IMAGES AND DEPTH FOR HIGH RESOLUTION, LOW-LATENCY SENSING AND SECURITY APPLICATIONS

BY

JIAJUN LU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2018

Urbana, Illinois

Doctoral Committee:

Professor David Forsyth, Chair
Professor Derek Hoiem
Professor Svetlana Lazebnik
Dr. Andy Wilson, Microsoft Reseach

ABSTRACT

The thesis focuses on using images and depths for high resolution, low latency sensing, and then using these sensing techniques to build security applications. First, we introduce the usefulness of high quality depth sensing, and the difficulty to acquire such depth stream via pure hardware approach. Then, we propose our sensor fusion approach, which combines depth camera and color camera. Chapter 2 puts forward a low cost approach to use a high spatial resolution color stream to help aggressively increase the spatial resolution of the depth stream. Continuing this direction, Chapter 3 proposes to use optical flow to forward warp the depth stream according to a high frequency, low latency CMOS color stream. The warping can create a high frequency, low latency depth stream. In both Chapter 2 and Chapter 3, we show that the improved depth sensing can benefit lots of applications. In Chapter 4, we propose a SafetyNet, which can reliably detecting and rejecting adversarial examples. With the revolutionary SafetyNet architecture and the advanced depth sensing, we can reliably prove to users whether a picture of a scene is real or not. In sum, the thesis focuses on improving sensing technologies and building vision and security applications around the sensing technologies.

To my PhD advisor, for his advice and support.
To my doctoral committee, for their help and support.
To my wife and parents, for their love and support.

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor Prof. David Forsyth for the continuous support of my Ph.D. study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Derek Hoiem, Prof. Svetlana Lazebnik, and Dr. Andy Wilson, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

My sincere thanks also goes to Dr. Nathan Carr, Dr. Kalyan Sunkavalli, Dr. Sunil Hadap and Dr. Matei Stroila, who provided me an opportunity to join their team as intern, and who gave access to the laboratory and research facilities.

I thank my fellow labmates in for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last five years.

Last, but not the least, I would like to thank my family: my parents and to my brothers and sister for supporting me spiritually throughout writing this thesis and my life in general.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

In recent decades, tons of various computer vision applications become possible. They include gesture recognitions, projection mapping, objects tracking, automated robotic systems, security applications and so on. However, there are still many problems for the current sensing systems. It is relatively cheap to acquire high resolution and low latency color stream, and the depth stream could significantly improve the comprehensive performance of all computer vision applications. However, the current hardware industry cannot create devices that can capture high resolution and low latency depth stream. On one hand, the high resolution color with depth sensing could greatly improve the accuracy of vision systems, especially artificial intelligence system. On the other hand, the low latency color with depth sensing could greatly improve the interaction experiences. The sensing improvements could also help various security applications. Photos with only RGB color information provides limited information and can easily be faked. Deep neural networks are not fully understood by researchers. Recent works about adversarial examples of DNN make all neural networks extremely insecure. Attackers only need to apply an imperceivable universal perturbations onto the image, and all the sophisticated neural network systems will crash. The problem becomes worse as these perturbations can be reliably applied in physical world. For example, by adding imperceivable perturbations to the stop sign, self-driving car can detect that as a go faster sign. My thesis works on techniques to enable high resolution and low latency sensing and build security applications of vision systems based on such sensing.

1.1 MOTIVATION

High resolution and low latency sensing with images and depth can greatly improve the current vision systems, and make them perform better in practical scenarios. Software single approaches in vision systems have been thoroughly investigated, and there is little chance to build sensing that have 10× advances. The combined hardware and software approaches to deal with the sensing problems can potentially greatly push forward the 10× advances. Sensor fusion is a wise place to start the revolution with limited budgets. RGBD data streams can greatly increase the easiness of interactions, and increase the security of vision systems. However, the current depth cameras suffer from low spatial & temporal resolution and long latency.

With stronger sensors and better data inputs, my thesis then begins to deal with problems of security. Safety first applies everywhere including vision systems, and as vision systems

being used in more important areas such as self-driving cars, safety definitely becomes one of the most important factor for industrializing AI techniques. My work focuses on dealing with various attacks to AI systems by using sensor fusion and our proposed SafetyNet.

There are various ways to achieve the goal of improved sensing and more secure system. For example, it might be possible to involve more human efforts to ensure the system security. In my thesis, automatic technical approaches are proposed to handle these problems, which greatly reduce the cost and improve the performance. The thesis focuses on both hardware and software approaches, including sensor fusion and SafetyNet. I will introduce them one by one in details.

1.2 HIGH SPATIAL RESOLUTION DEPTH SENSING

Recent work in HCI has demonstrated a variety of potential applications for depth sensors on mobile devices. Jones et al. [1] demonstrated that mobile depth sensors could enable fluid free-space gesture interactions. Sodhi et al. [2] presented BeThere, a proof-of-concept system designed to explore 3D depth input for mobile collaborative interactions in augmented reality. There are several such sensors in production. However, the relatively high power consumption of current depth sensors even at relatively low resolutions presents major difficulties in making these applications practical. Besides that, further improving the resolution of current depth sensors has high cost. The limitation of power consumption and cost also applies to Lidar system, which begins to be widely used in autonomous vehicles. In contrast, high resolution RGB images are comparatively cheap.

Therefore, there are two compelling reasons to reconstruct high resolution depth from low resolution samples. First, it could allow current sensors to be operated at lower power consumption, and thus make it practical to use them on mobile devices. Second, it could also provide resolution that current sensor systems cannot. It is cheap in money and in power to obtain high resolution image frames registered to the depth sensor, so it is natural to combine upsampling with image or video information.

The thesis describes a method to reconstruct high-resolution depth measurements accurately from sparse scattered samples. We propose to exploit image (resp. video) data obtained at the same time as the depth samples to produce a spatial model that governs our smoothing of depth samples. Our method segments an image with the guidance of sparse depth samples, then uses novel smoothing methods to reconstruct depth within each segment. For video, we use space-time segments and optic flow methods to move time-stamped samples in space.

Our reconstruction outperforms state of the art methods, and could help the improvements

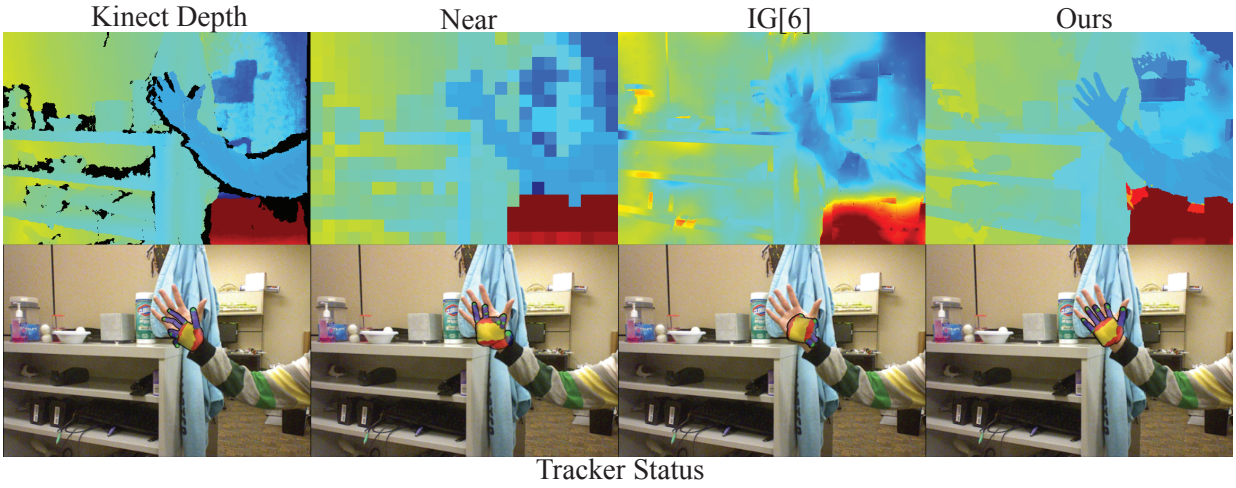

Figure 1.1: Our reconstruction methods support hand tracking from aggressively subsampled depth maps. The first row is depth map and the second row is the status of forth hand tracker [3] [4] (it takes both color image and depth image to generate the status of the hand tracker). In this example at the same moment with different depths, Kinect depth creates difficulty for the hand tracker, nearest neighbour based depth upsample and image guided depth upsampling [5] make the hand tracker fail (because the low quality depth map provides limited information and is even misleading), while our upsampled depth enables the hand tracker to function smoothly.

of various applications. The constructed depth could improve object insertion interaction experiences and results, which requires depth estimation during the insertion process. Hand tracking is an important application of Kinect which is sensitive to the quality of depth estimates. Our estimation could generate significantly better hand tracking results with forth hand tracker [3] [4], refer to Figure 1.1.

1.3 LOW LATENCY DEPTH SENSING

Consumer depth cameras such as Microsoft Kinect have been useful in various HCI applications. But the frame rate (30Hz) and high latency (at 60-80 milliseconds) of such cameras can limit their use in interactive applications sensitive to lag and framerate. Low frame rates can complicate tracking, particularly for fast moving, flexible objects like hands. High latency during interaction can confuse or even nauseate users, particularly in AR and VR applications, or when overlaying graphics on physical objects that move unpredictably.

In the absence of commercially available high frame, low latency depth sensors, we explore solutions involving commodity depth sensors and commonly available color cameras. We demonstrate a hybrid high frame rate depth (Hybrid HFR Depth) camera solution,

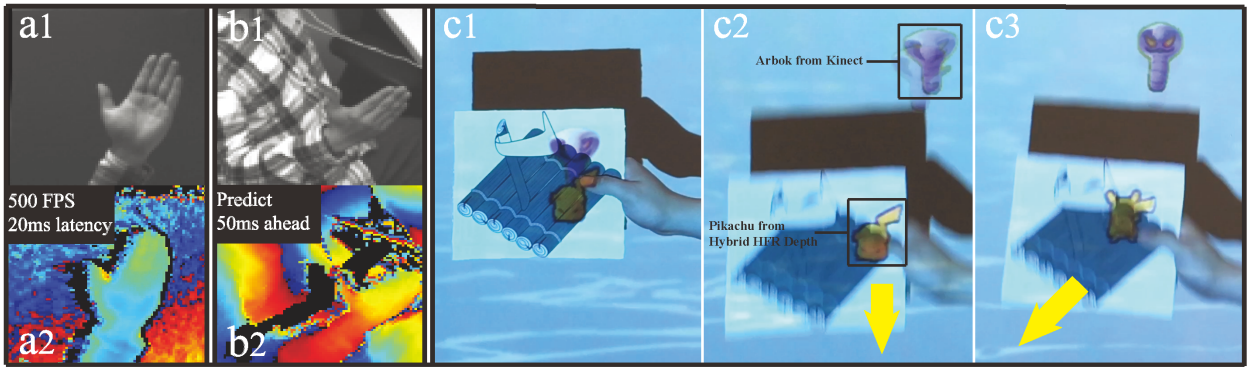

Figure 1.2: Hybrid HFR Depth is a high frame rate, low latency, configurable depth camera solution built from a Kinect and acolor camera. Depth output is 500 frames per second (maximum) and 20 milliseconds latency in (a1-2),and depth is predicted 50ms into the future in (b1-2). Our high frame rate, low latency depth solution supports applications sensitive to lag and framerate:(c1-3)illustrates interactive projection mapping, where Arbok andPikuchu are projected onto the raft using either standard Kinect depth stream (Arbok) or Hybrid HFR Depth stream (Pikachu) to track the moving hand. (c1)Arbok and Pikachu are projected on the raft when hand is still. (c2) (c3) Arbok is off the raft and Pikachu is on the raft when hand is moving because Kinect depth stream has long latency (the sensed hand position is behind the actual hand position) while Hybrid HFR Depth stream has almost no latency.

which registers a Kinect v2 to a Point Grey Grasshopper 3 camera, a commercially available configurable, high frame rate, high resolution and low latency color camera. With the two cameras aligned, we calculate optical flow over a small region of interest (ROI) of the color camera. The frame rate and latency of CMOS cameras such as the Point Grey camera is related to the size of the ROI, with small ROIs obtaining high frame rate (500Hz) and low latency. We apply the resulting flow field to warp the depth samples in the image plane. Our experiments demonstrate that the errors in measurement resulting from our approach are small. Our approach also allows a small amount of prediction to further reduce latency.

We demonstrate three applications of Hybrid HFR Depth. Our applications are chosen to expose the benefits of high frame rate, and low or negative latency. We show Hybrid HFR Depth tracking a ping pong ball with dense, accurate samples of depth. We use Hybrid HFR Depth to track a fingertip, producing an accurate, dense trace useful for rendering and gesture recognition. Finally, we show that Hybrid HFR Depth is well-suited to interactive projection mapping applications, refer to Figure 1.2. High latency in a projection mapping system can cause projected graphics to slide or shift on fast moving objects. Hybrid HFR Depth's high frame rate and latency reduction allows projection mapping onto a moving hand.

1.4 SECURITY APPLICATIONS

Alice wants to prove to Bob that her photo is real easily (without the intervention of a team of experts), and Bob would have high confidence in the proof. This proof could operate at large scales (i.e. anyone could produce a proof while taking a picture), and automatically. We call this **SceneProof**.

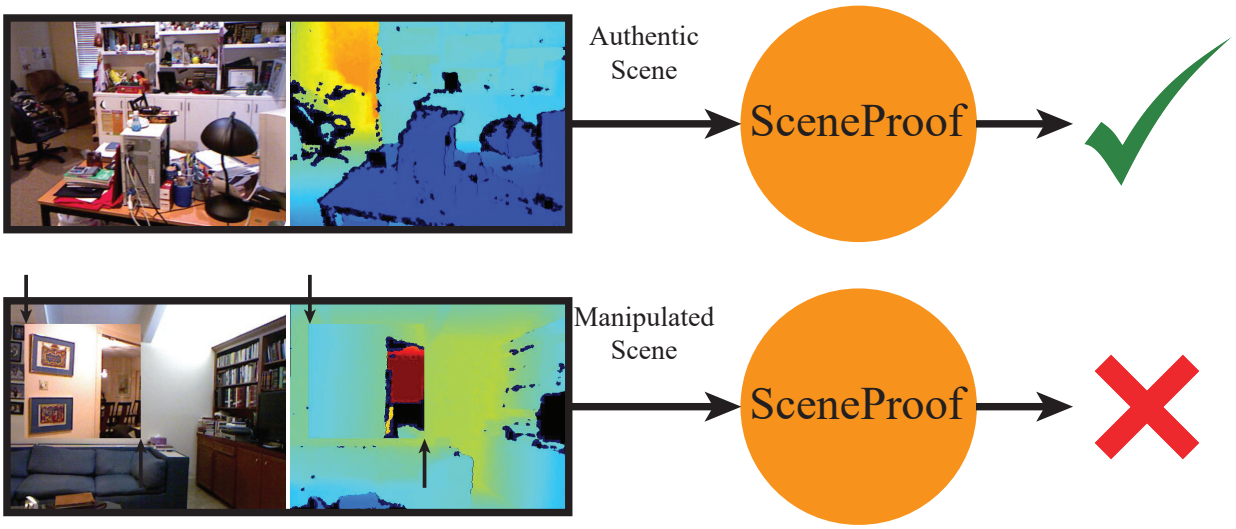

Figure 1.3: SceneProof takes a pair of depth and image, and check whether the scene is authentic or not. It should reject pairs that have been manipulated, such as adversarial examples. In this figure, the first example takes a pair of matched image and depth, and our SceneProof should classify it as authentic. In the second example, a rectangle region of the image and depth is inserted from another source, so our SceneProof should classify it as not authentic.

There are lots of applications that take images with depths. The security applications really require a system that can prove the authentic of the input data, and defend various types of attacks, including adversarial examples, such as face verification. In this thesis, we designed a system that takes depth images together with color images to prove scene authentic, and is very hard to fool, refer to Figure 1.3.

How easy for deep learning based systems to be fooled? Recall that adversarial examples might be used by attackers. Adversarial examples are images with tiny, imperceptible perturbations that fool a classifier into predicting the wrong labels with high confidence. $\mathbf{x}$ denotes the input to some classifier, which is a *natural* example and has label l. A variety of constructions [6, 7, 8, 9] can generate an *adversarial* example $\mathbf{a}(\mathbf{x})$ to make the classifier label it $m \neq l$. This is interesting, because $\|\mathbf{a}(\mathbf{x}) - \mathbf{x}\|_2$ is so small that we would expect $\mathbf{a}(\mathbf{x})$ to be labelled l.

In order to perform the scene proof task, we proposed a new approach SafetyNet. The model architecture consists of the original classifier, and an inspector which looks at the internal state of the later layers in the original classifier. If the inspector declares that an example is adversarial, then the sample is rejected. The inspector in SafetyNet, which needs to be hard to attack, uses an RBF-SVM on binary or ternary codes (activation patterns) to find adversarial examples. In experiments, we demonstrate that SafetyNet can robustly detect type I attacks, and type II attacks on SafetyNet fail.

CHAPTER 2: HIGH SPATIAL RESOLUTION DEPTH SENSING

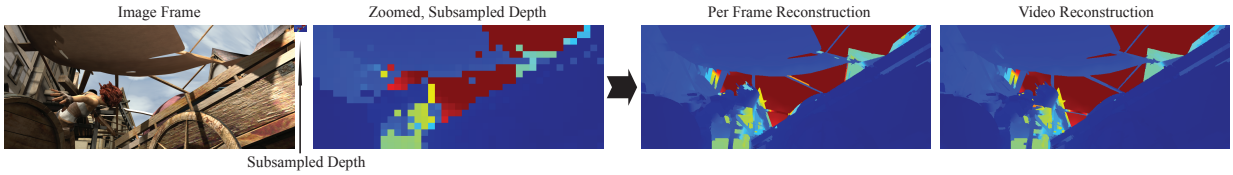

Figure 2.1: We describe a method to reconstruct high resolution depth maps from aggressively subsampled data, using a smoothing method that exploits image segment information to preserve depth boundaries. Our method is evaluated on four different datasets, and produces state of art results. This example shows a case where there is one depth sample per 24×24 block of image pixels (the tiny inset shows the depth map on the same scale as the image). Our method can exploit optic flow and space-time segmentation to produce improved reconstructions for video data.

2.1 OVERVIEW

We describe a method to produce detailed high resolution depth maps from aggressively subsampled depth measurements. Our method fully uses the relationship between image segmentation boundaries and depth boundaries. It uses an image combined with a low resolution depth map. 1) The image is segmented with the guidance of sparse depth samples. 2) Each segment has its depth field reconstructed independently using a novel smoothing method. 3) For videos, time-stamped samples from near frames are incorporated. The thesis shows reconstruction results of super resolution from x4 to x100, while previous methods mainly work on x2 to x16. The method is tested on four different datasets and six video sequences, covering quite different regimes, and it outperforms recent state of the art methods quantitatively and qualitatively. We also demonstrate that depth maps produced by our method can be used by applications such as hand trackers, while depth maps from other methods have problems.

2.2 RELATED WORK

There is a body of work on depth super-resolution methods, exploiting a variety of different approaches.

