f}{\partial y}$$

the normal is

$$\frac{1}{\sqrt{1 + p^2 + q^2}} [-p, -q, 1]^T$$

(although you can flip the sign depending on whether you want the normal pointing from the surface to the camera or from the camera to the surface).

# Normal from depth

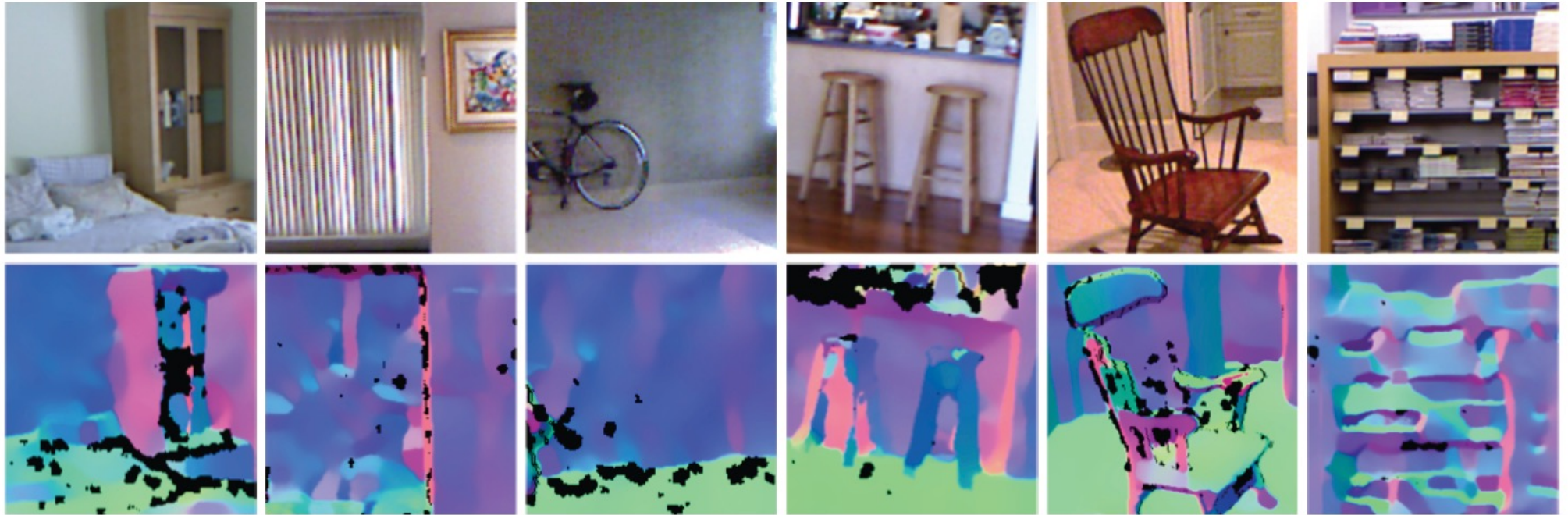
- Notice that derivatives  $\Rightarrow$  noise problems  $\Rightarrow$  smoothing
- Notice also that there is noise in measured depths
  - (property of depth cameras)

# Image and depth



Data example from NYUV2

# Normal estimates from depth data can be shaky



# Loss – building in a confidence score

Normal predictors require care, because the ground truth predicted from depth images can present problems (Figure 19.4). Pushing the normal predictor to reduce prediction error at locations where the ground truth isn't reliable can cause training problems. However, there are manageable procedures to estimate where the ground truth is right (Figure 19.6). Predictions can be notably improved by incorporating such an estimate. Write  $\kappa_i$  for an estimate of predicted normal uncertainty at pixel  $i$ , where  $\kappa_i$  is larger when the predictor is more certain. Write  $\mu_i$  for the predicted value of the normal at  $i$  and  $\mathbf{n}_i$  for the ground truth normal at  $i$ . Then

$$\mathcal{L}(\kappa_i, \mu_i, \mathbf{n}_i) = -\log(1 + \kappa_i^2) + \log(1 + \exp(-\kappa_i\pi)) + \kappa_i \arccos \mu_i \mathbf{n}_i$$

is a loss that encourages the network to reveal when the prediction is uncertain but also encourages the prediction to be like the ground truth.

