1 INTRODUCTION

contributions

- introduce idea of RGB + single SPAD depth imaging
- ... gesture recognition, face authentication, ... (e.g. "learning to be a depth camera" microsoft paper)
- build prototype, demonstrate results

2 RELATED WORK

Depth Imaging.

- stereo and multiview
- structured illumination and random patterns (kinect, etc.), active stereo
- time of flight (continuous wave and pulsed)
- what we do: like pulsed but much simpler setup; no scanning, no spad array, ...

Monocular Depth Estimation.

- summary of architectures and cost functions: u-net type architecture with reverse huber loss
- what we do: same thing, but augment with global hints (inspired by these approaches, we do ...)

Deep Sensor Fusion. global hints for super-resolution, colorization, depth estimation

- colorization
- david's 2018 paper for depth estimation and denoising (see david's 2019 sig paper for related work)
- what we do: slightly different application

3 THE METHOD

3.1 Image formation model

3.1.1 Single-pixel Time-of-flight Imaging. Laser emits a pulse with shape $g(t)$. That waveform passes through a diffuser which spreads the light evenly in space over some scene, where we express depth $z(x, y)$ as a function of position. To assess the distribution of returning photons from the whole scene, we first consider photons returning from a single location (x, y). The expected number of detected photon events detected from this location in the time interval $(t, t + \delta t)$ is given as

$$\lambda_{x,y}[n] = \int_{n\delta t}^{(n+1)\delta t} (f * g)(t - 2z(x, y)/c)dt. \tag{1}$$

Since the detection of each photon can be described as a Bernoulli random variable, the total number of accumulated photons in this time interval follows a Poisson distribution according to

$$h[n] = \mathcal{P}\left(\sum_{x,y} \alpha_{x,y}\eta\lambda_{x,y}[n] + d\right) \tag{2}$$

where $\alpha_{x,y} = r_{x,y}/z^2_{x,y}$ captures the effect of reflectance r and squared depth falloff $z^2_{x,y}$, η is the detection probability of a photon triggering a SPAD event, $d = \eta a + b$ is the combined effect of albedo a and dark count b.

Pick an example scene, show side-by-side with depth map and histogram of gt depth and histogram from SPAD (to make intuitive)

3.2 Monocular depth estimation with global depth hints

Given a single RGB image $I(x, y)$ and a histogram of photon arrivals $h[n]$ collected as in 2, we seek to reconstruct the ground truth depth map $z(x, y)$. Our method has two parts. First, we initialize our estimate of the depth map from the single RGB image via a monocular depth estimator. Second, we refine this depth map using the captured histogram $h[n]$ via a process we call Differentiable Histogram Matching. Differentiable histogram matching as a tool for post-processing the image to match the depth map to the statistics we capture from the SPAD.

3.2.1 Initialization via CNN. Convoluational Neural Networks have become increasingly capable of leveraging monocular depth cues from to produce accurate estimates of depth from only a single image. However, any depth estimator reliant on only a single view will be unable resolve the inherent scale ambiguity in the scene resulting from the tradeoff between size of and distance to an object.

3.2.2 Conventional Histogram Matching. Show figure comparing before and after histogram matching a black-and-white image to a given histogram. Show figure comparing the metrics of a depth map before histogram matching to a depth map after histogram matching.

3.2.3 Differentiable Histogram Matching. Given an image and a desired histogram, Unfortunately, the function that maps a grayscale image to its histogram is not differentiable. Figure showing just the differentiable histogram matching part of the model. From Depth Map to per-pixel density: Kernel Density Estimation From per-pixel density to simulated SPAD histogram: The SPAD image formation model. From simulated SPAD histogram to loss function: The Wasserstein Distance and Sinkhorn Iterations

Discuss Wasserstein Distance, Sinkhorn Iterations equation, Kernel Density Estimation "Minimize blah s.t. blah blah"

3.3 "How you actually solve it"

3.4 Implementation Details

For the Monocular Depth Estimator, we use a pretrained version of the the Deep Ordinal Regression Network (DORN) []. We use the implementation of the Sinkhorn Iteration (for calculating the entropy-regularized Wasserstein Loss) from **??**uturi et. al. We update the depth map using gradient descent. Everything is implemented in PyTorch.

Table with hyperparameters (defer to supplementary info?) - Kernel Density Estimation (sigma) - Sinkhorn Iterations (maximum number, epsilon tolerance) - Gradient Descent (learning rate, epsilon tolerance, max iterations)

4 HANDS AND FACES

5 ASSESSMENT

5.1 Implementation

hardware, calibration, etc.

5.2 Results

6 DISCUSSION