Markov random field methods can be used to infer high resolution depth from low resolution depth and high resolution intensity images, because the intensity images offer

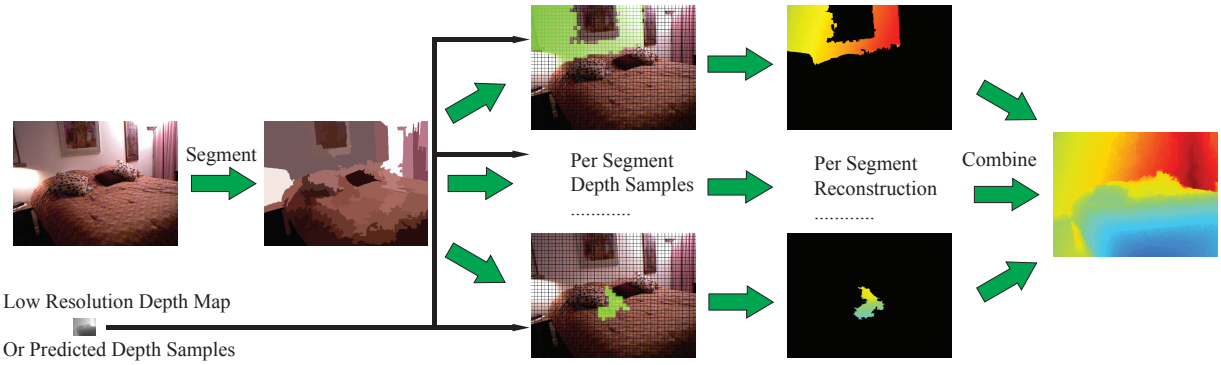

Figure 2.2: Our method first segments the high resolution image, then reconstructs a high resolution depth map for each segment independently using smoothing methods; finally, these reconstructions are composed.

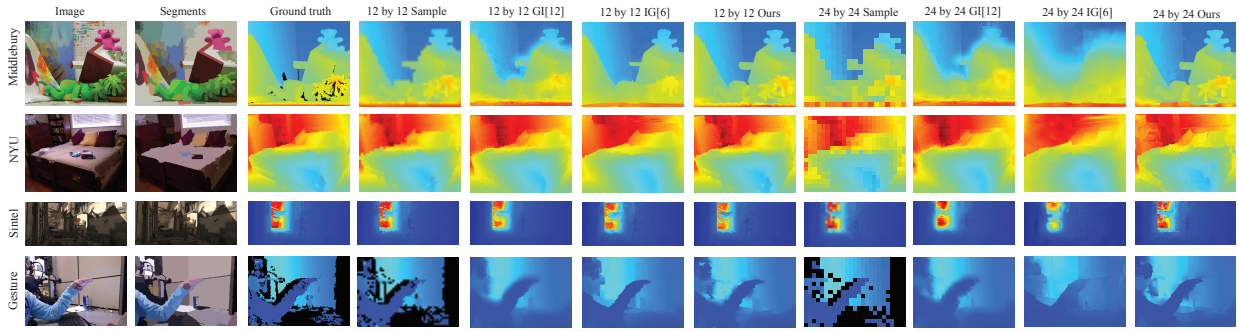

Figure 2.3: Examples of images from each dataset. Our method works better than [5] [10] on both 12 and 24 times. The results of [5] [10] become bad quickly when the upscale ratio increases, while our method is more stable with the upscale ratio.

cues to the location of depth discontinuities. Depth map refinement based on MRF was first explored in [11], extended in [12] with a depth specific data term, and combined with depth from passive stereo in [13]. Park et al. [14]add a non-local means term to their MRF formulation to preserve local structure better and to remove outlliers. Aodha et al. [15] treat depth super-resolution as an MRF labeling problem.

Multiple depth maps can be combined to produce a higher resolution depth map. The Lidarboost approach of Schuon et al. [16] combines depth maps acquired from slightly different viewpoints. The Kinect fusion approach of Izadi et al. [17] produces outstanding results by fusing a sequence of depth maps generated by a tracked Kinect camera into a single 3D representation in real-time. Gudmundsson et al . [18] presented a method for stereo and Time of Flight (ToF) depth map fusion in a dynamic programming approach.

Dictionary methods exploit the dependency between sparse representations of intensity and depth signals over appropriate dictionaries. Mahmoudi et al. [19] first learn a depth

dictionary from noisy samples, then refine and denoise these samples and finally learn an additional dictionary from the denoised samples to inpaint, denoise, and super-resolve projected depth maps from 3D models. [20] and [21] independently learn dictionaries of depth and intensity samples, and model a coupling of the two signal types during the reconstruction phase. In [22], a joint intensity and depth map model is built to recover the co-structure of image and depth.

RGBD image depth refinement methods exploit image shading to improve raw Kinect depth [23, 24]. Such methods estimate the lighting first, and then use shading information to refine the depth map from Kinect output. Complex spatial albedo maps and complex surface material properties present some difficulties for these methods.

Some other methods include [25][10][5] investigates the relationship between images and depth, and no training data is needed. Space does not allow a reasonable review of **image segmentation** or of **optic flow**. We use the method of [26] because it is stable, easy to implement and runs fast. For video data, we use the video segmentation method of [27], which produces space-time segments. For optic flow estimation, we use the method of [28], which is effective and accurate. We expect other methods would apply as well in each case.

Several recent papers have explored regressing depth against a single image (see review in [29]). The best performance to date has been displayed by a deep network [30].

2.3 METHOD

Assume we have an image and scattered depth samples. Our method builds the image segmentation tree, using the scattered depth samples to merge the segments in the tree, then uses smoothing methods to reconstruct depth within each segment independently. For video, we use space-time segments and optic flow methods to move time-stamped samples in space, but otherwise proceed as for static images. Pipeline is in Figure 2.2.

2.3.1 Image Segmentation

The segmenter of [26] is a form of agglomerative cluster, and so produces a tree of region merges (known as a dendrogram in the clustering literature). Each image segmentation is a choice of the region merges in the segmentation tree. Previous methods use pixel colors and cross-validation to do this. In our application, we have depth samples, which is useful in identifying segmentations. So we use depth samples to guide the process of region merging.

For each neighboring region pair, we compute a score measuring the consistency between the depth pairs on the segmentation boundary, where a higher score means better segmen-

tation. Write $\mathbf{c}_i$ for the i'th sample location, $d_f(\mathbf{c}_i, \mathbf{c}_j)$ for the difference in depth at the i'th and j'th sample points, and δ and μ for parameters. Write $\gamma(\mathbf{c}_i, \mathbf{c}_j; \delta)$ for

$$\frac{\min(d_f(\mathbf{c}_i, \mathbf{c}_j), \delta)}{\delta} \tag{2.1}$$

We score the consistency $C_s(\mathbf{c}_i, \mathbf{c}_j)$ of each neighboring sample depth pair using

$$C_s(\mathbf{c}_i, \mathbf{c}_j) = \begin{cases} \mu(e - e^{\gamma(\mathbf{c}_i, \mathbf{c}_j; \delta)}) & \mathbf{c}_i, \mathbf{c}_j \text{ share a segment} \\ e^{\gamma(\mathbf{c}_i, \mathbf{c}_j; \delta)} & \text{otherwise} \end{cases} . \tag{2.2}$$

We use $\mu = 1$ and fixed δ for each dataset. Notice that larger (resp. smaller) μ values would tend to produce fewer (resp. more) segments. For each image, we first build the segmentation tree with fixed number of levels and minimum area sizes for each dataset. Then, we start with the level that contains the largest segments, and recursively for each segment use the consistency score to decide whether go down to the next level of the segmentation tree. The algorithm is robust to segmentation results. This is mainly because the depth samples can fix segmentation problems, especially over-segmentation.

2.3.2 Depth Smoothing

We now have a set of image segments, and a collection of depth samples. For each segment, we will smooth the samples inside the segment to form a dense depth map. By using only the samples inside the segment, we can obtain sharp depth boundaries at image segment boundaries. Once each segment has a depth map, we compute an overall depth map by copying the depths from each segment to the image plane.

We describe two methods of smoothing. Our **simple depth smoothing** algorithm is a form of scattered data interpolation. It helps to understand the smoothing system, is faster than the advanced smoothing, and could easily be implemented in parallel. Our **advanced depth smoothing** algorithm allows some large derivatives of depth. Experimental results show the advanced method is better at dealing with complex depth images, poor segmentation, and noise in depth samples. Preliminary, experiments demonstrated the advanced method have an RMSE approximately 5% better than the sample method and all results reported are for the advanced method.

Simple Depth Smoothing

Write $\mathbf{c}_i$ for the location of the i'th sample, $z_s(\mathbf{c}_i)$ for the value at that sample point, $\mathbf{x}$ for a variable point in the image and $z(\mathbf{x};\mathbf{a})$ for the reconstructed depth function, $\mathbf{a}$ for the parameter vector. We start with a set of radial basis functions centered at each sample point. Write $\phi(\mathbf{x};\mathbf{c}_i) = f(\|\mathbf{x}-\mathbf{c}_i\|)$ for the radial basis function centered at $\mathbf{c}_i$. Write $d_{\max}$ as max influence range, we have

$$f(u) = \max(1 - \frac{u}{d_{\max}}, 0)^2 \tag{2.3}$$

We wish to blend depth estimates from nearby samples at a given point $\mathbf{x}$. So it is natural to work with a set of basis functions that add to one at every point. Define

$$\beta(\mathbf{x};\mathbf{c}_i) = \frac{\phi(\mathbf{x};\mathbf{c}_i)}{\sum_j \phi(\mathbf{x};\mathbf{c}_j)} \tag{2.4}$$

(assuming that $d_{\max}$ is chosen so that at least one ϕ has a non-zero value for all $\mathbf{x}$ in the image).

We assume that there is an initial prior model of depth $z_\pi(\mathbf{x})$. In our experiments, this is usually a zero offset plane, but when the samples are super sparse, using depth maps reconstructed from other methods as frontal plane is useful. However, when images are known to have a particular structure — for example, to be images of rooms as in [31] — it might be useful to have some more complex model. Our depth interpolation is

$$z(\mathbf{x};\mathbf{a}) = \sum_i a_i \beta(\mathbf{x};\mathbf{c}_i) + z_\pi(\mathbf{x}) \tag{2.5}$$

where $\mathbf{a}$ are determined by solving the linear system

$$z(\mathbf{c}_j;\mathbf{a}) = \sum_i a_i \beta(\mathbf{c}_j;\mathbf{c}_i) + z_\pi(\mathbf{c}_j). \tag{2.6}$$

With reasonable choices of $d_{\max}$, this system has full rank, and so the solution yield an interpolation. In our implementation, $d_{\max}$ is twice the average spacing between samples. This means that at any point relatively few ϕ have non-zero values, meaning that evaluating the blending weights is fast.

Advanced Depth Smoothing

There are some difficulties with the simple depth smoothing model. First, it yields an interpolation, which means that noisy depth measurements can seriously disrupt the reconstruction. Second, the basis functions do not encode large depth derivative particularly well. We introduce further basis functions to remedy these problems. At each sample point i we place a unit vector $\mathbf{u}_i$. Then

$$\psi(\mathbf{x}; \mathbf{c}_i, \mathbf{u}_i) = \phi(\mathbf{x}; \mathbf{c}_i) \left(\mathbf{u} \cdot (\mathbf{x} - \mathbf{c}_i)\right) \tag{2.7}$$

is a basis function with value 0 at $\mathbf{x} = \mathbf{c}_i$. The gradient on position $\mathbf{c}_i$ is large, and it can be steered by choice of $\mathbf{u}_i$. Our smoothed depth model becomes

$$z(\mathbf{x}; \mathbf{a}, \mathbf{b}, \mathbf{u}) = \left(\sum_i a_i \beta(\mathbf{x}; \mathbf{c}_i)\right) + \left(\sum_i b_i \psi(\mathbf{x}; \mathbf{c}_i, \mathbf{u}_i)\right) + z_\pi(\mathbf{x}). \tag{2.8}$$

We must now choose values of $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{u}$ to produce a reconstruction. We do so by minimizing an objective function that is a sum of three terms, each capturing a natural requirement of the problem. First, we expect that depth samples have relatively low noise, so we expect E_1 to be small.

$$E_1(\mathbf{a}, \mathbf{b}, \mathbf{u}) = \sum_j \left(z(\mathbf{c}_j; \mathbf{a}, \mathbf{b}, \mathbf{u}) - z_s(\mathbf{c}_j)\right)^2 \tag{2.9}$$

Second, we expect that there are relatively few sharp depth gradients, so that we expect E_2 to be small.

$$E_2(\mathbf{a}, \mathbf{b}, \mathbf{u}) = \| \mathbf{b} \|_1 \tag{2.10}$$

The L_1 norm here encourages most of the b_i to be zero.

Finally, at each grid sampling's grid box center $\mathbf{x}_k$, the depths predicted by any two distinct nearby sample points need to be consistent with one another. Write $\mathcal{N}(\mathbf{x}_k)$ for a neighborhood around $\mathbf{x}_k$. We expect that $E_3(\mathbf{a}, \mathbf{b}, \mathbf{u})$ to be small, which is

$$\sum_k \sum_{c_i, c_j \in \mathcal{N}(x_k)} \begin{pmatrix} a_i \phi(\mathbf{x}_k; \mathbf{c}_i) + b_i \psi(\mathbf{x}_k; \mathbf{c}_i, \mathbf{u}_i) \\ -a_j \phi(\mathbf{x}_k; \mathbf{c}_j) - b_j \psi(\mathbf{x}_k; \mathbf{c}_j, \mathbf{u}_j) \end{pmatrix}^2, \tag{2.11}$$

We choose parameters by optimization, and the weights are chosen by cross-validation.

$$\operatorname{argmin} \mathbf{a}, \mathbf{b}, \mathbf{u} \lambda_1 E_1 + \lambda_2 E_2 + \lambda_3 E_3 \tag{2.12}$$

[log10, RMSE], smaller is better.

Dataset	Middlebury Dataset								NYU Dataset							
Ratio	24×24 times (41,55)		16×16 times (69,82)		12×12 times (92,109)		8×8 times (138,163)		24×24 times (18,24)		16×16 times (27,36)		12×12 times (36,47)		8×8 times (54,71)	
Near	0.0121	0.0366	0.0082	0.0291	0.0056	0.0240	0.0036	0.0189	0.0134	0.1656	0.0111	0.1444	0.0065	0.0977	0.0061	0.0929
Bicubic	0.0139	0.0313	0.0096	0.0246	0.0070	0.0199	0.0047	0.0156	0.0115	0.1333	0.0099	0.1223	0.0055	0.0755	0.0057	0.0780
[10]	n/a	0.0340	n/a	0.0259	n/a	0.0198	n/a	0.0166	n/a	0.1588	n/a	0.1388	n/a	0.1026	n/a	0.1003
[5]	n/a	0.0406	n/a	0.0269	n/a	0.0190	n/a	0.0138	n/a	0.2157	n/a	0.1393	n/a	0.0989	n/a	0.0649
Our	**0.0055**	**0.0186**	**0.0042**	**0.0155**	**0.0034**	**0.0138**	**0.0027**	**0.0119**	**0.0083**	**0.1119**	**0.0051**	**0.0809**	**0.0039**	**0.0660**	**0.0026**	**0.0496**
Dataset	Sintel Dataset								Gesture Dataset							
Ratio	24×24 times (19,43)		16×16 times (28,64)		12×12 times (37,86)		8×8 times (55,128)		24×24 times (17,25)		16×16 times (25,37)		12×12 times (34,49)		8×8 times (50,73)	
Near	0.0278	1.7290	0.0181	1.4092	0.0157	1.3917	0.0085	0.9923	0.0165	0.0284	0.0114	0.0239	0.0092	0.0198	0.0059	0.0157
Bicubic	0.0339	1.4607	0.0226	1.1764	0.0196	1.1932	0.0114	0.8240	0.0180	0.0249	0.0128	0.0205	0.0106	0.0171	0.0068	0.0131
[10]	n/a	1.4461	n/a	1.1586	n/a	1.0219	n/a	0.7975	n/a	0.0249	n/a	0.0207	n/a	0.0193	n/a	0.0176
[5]	n/a	1.5691	n/a	1.1608	n/a	0.9071	n/a	0.6491	n/a	0.0286	n/a	0.0237	n/a	0.0215	n/a	0.0183
Our	**0.0135**	**0.9454**	**0.0088**	**0.7971**	**0.0066**	**0.6940**	**0.0047**	**0.5908**	**0.0123**	**0.0204**	**0.0094**	**0.0168**	**0.0079**	**0.0147**	**0.0056**	**0.0118**

Table 2.1: Depth error compared to ground truth on four kinds of datasets using nearest neighbor, bicubic, [10], [5] and our method. In the row Ratio, 12×12 means 12 times super resolution in each direction, and (92,109) means there are 92 by 109 boxes in the sample grid. log10 is the mean absolute error of log10 depth. RMSE is root mean square error of recovered depth with respect to the ground truth data.

2.3.3 Video Data

Our method also applies to video data with some modifications. We use space time segments, because we expect depth to be fairly smooth within a space time segment, but change on its boundaries (for example, an object moving behind an obstacle). We reconstruct from depth samples that are time-stamped (obtained at a certain time, but have effects at a time period). First, consider points nearby in space. We expect the depth at these points to be similar. This justifies using a smoothing where the influence of a sample declines as points get further away. But temporal smoothing is somewhat different. At each depth sample, we can compute optic flow, which is used to predicts the location of the sample forward and backward in time, by moving the depth sample along the flow dirction. For some time interval, we can trust these flow-based predictions, so we transport depth samples along the flow direction before interpolation. We allow the sample to have influence for times up to δt in the future and $-\delta t$ in the past, and the weighting of a sample declines as the inter-frame time interval increases. We use

$$\omega(\Delta t; \delta t, C) = \frac{\min(-\log(|\Delta t|)/\delta t, C)}{C} \tag{2.13}$$

Δt is the inter-frame time interval, and the function value decreases as $|\Delta t|$ approaches δt.

Now write $\mathbf{c}_i(t_i, T)$ for the location of i'th sample point, which was obtained at time t_i, and current time is T. Write $\mathbf{c}_i(t_i)$ for $\mathbf{c}_i(t_i, t_i)$. The location can be estimated from the optical flow vector $\mathbf{v}$ by approximate integration as

$$\mathbf{c}_i(t_i, T) = \mathbf{c}_i(t_i, T - \Delta t) + \sum_{t_d=1}^{\Delta t} \mathbf{v}(\mathbf{c}_i(t_i, T - t_d)). \tag{2.14}$$

We can now extend our simple depth interpolation model to obtain

$$z(\mathbf{x}, T; \mathbf{a}) = \sum_i a_i \left[\beta(\mathbf{x}; \mathbf{c}_i(t_i, T))\omega(T - t_i; \delta t, C)\right] + z_\pi(\mathbf{x}). \tag{2.15}$$

In principle, $\mathbf{a}$ can be solved by solving a linear system to interpolate the sample points. This linear system is large, and grows with the size of the video. It's hard to solve it directly, so we use a simple and effective approximation. Each depth sample arrives at a frame time, so we can collect the components of $\mathbf{a}$ into groups that apply to particular frames. We solve for each frame independently, then use the resulting $\mathbf{a}$ to reconstruct depth. This approximation is efficient.

Similarly, we extend our advanced depth interpolation model so that $z(\mathbf{x}, T; \mathbf{a}, \mathbf{b}, \mathbf{u})$ is given by

$$\sum_i \left[\begin{pmatrix} a_i\beta(\mathbf{x}; \mathbf{c}_i(t_i, T)) + \\ b_i\psi(\mathbf{x}; \mathbf{c}_i(t_i, T), \mathbf{u}_i) \end{pmatrix} \omega(T - t_i; \delta t, C) \right] + z_\pi(\mathbf{x}). \tag{2.16}$$

Again, we find it sufficient to solve $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{u}$ frame by frame, and then reconstruct for the whole sequence.