# Rough story

$$\mathcal{L}(\kappa_i, \mu_i, \mathbf{n}_i) = -\log(1 + \kappa_i^2) + \log(1 + \exp(-\kappa_i \pi)) + \kappa_i \arccos \mu_i \mathbf{n}_i$$

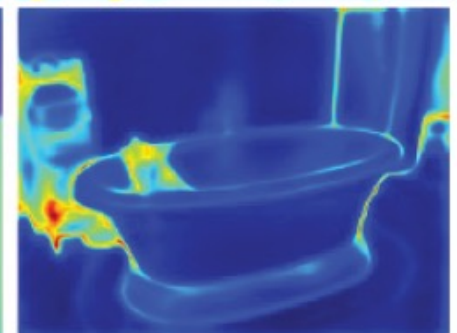
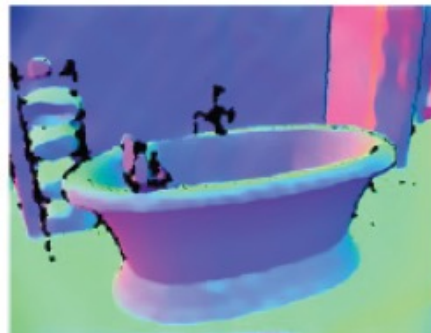
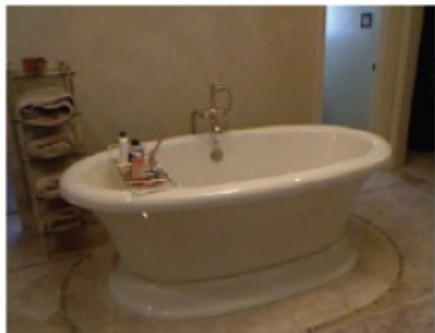
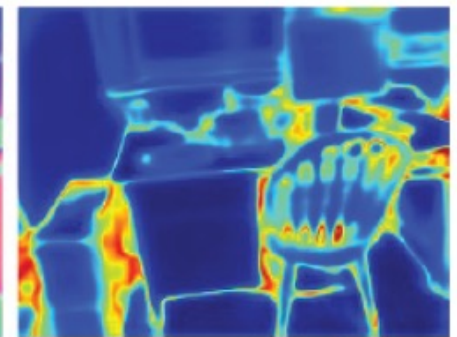
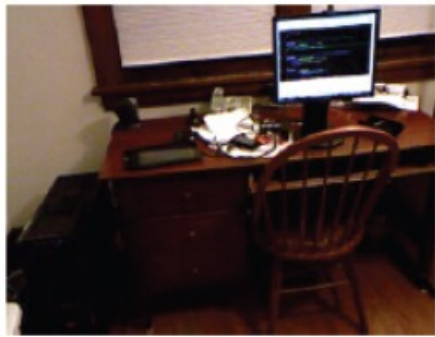
- If normals associated with similar patches are similar
  - $k$  can be big
- If they're not
  - $k$  must be small
- Result: you can estimate  $k$

# What normal predictors predict

What comes out of a normal predictor is usually not the normal. Write  $(n_x, n_y, n_z)^T$  for the components of the unit normal. Most normal predictors report  $(1/2)(n_x + 1, n_y + 1, n_z + 1)^T$ , which has the advantage of being in the range  $[0, 1]$ . If you don't know this, using the normal predictor can be difficult.

# Normal predictors work really well too

NYUv2



Image

GT

Ours - Normal

Ours - Uncertainty

# Evaluating normal predictors

Accuracy in normal predictors is usually evaluated using mean and median of the angle between true and predicted normals, and the percentage of normals where that angle is below some set of thresholds (typically  $11.25^\circ$ ,  $22.5^\circ$  and  $30^\circ$ ).