2.4 EXPERIMENTAL PROCEDURES

Generally, we obtain high resolution depth maps, subsample them, reconstruct using our methods, then compare the reconstruction to the original depth maps. For the Sintel dataset, the high resolution depth maps are ground truth; for others, they are the best available depth maps. We aim to produce reconstructions that are as close as possible to the high resolution depth. In the thesis, performance is measured in three ways. First, the recovered depth accuracy, such as RMSE. Second, the complexity of the algorithm, such as run time and parallelization. Third, qualitative and quantitative results in applications such as hand tracking.

2.4.1 Datasets

We apply our method to four different datasets: the Middlebury stereo dataset [32] [33], the NYU indoor scene dataset [34], the Sintel synthesised dataset [35] and our Gesture Kinect dataset. These datasets cover a wide range of different types of data (near views; distant views; simple depth profiles; complex depth profiles). Because these datasets are very large, we use 30 representative examples in each dataset; details are included in supplementary

materials. The Middlebury dataset is the most widely used stereo dataset and is also the dataset most super resolution methods use. This dataset contains high resolution depth with lots of details and the images contain lots of textures, which are relatively challenging for segmentation. The NYU dataset is collected from Kinect with extensive post processing, so the depth is better than the Kinect raw depth. This is a widely used RGBD dataset for data driven RGBD image analysis. The Sintel dataset is a synthesized dataset and contains lots of depth details and high quality images. This dataset uses physical simulation to synthesize complex scenes.

2.4.2 Subsampling Strategies

We adopt a strategy standard in ray-tracing circles by subdividing the image into grids, then drawing one sample per grid box. Note that we do not smooth depths before sampling, because we do not envisage that future depth sensors will be able to do so. This means that conventional reconstruction techniques applied to the sampled depths alone will alias significantly.

Generally, we describe a particular sampling regime by the size of the box of pixels replaced by a single sample, so that a 64×64 result refers to a case where there are 4096 times as many pixels to reconstruct as there are samples. We explored two protocols for drawing samples: **Fixed Sampling**, where the sample is in the center of the box (Z_{ij} for depth at pixel i,j, depth samples are $i = ka$, $j = kb$, $a = 1, 2, 3, b = 1, 2, 3, k$ is the integer ratio). Results in the thesis are from fixed samples; we have also experimented with **Gaussian Sampling**, where the location of the sample is a draw from a normal distribution with mean the box center and standard deviation $\frac{1}{6}$ box edge length. As one would expect, we find that differences in error statistics are small, slightly favoring Guassian Sampling. Details in supplementary materials.

2.4.3 Dealing with Noise

The gesture dataset uses raw Kinect depth, so there is noise, mis-alignment and missing values. We deal with depth missing values by using only known depth values. If the segment being smoothed contains only unknown values we mark the segment as having a unique missing value or use the average of the near known neighbors. We control measurement noise by using the advanced smoothing method, which does not interpolate and tends to make the surface smooth in most places. Two approaches help us deal with mis-alignment between the camera and the depth sensor. First, the advanced smoothing method is likely

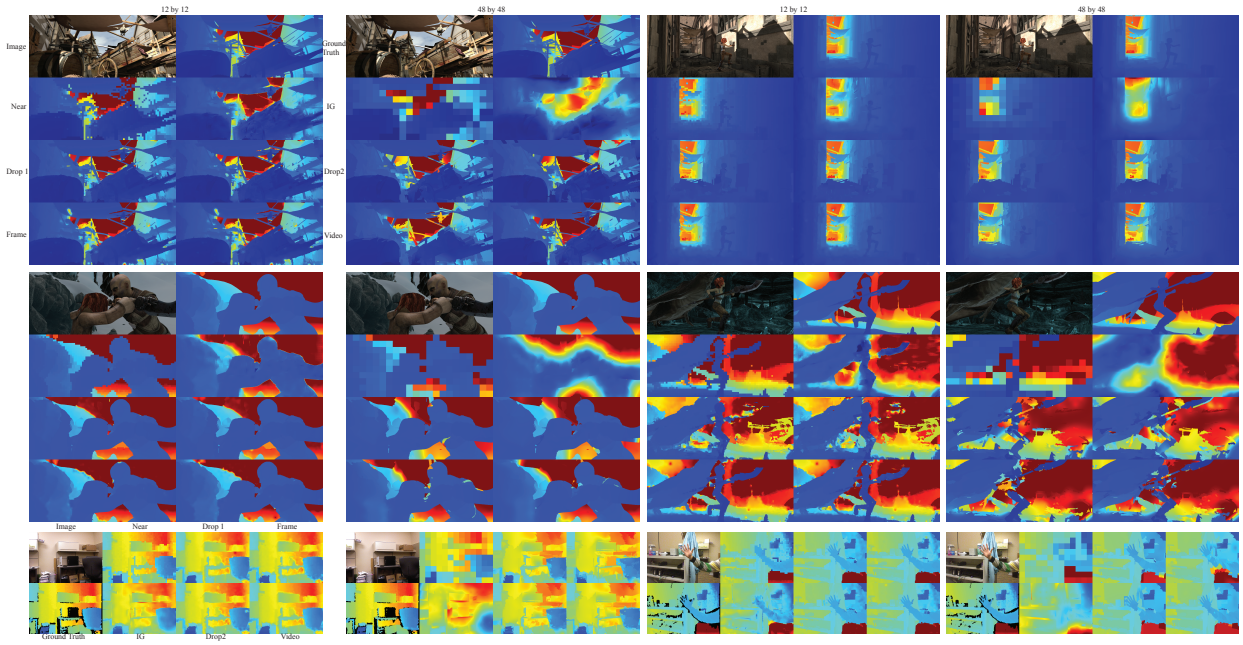

Figure 2.4: A comparison between [5], per-frame reconstructions, video reconstructions, skip 1 depth frame out of every 2 depth frames and skip 2 depth frames out of every 3 depth frames for examples from six sequences. Note that for drop frames, the frames we show here have no depth values. Our video reconstruction works best and even our drop 2 frames works better than [5], especially when the upscale ratio is big.

[bad pixel %, RMSE], smaller is better.

Ratio	Method	Tsukuba		Venus		Teddy		Cones	
	Near	3.53	1.189	0.81	0.408	6.71	1.943	5.44	2.470
	Bicubic	3.84	0.673	0.88	0.290	4.43	2.268	5.98	2.336
4	[36]	2.56	n/a	0.42	n/a	5.95	n/a	4.76	n/a
×	[37]	2.95	**0.450**	0.65	0.179	4.80	1.389	6.54	1.398
4	[22]	**1.73**	0.487	**0.25**	**0.129**	3.54	1.347	5.16	1.383
	[5]	3.03	0.898	0.43	0.201	6.03	0.817	2.98	0.942
	Ours	2.06	0.590	0.29	0.185	**2.03**	**0.593**	**2.56**	**0.938**
	Near	3.56	1.135	1.90	0.546	10.9	2.614	10.4	3.260
	Bicubic	6.67	0.972	2.03	0.427	10.9	2.758	11.9	3.300
8	[36]	6.95	n/a	1.19	n/a	11.50	n/a	11.00	n/a
×	[37]	5.59	**0.713**	1.24	0.249	11.4	1.743	12.3	1.883
8	[22]	3.53	0.753	**0.33**	**0.156**	6.49	1.662	9.22	1.871
	[5]	5.96	1.135	1.98	0.338	12.0	1.376	14.2	1.709
	Ours	**3.52**	0.773	0.47	0.258	**3.82**	**0.863**	**3.65**	**1.041**

Table 2.2: Comparison with state of art methods on Middlebury dataset. Our method is better than other methods, especially when the image quality is similar to current camera images (eg Teddy; Cones). Images in supplementary material. Bad pixel % is the percentage of bad pixels with respect to all pixels of the ground truth data with error threshold 1.

to get rid of a single point that is very different from other points. Second, we identify the largest and smallest ten percent of the depth values within a segment. Any of these points which are also close to the boundary of the segment may represent alignment problems. At problem points, we replace the depth value with a weighted average of the nearest neighbors within the segment, using neighbors that are not themselves problem points.

2.5 RESULTS

Our method yields state of the art results compared to strong recent methods (Table 2.1, Table 2.2, Table 2.3) in both image depth super resolution and video depth super resolution. Results for images depth super resolution are in Figure 2.3.

[log10, RMSE], smaller is better

Ratio	Method	Market		Alley		Ambush		Cave		Office		Gesture	
	Near	0.0531	4.9773	0.0174	2.0154	0.0204	1.0543	0.0319	3.9762	0.0086	0.0365	0.0196	0.0494
12	Bicubic	0.0734	4.2884	0.0182	1.7009	0.0276	0.8958	0.0508	3.3315	0.0092	0.0313	0.0210	0.0436
×	[5]	0.0667	3.7422	0.0215	1.7058	0.0232	0.5401	0.0505	3.4666	0.0131	0.0369	0.0264	0.0356
12	Frame	0.0261	2.5656	0.0131	1.3151	0.0082	0.4580	**0.0188**	**2.4652**	0.0081	0.0290	0.0147	0.0335
	Drop 1	0.0411	2.6793	0.0256	1.5682	0.0216	0.5901	0.0497	4.5012	0.0062	0.0233	0.0133	0.0323
	Drop 2	0.0531	2.8404	0.0310	1.6690	0.0279	0.6516	0.0708	5.7482	0.0066	0.0244	0.0138	0.0329
	Video	**0.0247**	**2.4703**	**0.0101**	**1.2110**	**0.0080**	**0.4511**	0.0200	2.4813	**0.0055**	**0.0208**	**0.0120**	**0.0318**
	Near	0.0899	6.3113	0.0301	2.7225	0.0366	1.3846	0.0626	5.4765	0.0153	0.0508	0.0324	0.0659
24	Bicubic	0.1210	5.4869	0.0294	2.3210	0.0486	1.1637	0.0903	4.6378	0.0157	0.0434	0.0343	0.0549
×	[5]	0.1426	5.1486	0.0437	2.3575	0.0842	1.1855	0.1306	5.4775	0.0255	0.0592	0.0486	0.0671
24	Frame	0.0433	3.2642	0.0227	1.7317	0.0125	0.5612	0.0303	**3.1358**	0.0124	0.0377	0.0219	0.0405
	Drop 1	0.0524	3.0734	0.0307	1.7939	0.0247	0.6407	0.0582	4.8841	0.0093	0.0303	0.0199	0.0369
	Drop 2	0.0660	3.2973	0.0361	1.9139	0.0313	0.7090	0.0793	6.0418	0.0098	0.0318	0.0209	0.0382
	Video	**0.0370**	**2.8828**	**0.0154**	**1.4826**	**0.0116**	**0.5364**	**0.0297**	3.1537	**0.0084**	**0.0281**	**0.0184**	**0.0364**
	Near	0.1396	8.0155	0.0555	3.5887	0.0738	1.9890	0.1197	7.6708	0.0304	0.0781	0.0619	0.0955
48	Bicubic	0.2034	6.8604	0.0547	3.1245	0.0941	1.7141	0.1468	6.6947	0.0293	0.0690	0.0640	0.0846
×	[5]	0.2347	6.2689	0.0781	3.1700	0.1195	1.7024	0.1633	6.8265	0.0356	0.0759	0.0795	0.0944
48	Frame	0.0761	4.3003	0.0322	2.2228	0.0216	0.7437	0.0560	4.2849	0.0165	0.0435	0.0358	0.0534
	Drop 1	0.0753	3.6223	0.0391	2.0803	0.0334	0.8215	0.0816	5.8359	0.0148	0.0411	0.0292	0.0456
	Drop 2	0.0932	4.0598	0.0452	2.1789	0.0405	0.8701	0.1054	7.0715	0.0155	0.0434	0.0309	0.0492
	Video	**0.0582**	**3.3440**	**0.0231**	**1.7372**	**0.0181**	**0.6774**	**0.0480**	**4.1890**	**0.0138**	**0.0387**	**0.0276**	**0.0433**
	Near	0.1588	8.1772	0.0544	3.9099	0.0681	1.9595	0.1040	7.4999	n/a	n/a	n/a	n/a
64	Bicubic	0.2113	7.2739	0.0488	3.2382	0.0965	1.6106	0.1312	6.3807	n/a	n/a	n/a	n/a
×	[5]	0.2206	7.3979	0.0857	3.8713	0.1371	2.1635	0.1791	8.2218	n/a	n/a	n/a	n/a
64	Frame	0.1087	5.1097	0.0418	2.6876	0.0333	0.9675	0.0863	5.6128	n/a	n/a	n/a	n/a
	Drop 1	0.1037	4.4940	0.0466	2.3461	0.0427	0.9905	0.1060	6.9270	n/a	n/a	n/a	n/a
	Drop 2	0.1183	4.7300	0.0543	2.5789	0.0497	1.0212	0.1274	7.9395	n/a	n/a	n/a	n/a
	Video	**0.0847**	**3.9590**	**0.0320**	**2.0638**	**0.0261**	**0.8438**	**0.0721**	**5.2936**	n/a	n/a	n/a	n/a

Table 2.3: Comparison between nearest neighor, bicubic, [5], our image super resolution, our video super resolution, our skip 1 depth frame out of every 2 depth frames and skip 2 depth frames out of every 3 depth frames results. Our image super resolution performs much better than nearest neighor and [5]; our video super resolution is better still; even our drop depth frames works reasonably well.

2.5.1 Video Super Resolution

Using the space-time structure of video yields better results than performing super resolution frame by frame. We evaluated our methods on the Sintel dataset (where there is high resolution ground truth depth), and on Kinect sequences. In each case, we used calculated optical flow (rather than ground truth, which Sintel provides). Table 2.3 shows our per frame super resolution is generally better than all other methods, and that our video super resolution is better than our per frame super resolution. We also works on dropping depth frames, which means at that frame, there are only RGB image and no depth vlaues. Figure 2.4 shows one of the video frames in different videos.

Looking at the video from Kinect, one can see smoothed noise in reconstructions, resulting from the relatively low sampling rate (depths appear to blink or flash). Note that there is a visible measurement noise in the kinect. Note also that this noise is suppressed, but not removed, by using video rather than per-frame reconstruction.

2.5.2 Comparisons

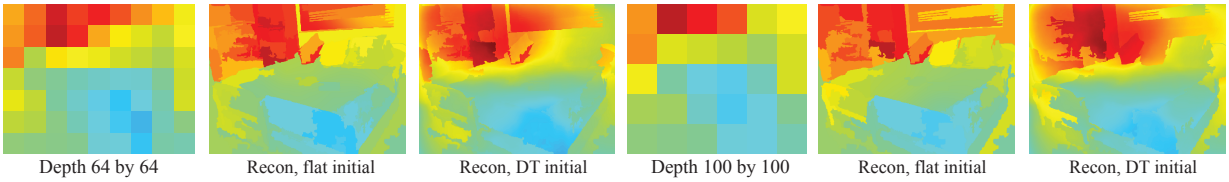

Figure 2.5: At very sparse depth sampling regimes, the depth map inferred by depth transfer can improve the inferred depth, likely by providing good estimates of low spatial frequency components lost in the sampling. We exploit this information using depth transfer as z_π in our reconstruction method. The figure shows an example from 64×64 and from 100×100 subsampling ("initial" refers to z_π). Note the significant improvements obtained by using depth transfer in this case.

Kinect fusion recovers improved accuracy depth maps by fusing multiple depth maps from many different viewpoints [17]. We have no ground truth depths available to do qualitative comparisons, so we show an example comparing our results (with downsampled raw kinect depth as input) with kinect fusion for a view of a keyboard, which contains many fine structures. Figure 2.6 shows the results and each key on the keyboard occupies about 100 pixels. The input Kinect raw depth is noisy and our results are better than the input raw depth. Our results are also comparable to kinect fusion when the upscale factor is relatively small. When the upscale factor is large, many key segments don't have sample data so our method cannot recover these depths.

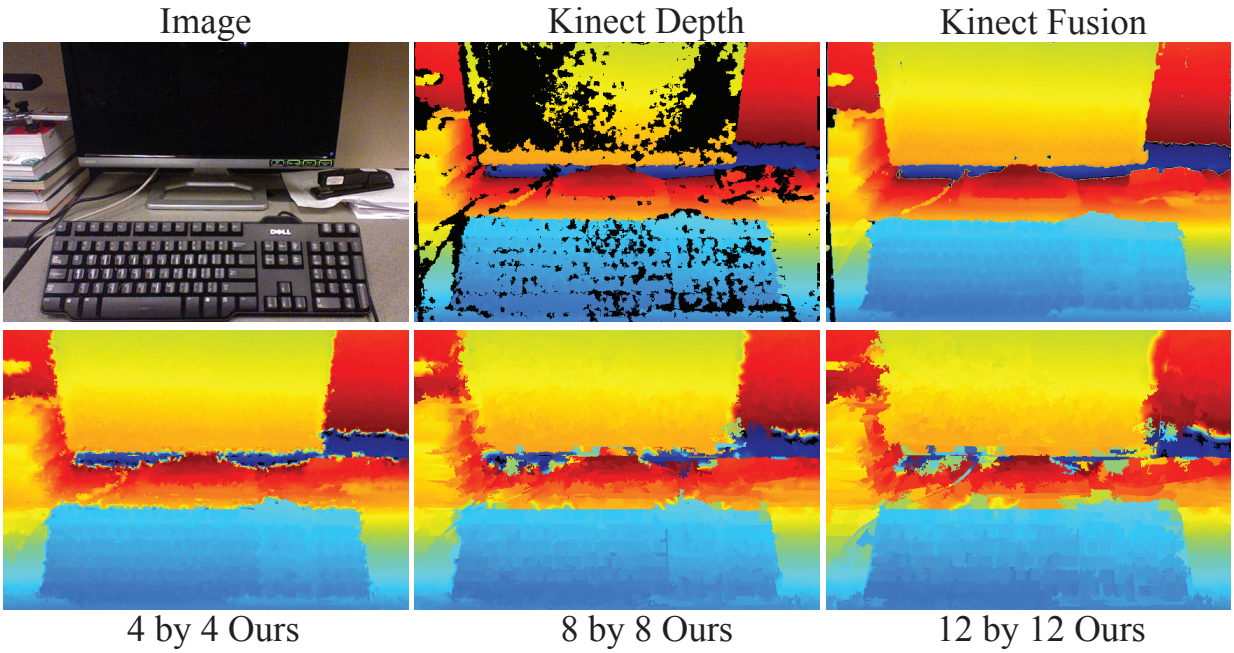

Figure 2.6: Our depth map is comparable to that obtained by kinect fusion, and rather better than raw depth, especially when the super resolution ratio is small. More information is in the movie.

Super sparse sampling presents particular challenges. Figure 2.5 demonstrates our method can operate in this regime. Notice that using a z_π obtained from depth transfer methods yields an improved reconstruction in this regime.

2.5.3 Applications

Depth information is very useful. We demonstrate that our depth super resolution method supports two representative and challenging applications: object insertion and hand tracking from Kinect.

Object insertion methods estimate a depth map for a legacy photograph using depth transfer, then allow users to drag new objects over the estimated scene [38]. Errors in the depth estimate lead to distracting effects in the drag interface. The improvements obtained by our method are sufficient to improve behavior of this interface (see the example in the movie).

Hand tracking is an important application of Kinect which is sensitive to the quality of depth estimates. We evaluated the forth hand tracker [3] [4], (the best currently available depth based hand tracker) on raw kinect depth maps, depth maps subsampled to 24 $\times$24 then reconstructed with nearest neighbors, [5] and our method. Using our depth works as

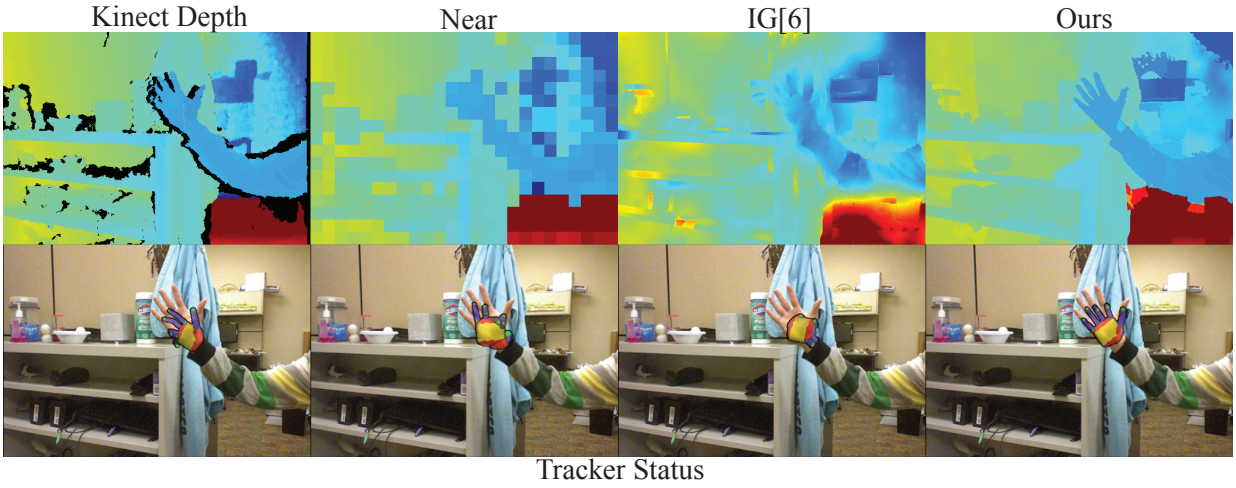

Figure 2.7: Our reconstruction methods support hand tracking from aggressively subsampled depth maps. The first row is depth map and the second row is the status of forth hand tracker [3] [4] (it takes both color image and depth image to generate the status of the hand tracker). In this example at the same moment with different depths, Kinect depth creates difficulty for the hand tracker, nearest neighbour based depth upsample and image guided depth upsampling [5] make the hand tracker fail (because the low quality depth map provides limited information and is even misleading), while our upsampled depth enables the hand tracker to function smoothly.

well as using raw kinect depth, while using nearest neighor and [5] depth creates significant difficulties for the tracker (as judged qualitatively by average tracker score in supplementary materials). Figure 2.7 shows visual results; there are more examples in the movie.

2.6 DEPTH FROM NO SAMPLES

Our method works well because the spatial model is effective; depth really does change quickly at segment boundaries, and slowly elsewhere. This suggests applying the method when there are no samples. There is a considerable literature on regressing depth against images (review in [29]). All methods tend to produce very smooth depth maps as a result of the estimation procedures. The best performing method uses deep network features to represent the image, and regresses depth against these features [30]. Because the features are largely invariant to small local shifts of image patches, the regressed depth is smoothed. It is simple to impose a spatial model that is more sensitive to depth boundaries using our method. We subsample the depth map produced by the method of [30], use mean median filter to process depth samples within each image segment for robustness reasons, then upsample using our method. This introduces stronger depth gradients at segment boundaries, and

leads to a useful improvement in reconstruction error. Qualitative and quantitative results are in supplementary materials.

2.7 DISCUSSION AND LIMITATIONS

Speed: Currently, our advanced method does not run in real time, but the basic method could. Our platform is Windows 8.1 and Matlab 2012b, with 12 GB memory and Intel Core i7. For the biggest images (1300*1100) in dataset, the times are as follows. [5] is about 800s, and [10] is about 30s. For our advanced version, time is about 20s, for our simple version, time is about 5s. The time for each frame in video is similar to single image. The simple version is likely to be real time with good parallel implementation.

Optical flow: Our method extends to upsampling optical flow fields rather well (because flow boundaries tend to appear at image segment boundaries). Detailed results appear in supplementary material. Currently, we have no evidence that a speedup is available from this observation.

Fine details: Our method cannot recover fine or complex surface relief, and will be unhelpful when there is little contrast at object boundaries. Our method tends to work poorly when there are many image segments without a depth sample. Segmentation errors will clearly disrupt our method. We believe that, in general, higher image resolution will have beneficial effects on our results, by producing more detailed segmentations and more accurate optical flow calculations. Finally, errors in sample depth can have serious consequences for our method.

We see a variety of interesting future avenues to explore. A depth camera that makes few depth samples may be able to make those samples more accurately. Finally, our method is adaptive, and it would be interesting to explore procedures that explicitly manage a budget of depth samples to produce the best reconstruction. There is good evidence (the results for skipped frames in Table 2.3) that one could manage this budget over time (as with a power budget).

CHAPTER 3: LOW LATENCY DEPTH SENSING

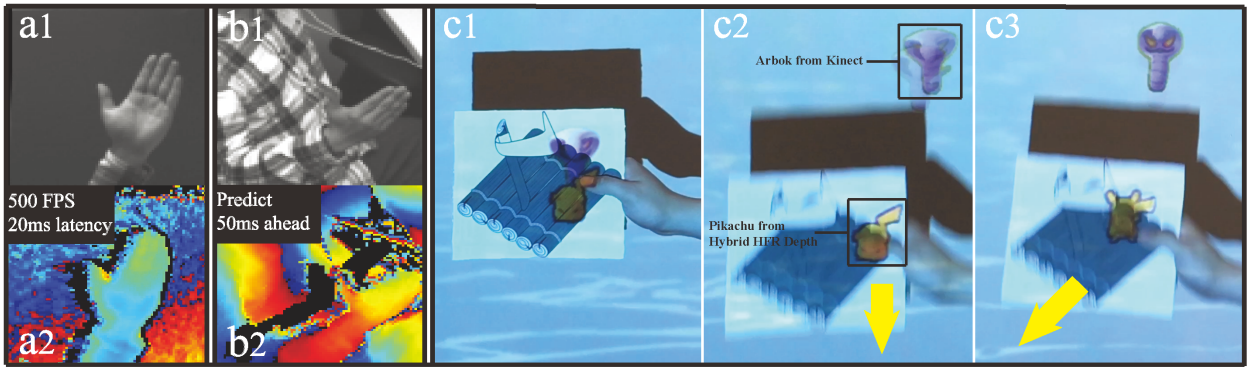

Figure 3.1: Hybrid HFR Depth is a high frame rate, low latency, configurable depth camera solution built from a Kinect and acolor camera. Depth output is 500 frames per second (maximum) and 20 milliseconds latency in (a1-2),and depth is predicted 50ms into the future in (b1-2). Our high frame rate, low latency depth solution supports applications sensitive to lag and framerate:(c1-3)illustrates interactive projection mapping, where Arbok andPikuchu are projected onto the raft using either standard Kinect depth stream (Arbok) or Hybrid HFR Depth stream (Pikachu) to track the moving hand. (c1)Arbok and Pikachu are projected on the raft when hand is still. (c2) (c3) Arbok is off the raft and Pikachu is on the raft when hand is moving because Kinect depth stream has long latency (the sensed hand position is behind the actual hand position) while Hybrid HFR Depth stream has almost no latency.

3.1 OVERVIEW

The low frame rate and high latency of consumer depth cameras limits their use in interactive applications. We propose combining the Kinect depth camera with an ordinary color camera to synthesize a high frame rate and low latency depth image. We exploit common CMOS camera region of interest (ROI) functionality to obtain a high frame rate image over a small ROI. Motion in the ROI is computed by a fast optical flow implementation. The resulting flow field is used to extrapolate Kinect depth images to achieve high frame rate and low latency depth, and optionally predict depth to further reduce latency. Our "Hybrid HFR Depth" prototype generates useful depth images at maximum 500Hz with minimum 20ms latency. We demonstrate Hybrid HFR Depth in tracking fast moving objects, handwriting in the air, and projecting onto moving hands. Based on commonly available cameras and image processing implementations, Hybrid HFR Depth may be useful to HCI practitioners seeking to create fast, fluid depth camera-based interactions.

3.2 RELATED WORK

There are a variety of previous works that seek to increase the frame rate and reduce the latency of depth cameras. They include a range of approaches including simple prediction using a dynamical model, modifications of existing depth image computation algorithms possibly involving exotic hardware, and highly specialized hardware.

3.2.1 Prediction by Dynamical Models

Dynamical models allow a degree of prediction in software, thereby reducing latency. Xia et al. [39] sought to find a camera and software-based approach to reduce the latency of touchscreen interaction. Based on collected training data, Xia et al. estimated touch down locations and triggered device interactions in advance. Knibbe et al. [40] used a Kalman Filter to predict the motion of a ball in flight. Kitani et al. [41] placed a camera inside a ball and used image processing to determine its speed of rotation, triggering the camera at precise moments to capture the scene below. At low frame rates, such dynamical models must be very accurate to make helpful predictions, limiting the utility of the approach.

3.2.2 Specialized Depth Processing

Other related works improve performance of depth cameras by exploiting low-level properties of how depth is computed. Such approaches tend to rely on exotic components, modifications of existing components, and intimate knowledge of how depth is computed on todays depth cameras. The focus of these works is on improving frame rate; improvements in latency are not discussed.

Schmidt et al. [42] presented a method to implicitly calibrate multi-tap 3D Time of Flight sensors, increasing the frame rate by a factor of two. Their prototype is a proof-of-concept written in Matlab.

Stuhmer et al. demonstrated that a modified Kinect v2 operating at 300Hz can track a fast moving ping pong ball accurately [43]. They modified the Kinect depth sensor to capture raw infrared images, and performed model-based tracking against these raw captures. The method requires models for both object and motion, and so works only for rigid simple shapes in simple motion. In contrast, our method obtains high frame rate and low latency depth map for a region of interest under general motion.

Fanello et al. [44] combined an exotic high-frame rate full-frame camera with a Kinect v1 structured light projector to obtain depth frames at 375Hz. They contribute a machine

learning approach to optimize the Kinect v1 computation of disparity to keep up with the high frame rate camera.

3.2.3 Increasing Frame Rate by Specialized Hardware

Specialized hardware may be used to reduce latency and increase frame rate for a particular application. Papadakis et al. [45] minimized latency in head-tracking immersive simulations by reducing buffering latency in their display hardware, achieving a 50% reduction in overall system latency. Lumospheres [46] presented a hardware optimization approach to accurately project on balls under projectile motion. Okumura et al. [47] introduced a low latency camera with a series of saccade mirrors for ball tracking technology. Their system is purely vision based (no depth), so it can only track visually salient objects.

Customized hardware generates impressive results, but the expense of such approaches put them out of reach of typical HCI practitioners. In contrast, our method uses off-the-shelf, affordable hardware.

3.2.4 Hybrid Cameras

Lu et al. [48] demonstrated improving the spatial resolution of the depth stream by calibrate the Kinect with a color camera. They used an edge map and optimization-based interpolation and optical flow to aggressively upsample depth. Their work demonstrated that depth is not hard to interpolate accurately; that current optical flow estimates are accurate enough to support depth interpolations; and registering a depth sensor with color camera can produce an improved depth stream.

In contrast to previous work, our approach shows that color camera can empower depth camera in all aspects. Our approach combines off-the-shelf hardware by vision algorithms to reduce latency and increase frame rate. It is a compromise between exotic and expensive customized hardware approaches and limited software-based approaches.

3.3 OVERVIEW AND CONTRIBUTIONS

Our Hybrid HFR Depth approach combines an off-the-shelf Xbox Kinect v2 and Point Grey Grasshopper 3 camera (see Figure 2(a)), and uses computer vision algorithms to create configurable high frame rate and low latency depth images. Our Grasshopper 3 GS3-U3-51S5M-C Mono supports configuration of region of interest, frame rate, resolution, shutter

speed and so on. It features a Sony IMX250 CMOS, 2/3" imager, running at 75Hz at full 2448x2048 resolution, and at 500Hz with a 256256 region of interest (ROI).

Table 3.1 compares Hybrid HFR Depth with the individual sensors used in our system. As the table shows, Hybrid HFR Depth inherits characteristics of both Point Grey camera and Kinect v2.

	Kinect v2	Point Grey	Hybrid HFR
Frame Rate	30Hz	75-500Hz	75-500Hz
Latency	60-93ms	10ms	22ms
Full frame	512424	24482048	410340
Min ROI	2121	126126	2121
Max ROI	170170	10201020	170170

Table 3.1: Specifications of Hybrid HFR Depth and the two sensors used. Hybrid HFR Depth full frame dimensions are less than that of Kinect due to Kinect having a slightly wider field of view than Point Grey camera.

A main contribution is to use off-the-shelf cameras to create a practical, affordable and easily accessible high frame rate, low latency depth image solution. Compared to customized hardware [46, 47, 45] and software-only approaches [44, 41, 40, 42, 43, 39], it is hybrid in nature. We evaluate high frame rate depth qualitatively and quantitatively and demonstrate the usefulness of our Hybrid HFR Depth in three application scenarios.

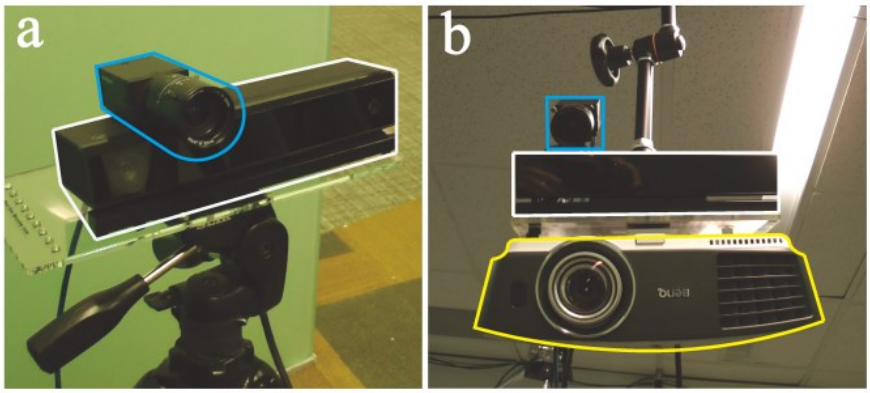

Figure 3.2: (a) Our Hybrid HFR Depth uses off-the-shelf Point Grey camera and Kinect v2. (b) Our projection mapping system uses our Hybrid HFR Depth and an off-the-shelf projector.

3.4 METHOD

Hybrid HFR Depth combines Kinect v2 and Point Grey RGB camera by vision algorithms. We first briefly introduce the pipeline and then include details of each part. First, we rigidly mount the two sensors together and calibrate them (Figure 3.2a). The Point Grey camera has about 6 times higher pixel count in each dimension than Kinect depth frames, so we down sample the color frames and align them with the depth frames. Second, we estimate the delay of each sensor to accurately align the two data streams in time to perform our vision algorithms. Third, we use GPU implementation of the Brox optical flow [49] to estimate the flow from Point Grey color images. This flow field is used to warp the Kinect v2 depth data in XY direction. Because the depth samples not only move in XY direction, but also move in Z direction (as depth value changes), we use linear extrapolation to predict movement in the Z direction. We can further reduce the latency due to the Point Grey camera and image processing. Because we have reliable high frame rate depth stream, we find a simple acceleration model works well to reduce or eliminate latency or predict future depth frames.

3.4.1 Sensor Calibration

We find that a simple, fast calibration modeling only lens distortion and affine transformation works well. First, we use OpenCV's camera calibration routines to estimate the lens distortion of Kinect IR camera and Point Grey color camera. Then, we use RANSAC to estimate the affine transformation matching corresponding points in color images and IR images.

3.4.2 Alignment in Time

Kinect v2 and Point Grey cameras have very different latencies. To align the sensors in time to perform vision algorithms and understand the latency of the system, we measure the latency of each sensor. Point Grey camera latency is measured by recording the time between sending an image request and receiving an image, around 10ms (varies slightly according to the image size). Kinect's latency is more difficult to measure. Instead we measure its latency relative to the Point Grey camera. We swing a scissor in circles like a clock (scissor is the hour hand of clock), and continuously take images with both Point Grey camera and Kinect. For each Kinect image, we find the image from Point Grey camera that best matches the clock (scissor in same rotation angle). Because all images have time stamps, it's easy to find that the relative latency is 50ms. Since the Point Grey camera has absolute latency of

10ms, the actual latency of each Kinect frame is about 60 milliseconds. Kinect generates depth frames every 33 milliseconds (30Hz), so the actual latency is between 60 and 93ms. With sensor latencies and time stamps for both data streams, we can accurately align Kinect depth frame and Point Grey color frame. We align depth frame D_k at time k to color frame at time $t = k - 50$ms.

Our Hybrid HFR Depth obtains a minimum 20ms latency, which is the sum of Point Grey camera latency and image processing time. Our depth stream can be as fast as 500Hz, so the actual minimum latency, without further prediction, is 20+2 milliseconds.

3.4.3 Region of Interest (ROI)

A common CMOS camera can run very fast (high frame rate and low latency) by reducing the region of interest (ROI). We exploit this feature by calculating our fast depth stream on a moving ROI, focusing on the object of interest. This strategy greatly increases effective frame rate while reducing computation cost. We crop the Kinect depth image to the corresponding ROI in software, and configure the Point Grey ROI using the camera's API. Determining the ROI in an application is generally easy, because we need only low frame rate Kinect depth to determine the position of fast moving objects of interest. For example, if we want to focus on the right hand, we can use the Kinect depth to find the center of the right hand at 30Hz and update the Point Grey ROI continuously to track the center of the hand. If the full size depth stream is needed, we can copy the calculated ROI depth back to the full size Kinect depth image. This full frame obtains high framerate and low latency inside the ROI and usual Kinect performance elsewhere.

3.4.4 Generating High Frame Rate Depth Images

High frame rate depth images are computed by warping the latest Kinect depth image by dense optical flow of high frame rate Point Grey images. From color images I_t and I_{t+1} we obtain the flow field $F_{t,t+1}$ and define an image warping function $I_{t+1} = \text{warp}_{F_{t,t+1}}(I_t)$. A high frame rate depth image H_{t+1} is obtained by warping the latest Kinect depth image D_k by the flow field: $H_{t+1} = \text{warp}_{F_{t,t+1}}(D_k)$. Subsequent frames H_{t+i}, $(i > 1)$ are not computed by warping the previous frame H_{t+i-1} but by updating the previous flow field so that it models all motion since the latest depth image. For $i > 1$:

$$F_{t,t+i} = \text{warp}_{F_{t,t+i-1}}(F_{t,t+i-1}) + F_{t+i-1,t+i} \tag{3.1}$$

$$H_{t+i} = \text{warp}_{F_{t,t+i}}(D_k) \tag{3.2}$$

where $F_{t+i-1,t+i}$ is the output of optical flow computation on the two latest color frames.