# SOTA

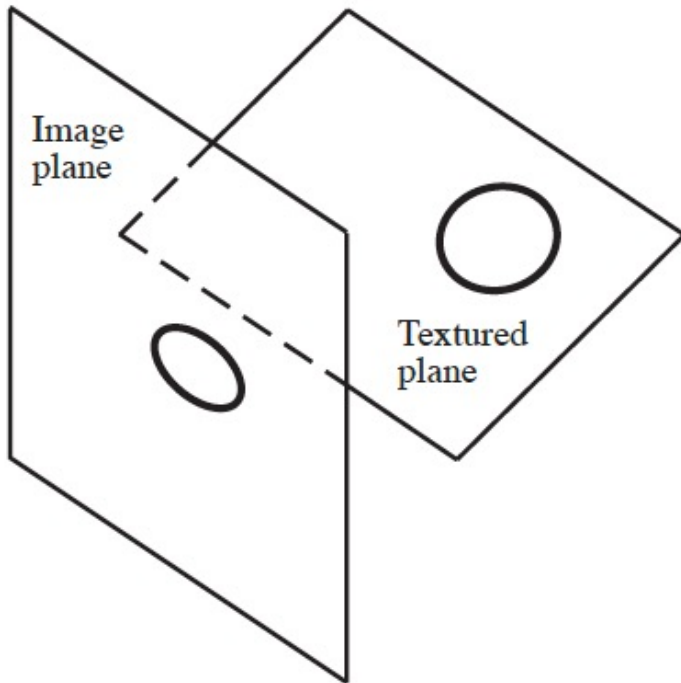
25%, and a frame rate of between 5 and 90 FPS. For normal predictors, you could expect to see mean angular errors of  $15^\circ$  to  $30^\circ$ , depending on model and dataset.

# Why does all this work? I

- The depth and normal at a pixel is usually rather similar to the depth and normal at neighboring pixels. Further, large changes in depth and normal are usually signalled by edges.
- Depth maps aren't all that complicated. Scenes tend to have fairly stylized depth maps, so, for example, rooms are boxes with other boxes in them, and so on. Further, many depth regions are roughly either inclined planes, cylinders or spheres.
- Surface textures and shading offer cues to the orientation of a surface and so to the normal. Shading cues are discussed in Section 15.10. Texture cues are

# Why does all this work? II

- Texture deformations are a cue to normals
  - (archaic term: shape from texture)



# Depth and normal are linked

- You can predict normal from depth
- You can predict depth from normal (next slide)
- Idea:
  - train depth and normal predictors together
    - with a consistency loss
  - improvements manifest

# Depth from normal

If you have a normal estimate that is good enough you can estimate depth up to a constant of integration. Assume the normal predictor predicts a unit normal at each location  $\mathbf{x} = (x, y)^T$ . Write  $[n_x(\mathbf{x}), n_y(\mathbf{x}), n_z(\mathbf{x})]^T$  for that normal. Now

$$\frac{-n_x(\mathbf{x})}{n_z(\mathbf{x})} = \frac{\partial z(\mathbf{x})}{\partial x} = p(\mathbf{x}) \quad \text{and} \quad \frac{-n_y(\mathbf{x})}{n_z(\mathbf{x})} = \frac{\partial z(\mathbf{x})}{\partial y} = q(\mathbf{x})$$

so you should be able to recover the depth map up to a missing constant by integration. This requires some care, because the estimates of the normal might not be exactly right. Recall that

$$\frac{\partial p}{\partial y} = \frac{\partial q}{\partial x}$$

a condition often referred to as *integrability*. An algorithm appears below. Better

# Things to think about...

- 20.3. Could you represent a sphere by  $(x, y, f(x, y))$  for some  $f$ ?
- 20.4. A surface is given by  $(x(s, t), y(s, t), z(s, t))$  where  $s$  and  $t$  are parameters. What expression gives the normal?
- 20.5. Section 20.3.2 says “It is likely that smoothing the depth before differentiation is a good idea; incorporate this into  $\mathcal{D}_x$ .” How would you do this?
- 20.6. Section 20.4.1 has “ $\hat{\mathcal{R}}$  should consist of mostly zeros or very small values, because interpolation is quite good.” Why is this true?