We use OpenCV's GPU implementation of Brox's algorithm. However, even with GPU acceleration, the flow calculation is slow (maximum 150Hz) compared to our Point Grey camera (maximum 500Hz). We find that the relation between flow calculation time and image size is not linear: the flow calculation time increases much more slowly than image size. We can use batch processing to compute flow over multiple pairs of images to speed up computation at a small cost in latency (refer to Figure 3.4, increasing batch size by one will increase minimum 2ms latency). For example, with a batch size of n, we concatenate n Point Grey images, run optical flow on this large image, and then split the result into n small flow images. This approach allows Hybrid HFR Depth to run as fast as the Point Grey camera.

The warping process described above operates over the spatial domain of the image but does not model changes in the depth values themselves (e.g.; an object moves closer or further away from the camera). To calculate accurate depth values for H we linearly extrapolate the Z component of Kinect depth images. We find the most recent two Kinect depth images D_{k-1} and D_k and their corresponding color images I_{t_1} and I_{t_2}. Assuming that change in depth is constant over a small period, we update the Z component of H_t as:

$$Z_t = \text{warp}_{F_{t_2,t}}(D_k + (D_k - \text{warp}_{F_{t_1,t_2}}(D_{k-1}))(t - t_2)/(t_2 - t_1)) \tag{3.3}$$

This relationship is depicted in Figure 3.3.

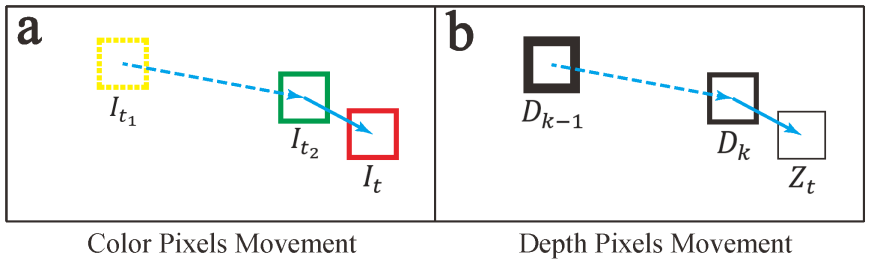

Figure 3.3: Generating high frame rate depth images. (a) Optical flow is used to estimate motion in the color images from I_{t_1} and I_{t_2}, which correspond to Kinect depth images D_{k-1} and D_k. High frame rate depth image H_t is generated by warping D_k by the flow I_{t_2} to I_t. (b) depth values Z are determined by linear extrapolation of the change in depth from D_{k-1} to D_k, following the motion in the color image .

3.4.5 Prediction

Because they are densely sampled in time, the high frame rate depth frames can be used to predict future frames using a second order model of motion. We estimate accurate velocity and acceleration with 30 samples, and assume the acceleration is constant during prediction. With the flow estimation and the acceleration model, we can predict future flows and future depth frames. We can use this approach to reduce latency, possibly to zero or even negative values. Aggressive prediction will naturally affect depth image accuracy, especially around discontinuities in the depth image. Applications that are less sensitive to inaccuracies around object boundaries in the depth image (e.g., object tracking) may benefit from significant prediction.

3.5 CONFIGURATION DETAILS

In this section, we consider various implementation details of the Hybrid HFR Depth approach. These can be configured to suit a given application.

3.5.1 Hybrid HFR Depth specification

There are four quantities that relate to the performance of the proposed technique: ROI, batch size, frame rate, and latency. ROI can be configured from 40 pixels in each dimension, enough to capture the whole hand, to 170 pixels in each dimension, enough to capture the whole upper body. As noted above, our implementation may compute optical flow over several images. Batch size indicates the number of images included in this computation. Changing batch size trades off latency and framerate. The relation between optical flow calculation time and batch size is not linear. For example, when the ROI is 50 pixels in each dimension and batch size is 1, the frame rate is around 100Hz. If batch size is 8, the frame rate is around 400Hz. In general, the frame rate varies from around 100Hz to 500Hz, depending on ROI and batch size, and can be configured to target a given application. Latency in our system is the sum of Point Grey camera latency and processing latency (greater with larger batch sizes). When prediction is used, latency can be reduced to zero or even become negative. We illustrate the relationship between frame rate and latency without prediction under different ROIs and batch sizes in Figure 3.4.

We give the configurations of three example applications we present later in Table 3.2. The basic principle is to tune the most critical aspect of configuration for a given application. In hand writing, for example, frame rate is the most important factor. In tracking, both

higher frame rate and lower latency is desirable. In projection mapping, frame rate similar to projector refresh rate (60Hz) is adequate, while aggressive prediction is needed.

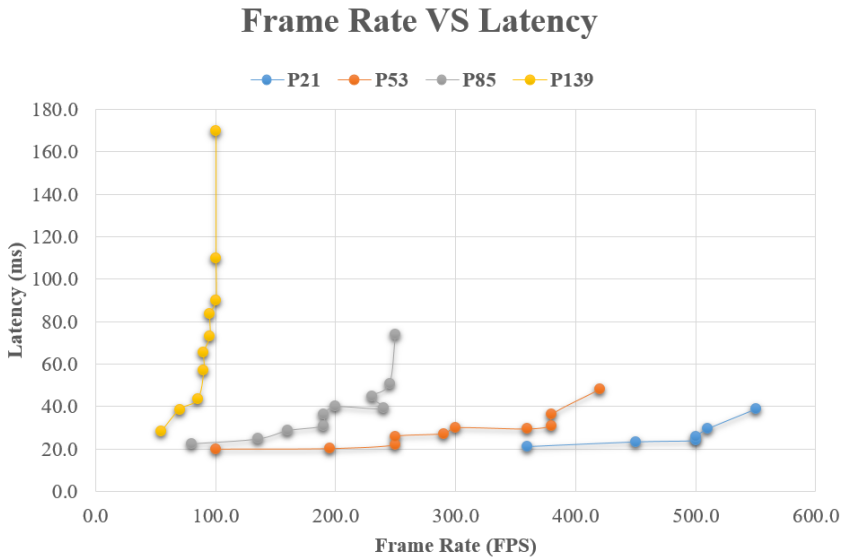

Figure 3.4: Batch processing optical flow calculations impacts overall frame rate and latency: when batch size increases, both frame rate and latency increase at a rate that depends on the size of the ROI. We illustrate the empirical relationship between frame rate and latency for four ROI sizes (P21, e.g., denotes an ROI 21 pixels in each direction). The first point along each curve indicates a batch size of one (no batching). Each point thereafter indicates a batch size of one more frame. In general, (1) larger ROI size causes greater latency, (2) higher frame rates correspond to greater latency (due to batch processing), and (3) latency increases slowly with batch size, at first.

	ROI Size	Batch Size	Latency	Frame Rate
Writing	8585	6	40ms	200
Tracking	7575	4	30ms	200
Mapping	106106	1	22 (-70)ms	70

Table 3.2: Hybrid HFR Depth configurations for the three example applications. With projection mapping, if no prediction is used, the latency is 22 milliseconds. If prediction is used, the latency is -70 milliseconds.

3.5.2 Other Configuration Options

Increased Field of View: Fixing the ROI while down sampling the input depth image sacrifices spatial resolution but significantly increases the effective field of view without

impacting computation time. For example, resizing the depth image to 25

Increased Accuracy: Optical flow calculations are subpixel in nature, but its accuracy is related to the resolution of the input images. Higher resolution images yield more accurate optical flow. In order to increase accuracy, we can use color images with resolution four times that of the depth image to more accurately estimate optical flow, and then down sample the flow. This approach will generate higher accuracy high frame rate depth at the cost of lower frame rate and higher latency.

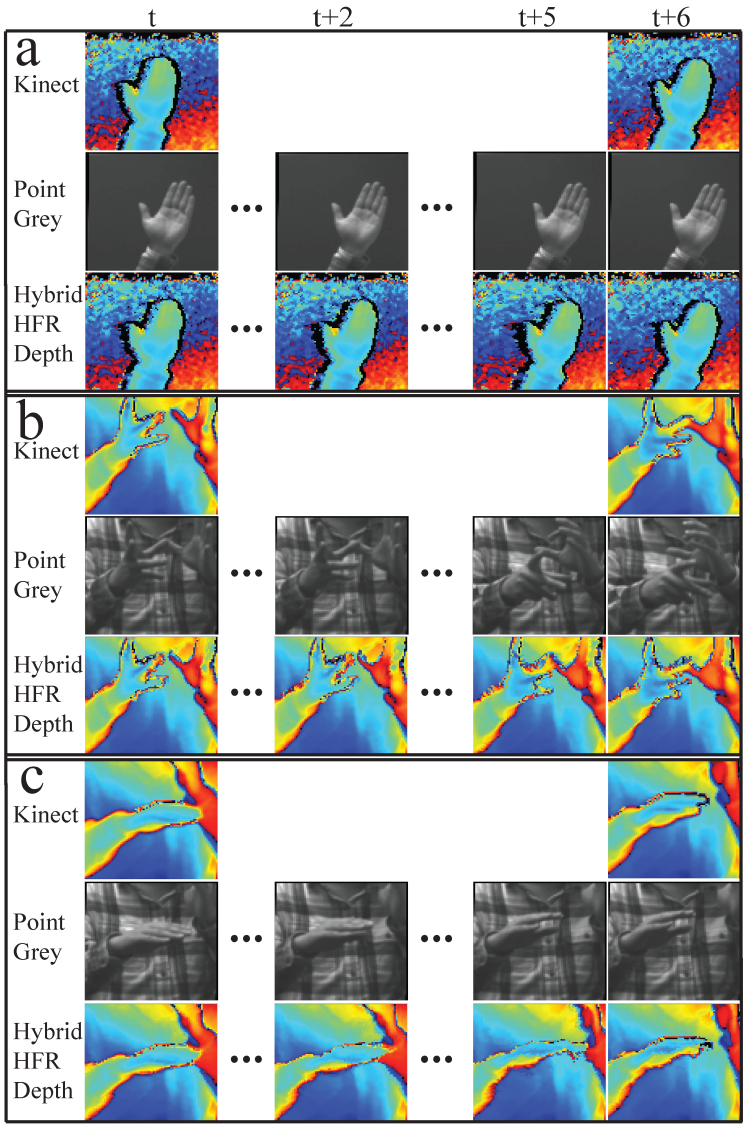

Figure 3.5: Example sequences of hand movement illustrate the quality of high frame rate depth images. Hybrid HFR Depth images have higher frame rate, lower latency (hands move ahead of Kinect depth) and quality similar to the input Kinect depth images.

3.6 EVALUATION

Figure 3.5 illustrates the quality of the high frame rate depth image on examples of hand motion. The accompanying video gives further examples and gives a better sense of the quality of the generated sequences. Depth direction hand motion example is not visually obvious, so we only include a quantitative example in Table 3.3.

3.6.1 Quantitative Evaluation

To evaluate the quality of the Hybrid HFR Depth images we would like to compare the generated 300Hz high frame rate depth image to an equivalently 300Hz high frame rate ground truth depth image. Because we have no such 300Hz high frame rate ground truth available, we instead use the 30Hz Kinect depth image stream as ground truth, and take every tenth Kinect depth image as 3Hz input. This essentially slows down the sequence by a factor of 10. During usage, we speed up both depth streams 10 times to simulate 300Hz ground truth and 30Hz input to the Hybrid HFR Depth algorithm (see Figure 3.6). In recording test sequences, we may take some care in producing motions that are approximately slowed down by a factor of 10.

Dataset

We collected seven sequences of hand motions: three simple hand motions S1-3 (similar to first row of Figure 3.7), three more complex hand motions C4-6 (similar to second row of Figure 3.7) and one hand motion only in depth direction D7 (scene is similar to first row of Figure 3.7 and hand moves in depth direction). In each sequence, the subject moved their hands in approximately one tenth normal speed. We save only frames that have both Kinect depth and Point Grey images, so we get a sequence of normal hand moving speed, and each Point Grey color image has a corresponding depth images. When these sequences are processed, we feed every Point Grey image and one out of ten "Ideal Kinect" depth images into the system, to simulate the different frame rates of two sensors. During evaluation, "Ideal Kinect" depth images are used as ground truth.

Metrics

To compare the generated depth image against "Ideal Kinect" ground truth, we calculate average absolute per pixel difference across the image, ignoring pixels that have no value

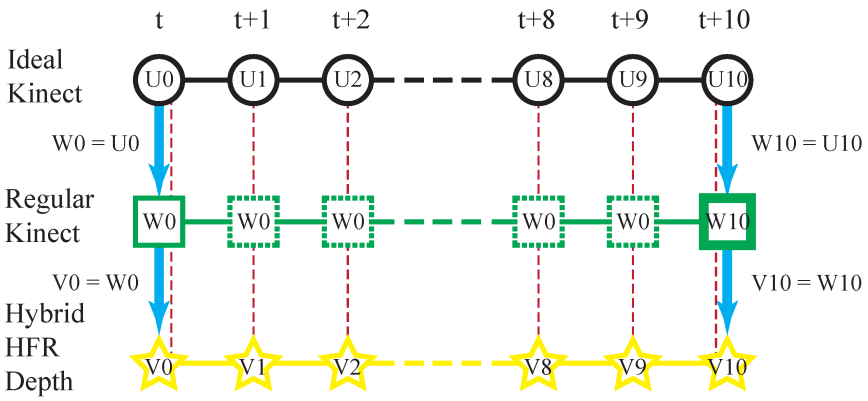

Figure 3.6: Illustration for quantitative evaluation. "Ideal Kinect" is simulated by capturing slow motion with Kinect (30Hz). "Regular Kinect" frames are meant to simulate the standard Kinect stream before the 10x slow simulation and are held for 10 frames. "Hybrid HFR Depth" takes as input every tenth Kinect frame. Symbols inside sample point (circle, rectangle and star) means the depth frames. Red dash line indicates comparisons reported in the text.

Sequence	"Regular Kinect"	Hybrid HFR	Hybrid HFR w/ prediction
S1	42.1mm	28.9mm	29.1mm
S2	65.8	35.2	37.4
S3	72.1	43.4	44.2
C4	65.3	45.9	47.2
C5	47.5	40.1	40.2
C6	65.2	50.6	52.0
D7	99.7	78.6	79.4

Table 3.3: Average per pixel error in millimeters between calculated depth image and ground truth depth image. S1-S3 are simple sequences, C4-C6 are more complex sequences, D7 is motion in depth direction sequence. "Regular Kinect" is described in Figure 6. "Hybrid HFR" refers to our high frame rate depth image with no prediction (minimum latency 20ms), while "Hybrid HFR w/ prediction" includes prediction to reduce latency to zero.

(black pixels in our figures).

Results

Pixel noise around discontinuities in depth can be large and can contribute greatly to our simple error metric. For example, Figure 3.7(f) and (g) illustrate two successive Kinect frames when the subject is not moving. Figure 3.7(h) depicts the absolute error, which can be very large at discontinuities in depth. In this example the average absolute per pixel difference is 15mm. Table 3.3 shows the error for the seven test sequences. In this table,

"Regular Kinect" refers to simply using the "Regular Kinect" stream described earlier as the high frame rate depth image. This serves as a simple baseline measure of performance if one were to naively upsample the Kinect depth image.

As expected, the "Hybrid HFR" and "Hybrid HFR w/ prediction" errors are substantially reduced from that of naive upsampling. "Hybrid HFR w/ prediction" refers to the high frame rate depth image generation with prediction to reduce latency to zero. This results in a slight increase in error compared to the regular "Hybrid HFR" which is synchronized with the Point Grey camera (no prediction).

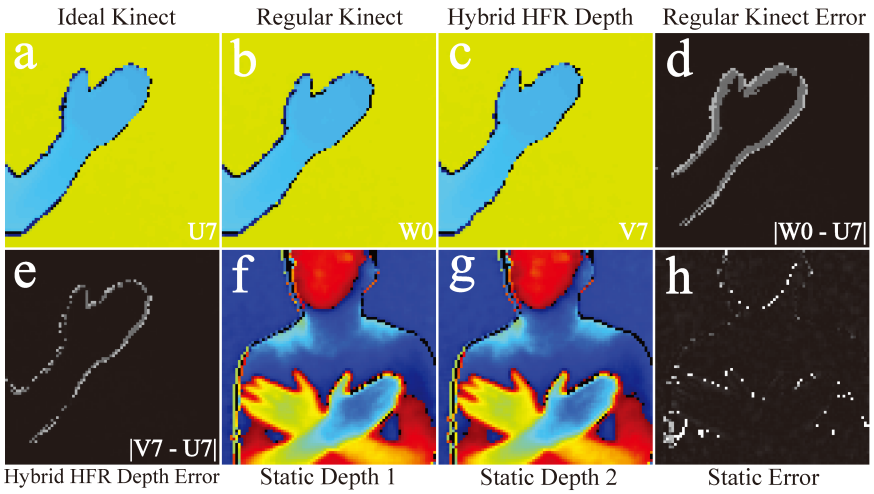

Figure 3.7: Depth errors are large around depth boundaries, (a-e) is an example, corresponds to Figure 6. (a) is ground truth depth U7 of the example from "Ideal Kinect", (b) is the corresponding "Regular Kinect" depth W0, seven frames before the ground truth image, (c) is Hybrid HFR Depth image V7, (d) shows the per pixel error for "Regular Kinect", compared to ground truth, (e) is the same for Hybrid HFR Depth. We evaluate the Kinect sensor noise in (f-h). (f) is a Kinect depth frame at time t, (g) is a Kinect depth frame at time t+1, (h) is the absolute difference between the two depth frames.

3.7 APPLICATIONS

Hybrid HFR Depth benefits a variety of applications that work better with high frame rate and/or low latency sensing. We show three important and representative tasks to demonstrate the usefulness of our system. First, we demonstrate that our low latency and high frame rate depth stream can be used to track fast moving objects. Then, we demonstrate that the high frame rate depth stream can be used in small gesture control and hand writing interfaces. Finally, we demonstrate using prediction to reduce latency in interactive projection mapping.

3.7.1 Object Tracking

Tracking moving objects is a fundamental task in computer vision. Kinect can be very useful in object tracking, particularly when the object moves against a known background depth. However, the low frame rate of Kinect can make tracking fast moving objects accurately more difficult. Our Hybrid HFR Depth can provide robust high frame rate and no latency object tracking, even for fast moving objects.

To demonstrate Hybrid HFR Depth in an object tracking, we give a simple but informative example of tracking a bouncing ping pong ball, which moves relatively quickly and changes direction quickly when it bounces. Tracking the ping pong ball throughout the bouncing motion is relatively easy with the high frame rate and low latency depth map. We record the ping pong ball bouncing events with Kinect, Hybrid HFR Depth and Point Grey cameras. From Figure 3.8, we find that Hybrid HFR Depth produces high quality depth images with high frame rate and low latency.

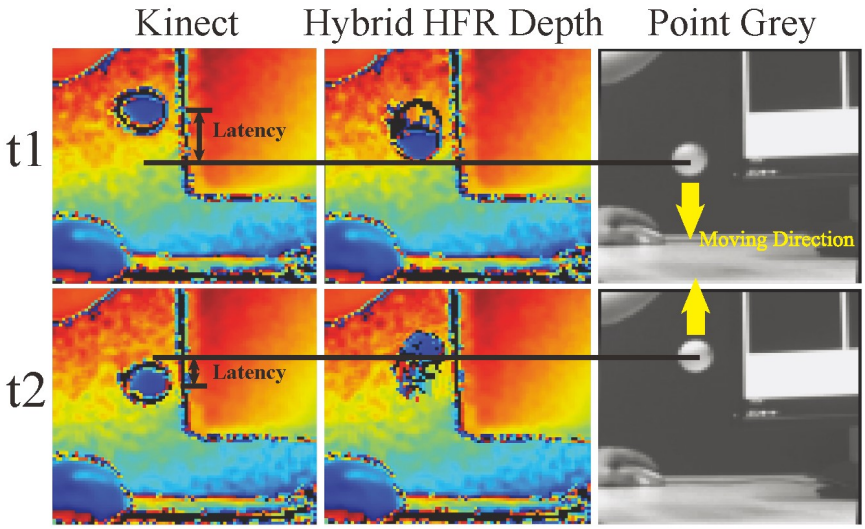

Figure 3.8: A ping pong ball is tracked with both Kinect v2 and Hybrid HFR Depth. The Hybrid HFR Depth image exhibits lower latency than Kinect. The high frame rate depth image is far ahead of the Kinect image but is slightly behind the Point Grey image (due to computation time).

3.7.2 Small Gesture Input

The Kinect sensor has been popularly applied to gesture recognition and control interfaces. Many existing systems either require the users to perform large scale hand gestures or require the users to be near to the controller. Our system enables the capture of small gestures

imaged from a long distance (1.5m to 4m). By accurately detecting fast fingertip gesturing and writing, we can give commands to the remote computer or write on the screen.

High frame rate is particularly important for hand tracking systems, because fingers are flexible and fast, making them especially hard to track. The best open source hand tracker (by Oikonomidis et al.; [50]) still suffers from periods of lost tracking due to high computation cost and low frame rate. Sharp et al. introduced a new pipeline that estimates hand pose per frame [51]. However, tracking is performed at low frame rate.

We can find the fingertip in each depth frame to obtain a trace of the fingertip. It often suffices to find the closest point in the region of the hand. A high quality trace can be difficult to obtain when the frame rate is low (i.e., the fingertip moves a lot between two frames) and the depth noise is high (i.e.; closest point is not always the fingertip). To combat noise, we can smooth the fingertip trace using a Kalman filter. However, such smoothing alters the shape of the trace in undesirable ways. Figure 3.9 shows a number of gestures as recorded by Hybrid HFR depth, and the regular Kinect image with and without smooth. The traces show that users can reliably draw simple gestures with small finger motions in the air at a distance of 1.5m to 4m. With a gesture recognizer such as the "$1 gesture recognizer" [52], traces can be converted to corresponding commands for a remote interface. This could allow for a user to control their Xbox with small finger movement while lying on the sofa, for example.

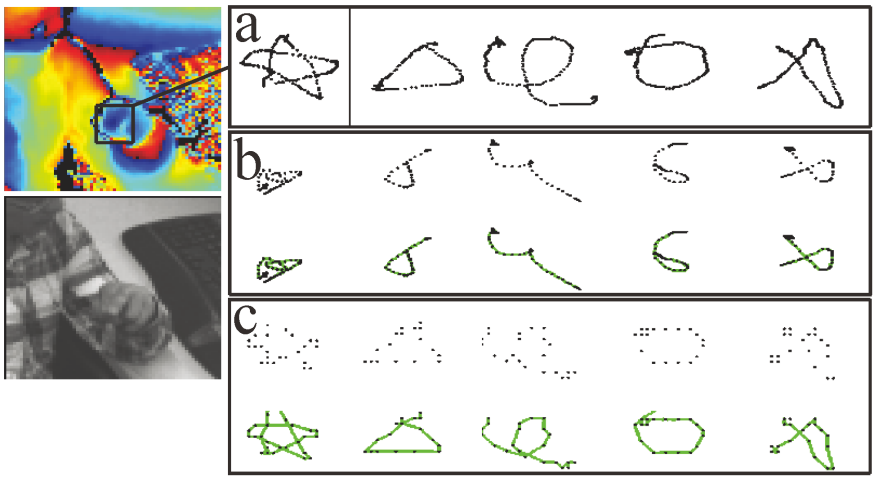

Figure 3.9: The user may draw simple shapes in the air to control a remote interface, for example. (a) shows fingertip trace captured by Hybrid HFR Depth, with Kalman Filter applied. (b) is the same motion captured by Kinect, with Kalman Filter applied. In this case the Kalman filter alters the shape of the input gesture because of aliasing. (c) is the trace of Kinect skeleton hand tip without a Kalman Filter. Clearly high frame rate and subpixel accuracy improves writing in the air.

Besides drawing simple shapes, users may write words in about 2 meters away. This is a challenging task because only high frame rate and accurate depth stream can preserve all the sharp curves, small circles and other details during fast finger movement. Figure 3.10 demonstrates writing a phrase, using the same three techniques as above. The algorithm for Hybrid HFR Depth and Kinect depth hand writing are identical. When the finger moves slowly, the differences between these methods are small. However, when the finger moves quickly we find that Hybrid HFR Depth tracks the fingertip more reliably, and the resulting trace looks better.

Figure 3.10: We demonstrate the advantages of hand writing with Hybrid HFR Depth. (a)(b)(c) are generated from the same motion sequence. The task is challenging because writing words involves many sharp turns and curves, requiring high frame rate and high accuracy. (b) illustrates using the regular Kinect depth stream (closest point), and (c) illustrates the trace of the Kinect skeleton hand tip. (a) illustrates Hybrid HFR Depth trace, which is of noticeably higher quality due to greater sampling frequency.

3.7.3 Interactive Projection Mapping

Projector-camera systems afford various interaction possibilities, combining both natural and mixed-reality 3D interaction. Jones et al. [53] introduced a system to augment the area surrounding a television with projected visualizations to enhance traditional gaming experiences. RoomAlive [54] transformed a room into an immersive, augmented entertainment

experience through the usage of video projectors. Later, Benko et al. [55] proposed a spatial augmented reality system that uses dynamic projection mapping to support interaction with 3D virtual objects. However, in current projection mapping systems, accumulated latency from the depth sensor, image processing, graphics rendering and projection introduces errors in alignment in dynamic scenes. Projected graphics can seem to slip from its expected position, adversely impacting on the immersive experience. Recently, Knibbe et al. [40] and Koike et al [46] used software-based prediction to reduce the system latency. However, they only work with rigid objects in projectile motion. Our Hybrid HFR Depth can be used to track arbitrary motions and objects.

A custom hardware approach such as that of Koike et al [46] can reduce latency, typically with some expense and complexity. Using fast touch sensors rather than cameras, Jota et al. [56] describe a custom hardware-based projection-based drawing system that demonstrates that users are sensitive to surprisingly small amounts of latency (approximately 1ms) in direct manipulation settings such as that of interactive projection mapping.

To make projection mapping work reliably for moving objects with non-customized hardware, we can use our high frame rate and no latency Hybrid HFR Depth stream. While systems such as [40] must predict up to 110ms + 33ms to eliminate constant latency and latency due to camera frame rate, our system can reduce the time to 70 + 4ms (10ms Point Grey camera latency, 10ms optical flow calculation latency and 50ms projection mapping system latency), making prediction easier. Moreover, the higher frame rate can result in higher quality prediction to further reduce latency.

Calibration

We use RoomAlive toolkit [54] to calibrate our projector-camera setup (see Figure 3.2(b)). By applying extrinsic and intrinsic camera parameters, a point in depth image space can be transformed to projector space. A low or zero latency depth image can be used to render virtual objects into a similarly low-latency projection.

Combating Latency

To obtain the best interactive experience with our projection mapping system, we aim to completely eliminate latency, so that graphics projected onto moving objects appear to stay on the object. Operating system, rendering and projector latencies combine to an approximate total of 50ms. Given the 20ms minimum latency of the Hybrid HFR Depth approach, we must apply a prediction of approximately 70ms.

Figure 3.11 shows projection mapping results with Hybrid HFR Depth and Kinect v2. To numerically evaluate how our system works, we count the number of frames in which the projected graphics is fully on the hand, fully off the hand, or partially on the hand, as in [40]. We count these frames manually by analyzing videos of the system in operation, and compare the performance of the standard Kinect depth stream with Hybrid HFR Depth. Table 3.4 shows these counts as a percentage of the test sequences for three speeds of motion.

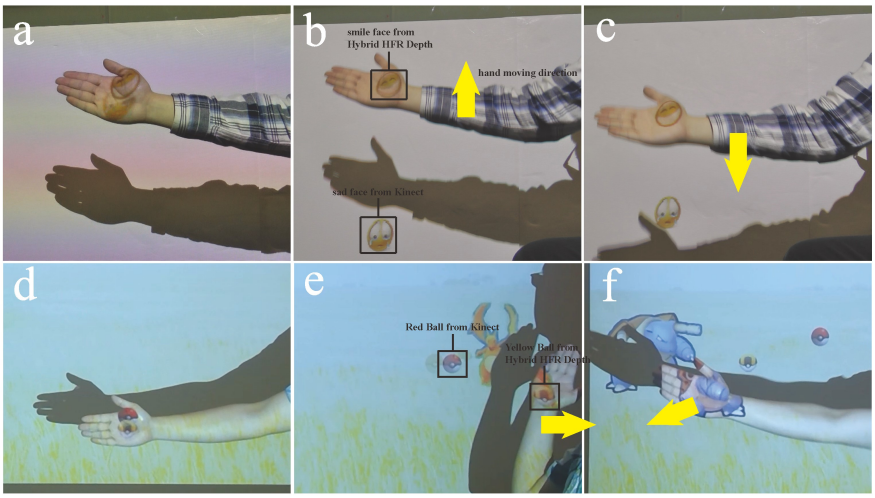

Figure 3.11: We compare dynamic projection mapping results with Kinect v2 and our Hybrid HFR Depth. We use Kinect v2 depth to project a sad face or a red ball onto the hand, and use Hybrid HFR Depth to project a happy face or a yellow ball onto the hand. In (a) and (d), hand is still, in others, hand is moving in the direction of the yellow arrow. In (f), hand is moving very fast, and yellow ball is off the hand, but still ahead of the red ball. In general, the Hybrid HFR Depth based methods stick graphics to moving hand better.

	Kinect			Hybrid HFR Depth		
	All on	Partially off	Fully off	All on	Partially off	Fully off
Slow	18.6%	23.2%	58.2%	71.1%	28%	0.9%
Medium	13.7%	20.4%	65.9%	63.3%	35.6%	1.1%
Fast	9.5%	16.7%	73.8%	44.2%	44.8%	11%

Table 3.4: Percentage of test frames for which projected graphics is fully on, partially off, or fully off the users hand. We manually count the percentage from three 900 frame sequences of slow, medium and fast motions.

3.8 LIMITATIONS

Our Hybrid HFR Depth system has a number of limitations. First, the system requires good, stable lighting. Dim or very strong lighting, or rapidly changing lighting will affect the quality of the optical flow-based motion estimation and therefore impact Hybrid HFR Depth quality. The high frame rate of the Point Grey camera limits the effective exposure time and light gathering capability of the camera; for example, 500Hz capture leaves only 2ms to collect light. In our projection mapping example, rapidly changing graphics might significantly affect the result quality. In this case it may be possible to use infrared imaging, though some care must be taken to avoid the narrow-band illumination of the Kinects time-of-flight imager. Second, optical flow has other limitations that can impact Hybrid HFR Depth quality. For example, optical flow can fail when there is a lack of texture, repeating textures, or strong reflections in the scene.

Third, optical flow implementation of OpenCV is still computationally expensive on todays hardware. Our current Hybrid HFR Depth prototype optionally employs batch processing of optical flow, potentially creating an undesirable tradeoff between latency, frame rate and ROI size (see Figure 3.4). Recently, Kroeger et al. [57] demonstrated a fast optical flow algorithm, which runs at 300-600Hz on a single CPU core with high resolution images. This may remove the need for batch processing, help to obtain $>$ 1000Hz through multi-core scheduling, decrease depth latency and enlarge ROI.

Fourth, the specification of the Point Grey camera used in our prototype is another limitation. With Hybrid HFR Depth, we can reach the maximum frame rate of the Point Grey camera in most ROI size by manipulating batch size. A better color camera might allow a higher frame rate.

Finally, we still need to predict 70ms in applications like projection mapping. Our prediction model uses a simple acceleration model and is optionally further smoothed by Kalman filter. Prediction works fine at most times because our inputs have higher frame rate and less time needs to be predicted. However, during sharp turns and accelerations, predictions are likely to be inaccurate. We still observe that our predictions will come back to the correct values faster than Kinect depth. A better learning based prediction algorithm is likely to further improve the results.

3.9 FUTURE WORK

Hybrid HFR Depth could be improved in a number of ways to make it more powerful and useful.

One interesting direction of future work is to further exploit the complementary nature of the two cameras by creating real time super resolution depth images. This would help in applications where Kinect struggles to deliver adequate resolution. For example, when at a distance of 4m, an adult hand may be 15 pixels wide, making it difficult to resolve individual fingers. Meanwhile, our prototypes color image has about six times more pixels in each dimension. Combining depth information from Kinect with the color information from Point Grey camera, we can potentially generate a depth map six times that of Kinect.

While exploiting faster and more exotic cameras may enable higher frame rates, lower latency and a larger ROI, we also see the value in limiting our implementation to cameras that use inexpensive, commodity imagers, such as common web cameras. Employing an inexpensive common camera might broaden the appeal of the Hybrid HFR Depth approach among practitioners, much as the original Kinect did for interactive applications. CMOS imagers continue to rapidly advance in technology, with many smartphones able to record 240Hz videos.

Regarding applications, Hybrid HFR Depth might be particularly useful in Augmented Reality and Virtual Reality systems, where higher frame rate and lower latency are important factors in rendering stable, accurately aligned graphical augmentation, and in improving user comfort.

3.10 CONCLUSION

This thesis proposes an approach to combine off-the-shelf hardware to create a high frame rate, low latency configurable depth stream. Hybrid HFR Depth is more powerful than a purely software-based approach, and less demanding than a customized hardware approach. We presented detailed specifications of Hybrid HFR Depth to demonstrate the capability and flexibility of the system needed to address a variety of applications. Our evaluation of Hybrid HFR Depth shows that the quality of the generated depth image is good enough to benefit three demonstrated interactive applications.

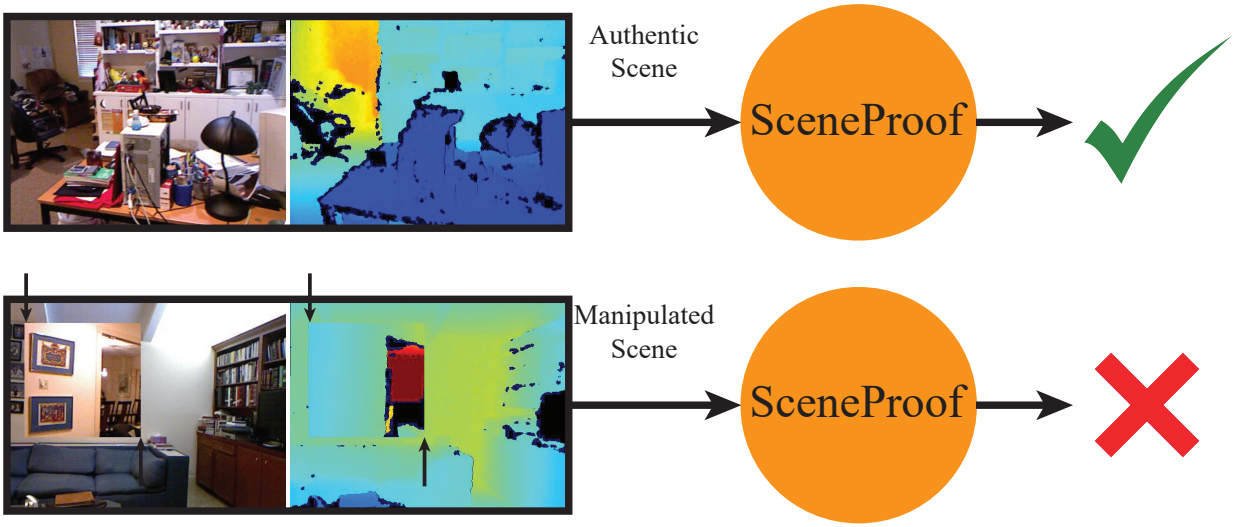

Figure 4.1: SceneProof takes a pair of depth and image, and check whether the scene is authentic or not. It should reject pairs that have been manipulated, such as adversarial examples. In this figure, the first example takes a pair of matched image and depth, and our SceneProof should classify it as authentic. In the second example, a rectangle region of the image and depth is inserted from another source, so our SceneProof should classify it as not authentic.

4.1 OVERVIEW

We build a SceneProof system that can reliably detect whether an image is a picture of a real scene or not. Our system applies to images captured with depth maps (RGBD images) and checks if a pair of image and depth map is consistent. It relies on the relative difficulty of producing naturalistic depth maps for images in post processing. We demonstrate that our system is robust to adversarial examples built from currently known attacking approaches, such as DeepFool. The system depends on our novel network architecture SafetyNet, which utilizes a RBF-SVM inspector to classify adversarial examples based on the internal activations of the neural networks. Our construction suggests some insights into how deep networks work. We provide a reasonable analyses that our construction is difficult to defeat, and show experimentally that our method is hard to defeat using several standard networks and datasets.

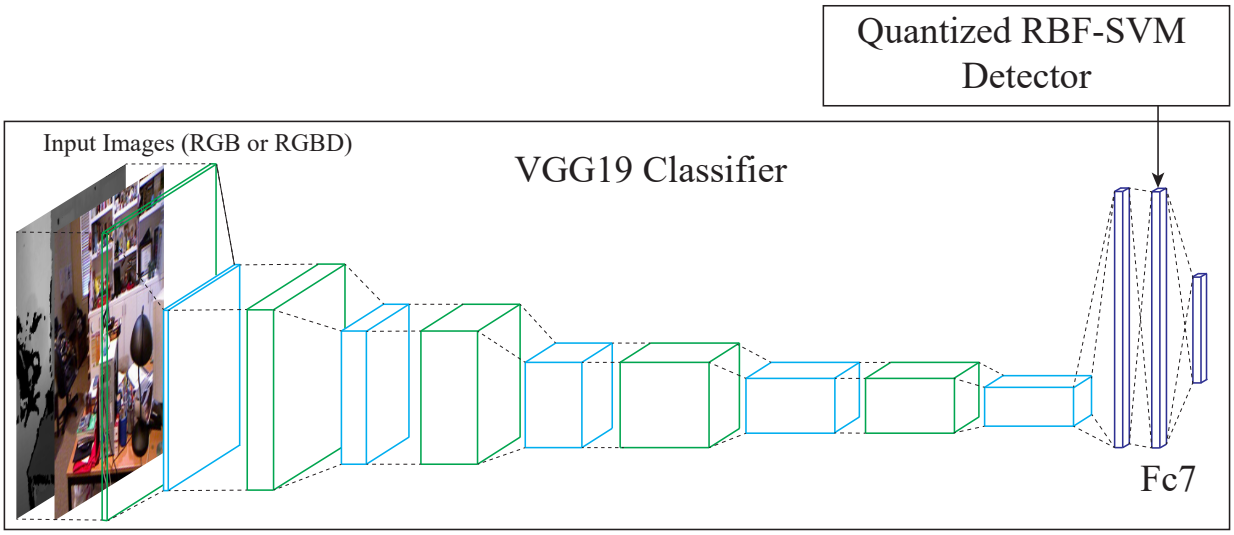

Figure 4.2: SafetyNet consists of a conventional classifier (in our experiments, either VGG19 or ResNet) with an RBF-SVM that uses discrete codes computed from late stage ReLUs to detect adversarial examples. We show that (a) SafetyNet detects adversarial examples reliably, even if they are produced by methods not represented in the detectors' training set and (b) it is very difficult to produce examples that are both misclassified and slip past SafetyNet's detector.

We would like Alice to be able to prove to Bob that her photo is real easily (without the intervention of a team of experts), and we'd like Bob to have high confidence in the proof. This proof needs to operate at large scales (i.e. anyone could produce a proof while taking a picture), and automatically. Our proof will work as follows: Alice will capture a depth map at the same time as an image, and will offer the depth map as a proof; a classifier will then judge whether the depth map and the image match. This proof relies on the fact that it is quite difficult to edit a depth map and an image together. However, there are methods to recover depth maps from images, etc., so for the method to be successful, it should be hard to fool the classifier with a depth map concocted after the image was captured.

Current best methods to identify fake images require careful analysis of vanishing points [58], illumination angles [59], and shadows [60] (reviews in [58, 61]). Such analyses are difficult to conduct at large scales or automatically. We construct a proof by capturing an RGBD image. The proof of realness is achieved by a classifier that checks both image and depth and determines whether they are consistent.

The main concern is attacks that cause "fake" images to be labeled "real". Natural attacks on our system are:

- produce a depth map for an RGB image using some regression method to obtain an RGBD image (regression);
- manipulate RGBD image by inserting new objects;
- take an RGBD image labeled "fake" and manipulate it to be labeled "real" (type I adversarial);
- take an RGBD image labeled "fake" and manipulate it to be labeled "real" in a way that fools SafetyNet's adversary inspector (type II adversarial).

There is a wide range of available regression/adversarial attacks, and our system needs to be robust to various methods which might be used to prepare the regression/adversarial attack.

4.2.1 Difficulties

For the SceneProof task, the system works if (a) the classifier is acceptably accurate (i.e. it can determine whether the pair is real or not accurately); (b) it can detect a variety of adversarial manipulations of depth or image or both.

We can train an accurate classifier as long as we have enough data, however, adversarial examples are harder to defend. We achieve this by designing out novel SafetyNet architecture, which is able to yield a reasonably reliable scene proof.

Hypothesis 1 *adversarial attacks work by producing patterns of activation in late stage ReLUs that are different from those produced by natural examples.*

4.3 ADVERSARIAL EXAMPLES

Write $P(X)$ for the probability distribution of examples. At least in the case of vision, $P(X)$ has support on some complicated subset of the input space, which is often referred to as the "manifold" of "real images". Nguyen *et al.* show how to construct examples that appear to be noise, but are confidently classified as objects [62]. This construction yields $\mathbf{a}(\mathbf{x})$ lies outside the support of $P(X)$, so the classifier's labeling is unreliable because it has not seen such examples.

Common constructions for adversarial examples include: line search along the gradient [6]; LBFGS on an appropriate cost [63]; DeepFool [8]. Each relies on the gradient of the network, but it is known that using the gradient of another similar network is sufficient [9], so concealing the gradient does not work as a defense for current networks.

Recently, there are stronger adversarial attacking methods. Papernot et al. used Jacobian-based saliency map to attack [64]. Carlini and Wagner launched a targeted C&W attack [65]. Su et al. proposed one pixel attack, which only need to modify one pixel of the image [66]. Liu et al. used model-based ensembling attack [67]. These stronger attacking methods still depend on gradients. Different from gradient-based adversarial generating approaches, Chen et al. proposed a Zeroth Order Optimization (ZOO) based attack [68]. Researchers even generate optical flows to spatially transform adversarial examples to attack [69].

Networks that generalize very well remain susceptible to adversarial examples that are very similar to real images (eg [63], figure 5). These examples are relatively easily found with straightforward searches and are close to the initial point of the search. Adversarial examples that "look like" images to humans, like figure 5 in [63], are likely to lie within the support of $P(X)$. However, the *measure* of such examples under $P(X)$ must be small, otherwise the classifier would generalize poorly. This means that the set of adversarial examples contains a component that is small in the measure of $P(X)$, but is close to, and easy to find from, any point in the support of $P(X)$. Experiments suggest some form of universal property of this set, because examples that are adversarial for one network tend to be adversarial for another as well [63, 70, 71, 72].

Some network architectures appear to be robust to adversarial examples [73, 74]. Showing that a classifier is subject to adversarial constructions just requires a construction; but showing it is *not* requires theory. There is no current theory that explains how to construct classifiers that are not subject to adversarial examples. This makes it difficult to know what defenses can be relied on practically (for example, distillation was proposed as a defense [75], but was later shown to not work [76]).

There are not many works on defending adversarial exmaples, mainly because it is a difficulty problem. Recently, Guo et al. [77] showed that simple image processing, such as down-up sampling and total variation smoothing defense, could defeat a majority of imperceivable adversarial attacks. However, they have difficulty to defend more obvious adversarial perturbations [78].

One strategy for building a network that is robust to adversarial examples is to train networks with enhanced training data (adding adversarial samples [79]). The amount of training data required must be exponential in some parameter encoding the dimension of the support of $P(X)$. The value of the parameter is unknown, but it is big enough that an unreasonable quantity of training data is required by these approaches.

4.3.1 Adversarial Constructions

We use the following standard and strong constructions of adversarial examples [76], with various choice of hyper-parameters, to test the robustness of the systems. Each attack searches for a nearby $\mathbf{a}(\mathbf{x})$ which changes the class of the example and does not create visual artifacts. We use these methods to produce both type I attack (fool the classifier) and type II attack (fool the classifier *and* sneak past the detector).

Fast Sign method: Goodfellow et al [6] described this simple method. The applied perturbation is the direction in image space which yields the highest increase of the linearized cost under l_∞ norm. It uses a hyper-parameter ϵ to govern the distance between adversarial and original image.

Iterative methods: Kurakin et al. [7] introduced an iteration version of the fast sign method, by applying it several times with a smaller step size α and clipping all pixels after each iteration to ensure that results stay in the ϵ neighborhood of the original image. We apply two versions of this method, one where the neighborhood is in L_∞ norm and another where it is in L_2 norm.

DeepFool method: Moosavi-Dezfooli et al. [8] introduced the DeepFool adversary, which is able to choose which class an example is switched to. DeepFool iteratively perturbs an image x_0^{adv}, linearizes the classifier around x_n^{adv} and finds the closest class boundary. The minimal step according to the l_p distance from x_n^{adv} to traverse this class boundary is determined and the resulting point is used as x_{n+1}^{adv}. The algorithm stops once x_{n+1}^{adv} changes the class of the actual classifier. We use a powerful L_2 version of DeepFool.

Transfer method: Papernot et al. [9] described a way to attack a black-box network. They generated adversarial samples using another accessible network, which performs the same task, and used these adversarial samples to attack the black-box network. This strategy has been notably reliable.

4.4 SAFETYNET ARCHITECTURE: SPOTTING ADVERSARIAL EXAMPLES

In SafetyNet, the model architecture consists of the original classifier, and an inspector which looks at the internal state of the later layers in the original classifier, as in Figure 4.2. If the inspector declares that an example is adversarial, then the sample is rejected. In experiments, we demonstrate that SafetyNet can robustly detect type I attacks, and type II attacks on SafetyNet fail.

4.4.1 SafetyNet Model

The inspector needs to be hard to attack. We force an attacker to solve a hard discrete optimization problem. For a layer of ReLUs at a high level in the classification network, we quantize each ReLU at some set of thresholds to generate a discrete code (binarized code in the case of one threshold). Our hypothesis 1 suggests that different code patterns appear for natural examples and adversarial examples. We use an inspector that compares a code produced at test time with a collection of examples, meaning that an attacker must make the network produce a code that is acceptable to the detector (which is hard; section 4.2). The inspector in SafetyNet uses an RBF-SVM on binary or ternary codes (activation patterns) to find adversarial examples.

We denote a code by $\mathbf{c}$. The RBF-SVM classifies by

$$f(\mathbf{c}) = \sum_{i}^{N} \alpha_i y_i \exp(-||\mathbf{c} - \mathbf{c}_i||^2/2\sigma^2) + b \tag{4.1}$$

In this objective function, when σ is small, the detector produces essentially no gradient unless the attacking code $\mathbf{c}$ is very close to a positive example $\mathbf{c}_i$. Our quantization process makes the detector more robust and the gradients even harder to get. Experiments show that this form of gradient obfuscation is quite robust, and that confusing the detector is very difficult without access to the RBF-SVM, and still difficult even when access is possible. Experiments in section 4.2 and theory in section 4.6 confirms that the optimization problem is hard.

4.4.2 Classifier Uncertainty for Type II Attacks

Type II attacks are likely more important than type I attacks, because any deployment of defenses will lead to attacks that attempt to defeat the defenses. However, a type II attack must not only fool a classifier, but also fool a detection subnetwork whose purpose is to find odd examples. Adversarial example constructions attempt to push the posterior of the right class label below that of some wrong class label. Some small perturbations are enough to fool the detection subnetwork, but these constructions do not look at the gap between the posteriors.

We hypothesize that larger gaps require larger perturbations, which are easier to detect with SafetyNet. This suggests the following strategy for boosting SafetyNet accuracy on type II attacks: compute some statistic representing classifier uncertainty; then reject as adversarial any example that has too high an uncertainty, *or* is rejected by SafetyNet. This

approach produces significant improvements in accuracy, in applications that can tolerate some rejection of natural images.

Our uncertainty is computed as the ratio of the posterior for the second highest class to the posterior for the highest class. So, for example, if $p(\texttt{dog}|\mathbf{x}) = 0.6$ and $p(\texttt{cat}|\mathbf{x}) = 0.15$, then our classification uncertainty is 0.25. Write $U(\mathbf{x})$ for this uncertainty ratio, and $H(\mathbf{x})$ for the posterior of the highest class conditioned on $\mathbf{x}$. Incorporating classifier uncertainty is straightforward. An example is rejected if either the classifier uncertainty is too high, or the detection subnetwork detects an adversarial example.

4.5 SCENEPROOF SYSTEM BASED ON SAFETYNET

4.5.1 Network Architecture

Our SceneProof architecture uses the SafetyNet model. Since we take RGBD images as input, our original classification network takes four input channels with either VGG19 or ResNet architectures. The default setting uses VGG19 except otherwise mentioned. Our adversarial inspector uses RBF-SVM, and for VGG19, we perform the detection on the fc7 (A), fc6 (B), or pool5 (C) layer (20% quantized), for ResNet, we perform the detection on the quantized PreLogits layer.

4.5.2 SceneProof Dataset

Real test data is easily obtained. We use the raw Kinect captures of LivingRoom and Bedroom from NYU v2 dataset [34]. However, fake data requires care. "Regression" methods used in both train and test are:

- random swaps of depth and image planes;
- single image predicted depth using the method of [30];
- rectangle cropped region insertion and random shifted or scaled misaligned depth and image.

"Regression" methods used *only* in test are:

- all zero depth values;
- nearest neighbor down-sample and up-sampled images and depths;

- low quality JPEG compressed images and depths;
- Middlebury stereo RGBD dataset [80] and Sintel RGBD dataset [35](which should be classified "fake" because they are renderings).

Refer to Figure 4.3 for dataset and attacks.

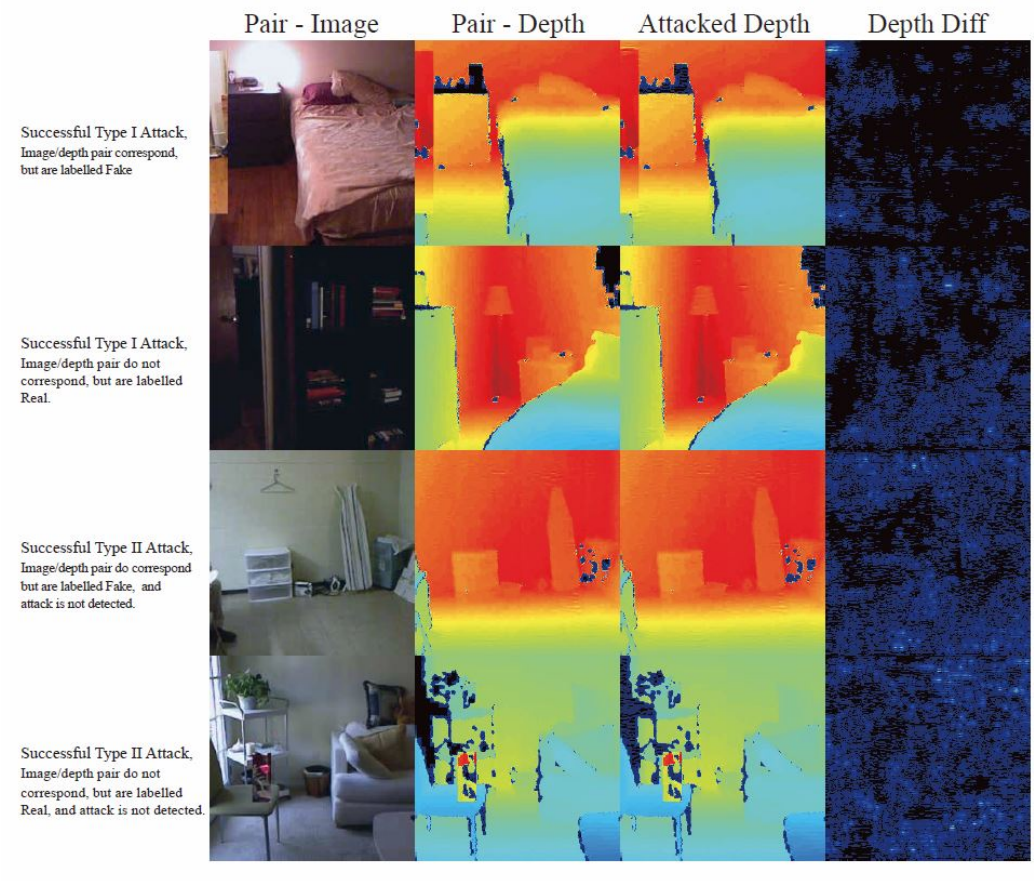

Figure 4.3: SceneProof classifies an image/depth pair as Real when they correspond, and Fake otherwise. An attack on SceneProof involves manipulating one of the pair (in this case, adversarial attacks on the depth map) to change the label. This figure shows examples of successful type I (fool the classifier) and type II (fool the classifier and the inspector) attacks.

4.5.3 Evaluating SceneProof

We see SceneProof's most likely application in consumer level certification that a picture is real. For example, Alice can prove to Bob that she went on vacation at some scenic spot. For this scenario, users don't need to know about failed attacks (which are important in other applications).

SceneProof classifies natural RGBD pairs accurately, and type I attacks on SceneProof using a familiar adversary (i.e. one used to train the inspector) fail (tables 4.1 and 4.2). We show results for four different possible configurations of the SceneProof system, using different inspectors or combinations. An image passes through the classifier, and is then subjected to inspection. A summary evaluation must determine (a) whether the system works for natural images and (b) is the system robust to natural attacks. We assume that the natural attack is to take a false pair of images and try to make the system accept it. In this case, the most important summary statistics are:

- **tna**, the rate at which a **t**rue pair of **n**atural images is **a**ccepted (classifier classifies it correctly and the inspector accepts it); **larger** is better;
- **fnr**, the rate at which a **f**alse pair of **n**atural images is **r**ejected (classifier classifies it correctly *or* the classifier classifies it incorrectly and inspector fails it); both **large and small** are good;
- **far**, the rate at which a **f**alse pair of images that is attacked by an **a**dversary (intending to make it true) but is **r**ejected (either classifier classifies it correctly *or* the classifier classifies it incorrectly and inspector fails it); **larger** is better;

Besides these, we also used some other evaluation summary statistics in our experiments.

- F_u, the fraction of attacks detected; **larger** is better;
- F_d, the fraction of undetected attacks that change a label; **smaller** is better;
- **tnt**, the rate at which a **t**rue pair of **n**atural images is classified as **t**rue (used when no inspector involved); **larger** is better;
- **fnf**, the rate at which a **f**alse pair of **n**atural images is classified as **f**alse (used when no inspector involved); **larger** is better;
- **faf**, the rate at which a **f**alse pair of **a**dversarial images is classified as **f**alse (used when no inspector involved); **larger** is better;
- **taf**, the rate at which a **t**rue pair of **a**dversarial images is classified as **f**alse (used when no inspector involved); **larger** is better;

Attack	**tna**	**fnr**	**far**
Familiar	83.8	37.7	84.6
zero D channel		29.6	79.0
down-up sample		68.3	72.8
low quality JPEG		87.2	86.7
Sintel RGBD		68.8	62.5
Middlebury RGBD		59.4	50.0

Table 4.1: Summary statistics for SceneProof, facing a familiar adversary (first row) and a set of unfamiliar adversaries (following rows). Natural image statistics are not affected by the adversary, and are not repeated. SceneProof classifies natural RGBD pairs accurately, and type I attacks on SceneProof using both a familiar (i.e. one used to train the inspector) and an unfamiliar adversary fail. In this table, results are calculated on the whole SceneProof dataset test split (section 4.5.2). These are results for ABC detector.

Test example type	Classifier Acc	A	B	C	ABC	ABC, T
Natural RGBD, False	91.8	17.1	15.2	30.1	37.7	50.0
Natural RGBD, True	97.7	11.6	10.1	10.7	16.2	13.5
Type I adversarial RGBD, False	33.1	88.6	89.1	70.4	84.6	80.5
Type I adversarial RGBD, True	15.3	81.0	81.3	88.7	93.7	77.2

Table 4.2: In this table, results are calculated on the whole SceneProof dataset test split (section 4.5.2). The type I attacks are by a familiar adversary, LBFGS attack. We report results for three inspectors: A is applied to fc7 of VGG19; B is applied to fc6 of VGG19; C is applied to pool5 for VGG19. Each inspector column gives the rate at which the inspector labels examples as adversarial. ABC (A∨B∨C) means the example is reported adversarial if either inspector says it is adversarial. This configuration is best, because it reliably identifies adversarial examples better and is not much more inclined to produce false positives than the others. ABC, T gives the fraction of examples that were detected as adversarial among these that were labelled True by the classifier (this is required to construct table 4.1). We break out results by type and ground truth label. For example, for natural RGBD images that do not represent real scenes (first row), the classifier labels 91.8% correctly; of the 8.2% incorrectly labelled true, the ABC adversary inspector rejects 50.0% as adversarial examples (last column).

4.5.4 Experimental Results: Summary

During the evaluations with VGG19, there are three natural layers at which to build inspectors: A (applied to fc7 of VGG19), B (applied to fc6 of VGG19), and C (applied to of VGG19) and natural logical combinations of these inspectors. Table 4.2 give summary statistics for the effectiveness of different inspectors. Our results suggest that ABC ($A \vee B \vee C$) is most effective (it reliably detects adversarial examples at a higher rate, and is

Test example type	Classifier Acc	A	B	C	ABC	ABC,T
zero D channel	76.1	8.0	7.5	26.9	29.6	83.7
down-up sampled	77.9	51.0	54.7	59.9	68.3	90.6
low quality JPEG	35.8	81.5	80.1	79.2	87.2	85.4
Sintel RGBD [35]	34.3	34.3	50.0	56.3	68.8	61.9
Middlebury RGBD [80]	25.0	46.9	37.5	37.5	59.4	66.7
type I adv, zero D channel	0.1	30.4	66.7	73.5	79.0	79.0
type I adv, down-up sampled	3.8	52.6	64.8	62.6	72.8	73.7
type I adv, low quality JPEG	0.3	77.7	81.7	78.5	86.7	86.8
type I adv, Sintel RGBD [35]	6.3	31.3	56.3	48.4	62.5	66.7
type I adv, Middlebury RGBD [80]	0.0	34.4	28.1	40.6	50.0	50.0

Table 4.3: In this table, results are calculated on the whole SceneProof dataset test split (section 4.5.2). The type I attacks are by unfamiliar adversaries. All these examples should be labelled false, *OR* rejected as adversarial. We report results for four inspectors: A is applied to fc7 of VGG19; B is applied to fc6 of VGG19; C is applied to pool5 for VGG19. Each inspector column gives the rate at which the inspector labels examples as adversarial. ABC means the example is reported adversarial if either inspector says it is adversarial. This configuration is best, because it reliably identifies adversarial examples better and is not much more inclined to produce false positives than the others. ABC, T gives the fraction of examples that were detected as adversarial among these that were labelled True by the classifier (this is required to construct table 4.1). We break out results by type and ground truth label.

not much worse than any other inspector for natural examples), so we use *ABC* setting to perform most of the experiments with SceneProof.

The summary statistics for SceneProof with familar input data and unfamiliar input data, together with their corresponding type I attacks are shown in Table 4.1 (We feed in these data for SceneProof, and use our RBF-SVM detector trained on adversarial examples to detect these samples). Unfamiliar input data includes: all zero depth values; nearest neighbor down-sampled and up-sampled images and depths; low quality JPEG compressed images and depths; Middlebury stereo RGBD dataset [80] (false examples, and so an attack, because depth is not captured but is computed by stereo); and the Sintel RGBD dataset [35] (false examples, and so an attack, because images are not captured but rendered). The detailed breakdown for familiar adversarials can be found in Table 4.2 and for unfamiliar adversarials can be found in Table 4.3.

The summary statistics for non attack data, type I attacks and type II attacks on SceneProof are shown in Table 4.4. This table contains multiple attacking methods with multiple settings and some numbers are expensive to acquire, so we use a random subset of 3200 images from the dataset test split. Type I attacks and type II attacks are further divided into white box attacks and black box attacks. The detailed breakdown for type I attacks

can be found in Table 4.5 and for type II attacks can be found in Table 4.6.

SceneProof is accurate: From these tables, we find that summary statistics are large and errors are small; SceneProof mostly correctly classifies familiar natural images if there is no adversarial attack, and rejects most unfamiliar natural images as well as adversarial attacks. It is also robust to type II attacks which focuses on attacking both classifier and inspectors.

4.5.5 Experimental Results: Type I Attacks

White box type I attacks assume that the classifier is known to the attacker; this is the harder case for the defender. We use LBFGS based attacks, and the attacking methods on training data and testing data are same. Black box type I attacks assume that the attacker can access to another network trained on the same dataset, but cannot access the classifier he wants to attack. Our classification network is trained with SceneProof Dataset training split.

Type I attacks on SceneProof are detected reliably: Type I attacks on SceneProof on both familar and unfamiliar input data are reliabley detected without sacrificing much classifier accuracy, refer to Table 4.1, Table 4.2 and Table 4.3. In Table 4.5, we show detailed type I attack results for various inspector settings (optionally joint with confidence rejection), and various white box & black box attacking methods. From the results, we can tell that VGG19 with *ABC* inspector joint with confidence rejection defenses type I attacks the best.

4.5.6 Experimental Results: Type II Attacks

A type II attack involves a search for an adversarial example that will be (a) mislabelled and (b) not detected. A type II attack must both fool the classifier *and* sneak past the detector. We distinguish between two conditions. In non-blackbox case, the internals of the SafetyNet system are *accessible* to the attacker. Alternatively, the network may be a black box, with internal states and gradients concealed. In this case, attackers must probe with inputs and gather outputs, or build another approximate network as in [9].

Let Θ be the parameters of the neural network model, $\mathbf{x}$ be the input to the model, y be the targets associated with $\mathbf{x}$, Σ be the parameters of the inspector. We use $J(\Theta, \mathbf{x}, y)$ to represent the cost to train the neural network, and use $D(\Theta, \Sigma, \mathbf{x})$ to represent the cost from the inspector. When RBF-SVM is used as the inspector, $D(\Theta, \Sigma, \mathbf{x})$ is not differentiable with respect to $\mathbf{x}$ because of the binarization process. We use sigmoid function to smooth the binarization to make it differentiable. Our gradient descent based type II attacks use

Configuration	No Attack		Type I Attack				Type II Attack			
			White Box	Black Box			White Box	Black Box		
			GradDesc	ResNet GD	ResNet FS	BBA	GradDesc	ResNet FT	ResNet SN	BBA
	tnt	**fnf**	**faf**	**faf**	**faf**	**faf**	**faf**	**faf**	**faf**	**faf**
No Inspector	97.6	79.9	11.8	37.8	58.0	47.9	n.a	36.2	40.4	38.3
No Inspector - R	96.5	83.0	21.9	42.6	66.7	54.6	n.a	44.3	48.4	46.4
	tna	**fnr**	**far**	**far**	**far**	**far**	**far**	**far**	**far**	**far**
Subnet	81.0	26.2	64.7	86.8	98.0	92.4	50.9	92.1	88.7	90.4
ABC	83.2	33.0	74.3	91.0	94.0	92.5	55.5	93.5	90.2	91.9
Subnet - R	85.4	9.7	100	98.3	100	99.1	70.6	100	97.9	98.9
ABC - R	86.1	16.1	100	98.4	100	99.2	78.2	100	98.0	99.0

Table 4.4: Summary statistics comparing SceneProof with no inspector and the Subnet method. Because this table contains multiple settings of attacks and some of them take long time to generate, we randomly subsample 3200 images from the whole SceneProof dataset test split to calculate the results. Notice the difference to Table 4.2, which uses the whole test split. The classifier always uses VGG-19. Note that **tna** and **fnr** depend only on inspector, but not on attack (so are not repeated); and there is no **far** for no attack, because that statistic is meaningless in that case. Note that the ABC SceneProof is significantly resistant to both white and black box attacks, with good performance on natural data. The joint classification confidence rejection greatly helps to defend adversarial examples. The detailed break down for type I attacks can be found in Table 4.5 and for type II attacks can be found in Table 4.6.

gradients calculated as (λ is the parameter to adjust the relative weight of the two costs)

$$\eta = \nabla_{\mathbf{x}}(J(\Theta, \mathbf{x}, y) + \lambda D(\Theta, \Sigma, \mathbf{x})) \tag{4.2}$$

Accessible SceneProof is difficult to attack, because searching is made difficult by the quantization procedure and by the narrow basis functions in the RBF-SVM, so we smooth the quantization operation and the RBF-SVM kernel operation. Smoothing is essential to make the search tractable, but can significantly misapproximate SafetyNet (which is what makes attacks hard). Our smoothing attack uses a sigmoid function with parameter λ to simulate the quantization process. We also help the search process by increasing the size of the RBF parameter σ to form smoother gradients. Then we obtain gradients by Equation 4.2. Even after smoothing the objective function, attacks based on gradients tend to fail, likely because it is hard to make an effective tradeoff between easy search and approximation.

Type II attacks on accessible SceneProof fail: Table 4.4 includes summary of type I and type II, blackbox and non-blackbox attacking results on SceneProof dataset, while Table 4.6 includes the details for type II attacks with various settings. From the results, we can tell that joint rejection approach increased the model's robustness, and that our ABC inspector is much better at defending white box type II attacks. In sum, we can tell that our VGG19 with ABC inspector and joint rejection approach is the most robust architecture to

Type I attacks		White Box	Black Box		
		GradDesc	ResNet GradDesc	ResNet FastSign	Black Box Average
VGG19 / nodet	**faf**	11.8	37.8	58.0	47.9
	taf	88.0	24.7	64.1	44.4
VGG19 / Subnet Det	**far**	64.7	86.8	98.0	92.4
	F_u	17.2	6.0	25.4	15.7
	F_d	81.2	78.4	67.1	72.7
VGG19 / Det ABC	**far**	74.3	91.0	94.0	92.5
	F_u	8.7	5.6	22.1	13.8
	F_d	90.0	82.0	72.2	77.1
VGG19 / nodet - R	**faf**	21.9	42.6	66.7	54.6
	taf	92.1	33.1	66.2	49.6
VGG19 / Subnet Det - R	**far**	100	98.3	100	99.1
	F_u	7.1	3.91	21.5	12.7
	F_d	92.9	90.6	71.4	81.0
VGG19 / Det ABC - R	**far**	100	98.4	100	99.2
	F_u	0.39	4.6	16.7	10.6
	F_d	99.6	90.0	78.6	84.3

Table 4.5: This table shows a VGG19 network without detector (VGG19 / nodet), with the detection subnetwork of [81] (VGG19 / Subnet Det), and SafetyNet with ABC detector (where we have an RBF-SVM on each of fc7, fc6 and pool5, and declare an adversary when any detector responds). The table is gather by attacking a randomly selected subset of 3200 images from the whole SceneProof dataset test split (115K samples). This table shows results for type I whilte box attacks and black box attacks. For **white box attacks**, attacker uses a gradient descent based attacking method (L-BFGS) to attack the accessible VGG19 based SceneProof classifier. When there is no adversarial detector, the SceneProof classifier's classification accuracy significantly decreased compared to Table 4.4 non attack data. When adversarial detectors are included, they rejects most of the attacks and have high **far** and F_d, and low F_u. We can tell that SafetyNet *ABC* detectors are significantly better than Subnet detector in the type I white box attack setting. For **black box attacks**, attacker uses either GradDesc (L-BFGS) or FastSign to attack an accessible ResNet, and then transfer these adversarial examples to the VGG19 based SceneProof networks. GradDesc attacks are more stronger than FastSign attacks. When there is no adversarial detector, the SceneProof classifier's classification accuracy significantly decreased compared to Table 4.4 non attack data. When adversarial detectors are included, they rejects most of the attacks, and have high **far** and F_d, and low F_u. Compared to white box attacks, we can tell that the adversarial detectors rejects less type I black box attack samples than type I white box attack samples (lower F_d), but rejects more false label type I black box attack samples (higher **far**). What's more, the joint classification confidence rejection greatly increase the network's defend ability against adversarial examples. In sum, SafetyNet *ABC* detectors are better than Subnet detector in the type I attacks, and joint classification confidence rejection helps.

various type II attacks.

Gradient descent type II attacks		White Box GradDesc	Black Box ResNet FT nodet	Black Box ResNet Subnet Det	Black Box Black Box Average
VGG19 / nodet	**faf**	n.a	36.2	40.4	38.3
	taf	n.a	47.2	46.2	46.7
VGG19 / Subnet Det	**far**	50.9	92.1	88.7	90.4
	F_u	21.9	14.1	13.8	13.9
	F_d	47.1	77.1	76.2	76.6
VGG19 / Det ABC	**far**	55.5	93.5	90.2	91.9
	F_u	11.8	8.4	9.0	8.7
	F_d	57.5	84.1	82.9	83.5
VGG19 / nodet - R	**faf**	n.a	44.3	48.4	46.4
	taf	n.a	59.0	57.0	58.0
VGG19 / Subnet Det - R	**far**	70.6	100	97.9	98.9
	F_u	15.7	15.7	14.9	15.3
	F_d	51.4	80.4	80.4	80.4
VGG19 / Det ABC - R	**far**	78.2	100	98.0	99.0
	F_u	5.4	8.2	8.5	8.3
	F_d	64.4	88.0	87.0	87.5

Table 4.6: The table layouts and experiment settings are same as Table 4.5, except that type II gradient descent white box attacks do not apply on VGG19 / nodet case (because there is no type II gradient descent attacks when there is no detector). We show both white box attacks and black box attacks for type II attacks. For type II white box attacks, attacker uses a gradient descent based attacking method (L-BFGS) to attack the accessible VGG19 based SceneProof classifier along with a detector. We can also tell that SafetyNet *ABC* detectors are significantly better than Subnet detector in the type II white box attack setting. For type II black box attacks, attacker uses GradDesc (L-BFGS) to attack accessible networks along with the detectors (if exist), and then transfer these adversarial examples to the VGG19 based SceneProof networks. ResNet FT nodet finetunes a ResNet network with adversarial examples labelled false, there is no detector involved and we generate L-BFGS type I adversarials to ResNet FT to simulate type II adversarials to ResNet; ResNet Subnet Det uses a ResNet network with the detection subnetwork, and we generate L-BFGS type II adversarials. ResNet FT nodet and ResNet Subnet Det performs similarly. When there is no adversarial detector, the SceneProof classifier's classification accuracy decreased significantly compared to Table 4.4 non attack data. When adversarial detectors are included, they rejects most of the attacks and have high **far** and F_d and low F_u. We can also tell that the adversarial detectors rejects more type II black box attack samples than type II white box attack samples, so there is a significant decrease in F_d and **far**. What's more, the joint classification confidence rejection greatly increase the network's defend ability against adversarial examples. In sum, detectors are more likely to reject type II black box attacks, SafetyNet *ABC* detectors are better than Subnet detector in type II attacks, and joint classification confidence rejection helps.

4.6 THEORY: BARS AND P-DOMAINS

4.6.1 Definition

We construct one possible explanation for adversarial examples that successfully explains (a) the phenomenology and (b) why SafetyNet works. We have a network with N layers

of ReLU's, and study $y_i^{(k)}(\mathbf{x})$, the values at the output of the k'th layer of ReLUs. This is a piecewise linear function of $\mathbf{x}$. Such functions break up the input space into *cells*, at whose boundaries the piecewise linear function changes (i.e. is only C^0). Now assume that for some $y_i^{(k)}(\mathbf{x})$ there exist *p-domains* (union of cells) $\mathcal{D}$ in the input space such that: (a) there are no or few examples in the p-domain; (b) the measure of $\mathcal{D}$ under $P(X)$ is small; (c) $|y_i^{(k)}(\mathbf{x})|$ is large inside $\mathcal{D}$ and small outside $\mathcal{D}$. We will always use the term "p-domain" to refer to domains with these properties. We think that the total measure of all p-domains under $P(X)$ is small.

By construction, ReLU networks can represent such p-domains. We construct a p-domain using a basis function with small support. $\mathsf{R}(\mathbf{u})$ denote a ReLU applied to $\mathbf{u}$. We have *basic bar function* ϕ.

$$\phi(\mathbf{x}; i, s, \epsilon) = \frac{1}{\epsilon} \begin{pmatrix} \mathsf{R}((x_i - s) + \epsilon) - \\ 2\mathsf{R}((x_i - s)) + \\ \mathsf{R}((x_i - s) - \epsilon) \end{pmatrix}$$

where ϕ has support when $|x_i - s| < \epsilon$ and has peak value 1. For an index set $\mathcal{I}$ with cardinality $\#\mathcal{I}$ and vectors $\mathbf{s}$, ϵ, we write *bar function* b as

$$b(\mathbf{x}; \mathcal{I}, \mathbf{s}, \epsilon) = \mathsf{R}(\left(\sum_{i \in \mathcal{I}} \phi(\mathbf{x}; i, s_i, \epsilon_i) - \#\mathcal{I} + 1\right))$$

where b has support when $\|\mathbf{x}_\mathcal{I} - \mathbf{s}_\mathcal{I}\|_1 < \epsilon_\mathcal{I}$. Figure 4.4 illustrates these functions. It is clear that a CNN can encode bars and weighted sums of bars, and that for at least $k \geq 2$ every $y_i^{(k)}$ could in principle be a bar function. Appropriate choices of $\mathbf{s}$, ϵ and $\mathcal{I}$ choose the location and support of the bar and so can produce bars which have low measure under $P(X)$. Now the functions presented to the softmax layer are a linear combination of the $y_i^{(N)}(\mathbf{x})$. This means that with choice of weight and parameters, a bar can appear at this level, and create a p-domain.

Figure 4.4: Simple example bar functions on the x, y plane, where black is 0 and white is 1. **Left:** $\phi(\mathbf{x}; 1, 0, 1)$ (i.e. a bar in x, independent of y); **center:** $\phi(\mathbf{x}; 2, 0, 1)$; and **right:** $b(\mathbf{x}; \{1, 2\}, \mathbf{0}, 1)$.

4.6.2 Property

We expect such p-domains to have several important properties.

Adversarial fertility

P-domains can be used to make adversarial examples by choosing a point in a p-domain close to $\mathbf{x}$. Because there are no or few examples in the p-domain, the loss may not cause the classifier to control the maximum value attained by $y_i^{(k)}(\mathbf{x})$ in this p-domain; and the large range of values inside the p-domain can be used to change the values in layers upstream of k, by moving the example around the p-domain.

Generalization-neutral

The requirement that p-domains have small measure in $P(X)$ means that both train and test examples are highly unlikely to lie in p-domains. A system with p-domains could generalize well without being immune to adversarial examples. Some subset of p-domains are likely **findable by LBFGS.** Consider the gradient of $y_i^{(N)}(\mathbf{x})$ with respect to $\mathbf{x}$ in two cells separated by a boundary, where some ReLU changes state, weight decay encourages a relatively small change in gradient over these boundaries. If cells neighboring a p-domain have no or few examples in them, we can expect that the gradient change within cell is small too and a second order approximation of $y_i^{(N)}(\mathbf{x})$ could be reliable. We also expect cells to be small, so search and entering a p-domain are possible and requires crossing multiple cell boundaries, which means many changes in ReLU activation. This argument suggests p-domains present **odd patterns of ReLU activation**, particularly in p-domains where some of the $y_i^{(k)}(\mathbf{x})$ are large in the absence of examples.

4.6.3 Explanation

Why p-domains could exist: As Zhang *et al.* point out, the number of training examples available to a typical modern network is small compared to the relative capacity of deep networks [82]. For example, excellent training error is obtainable for randomly chosen image labels [82]. We expect that $y_i^{(N)}(\mathbf{x})$ will have a number of cells that is exponential in the dimension of $\mathbf{x}$, ensuring that the vast majority of cells lack any example. However, the weight decay term is not sufficient to ensure that $y_i^{(N)}$ is zero in these cells. Overshoot by stochastic gradient descent, caused by poor scaling in the loss, is the likely reason that $y_i^{(N)}(\mathbf{x})$ has support in these cells. Szegedy *et al.* demonstrate that, in practice, ReLU layers

can have large norm as linear operators, despite weight decay (see [63], sec. 4.3), so large values in p-domains are plausible. This large norm is likely to be the result of overshoot. Recall that the value of $y_i^{(N)}(\mathbf{x})$ is determined by the *product* of numerous weights, so in some locations in $\mathbf{x}$, the value of $y_i^{(N)}$ could be large, which is a result of multiple layer norms interacting poorly.

4.6.4 Influence

An alternative to attacking by search using smoothed RBF gradients is as follows. One might pass an example through the main classifier, determine what code it had, then seek an adversarial example that produces that code (and so must fool the RBF-SVM). We sketch a proof that the optimization problem is extremely difficult. Choose some threshold $t > 0$. We use $b_t(u)$ for the function that binarizes its argument with t. Assume we have at least one unit $y_i^{(k)}$ that encodes a weighted sum of bar functions. We wish to create an adversarial example $\mathbf{a}(\mathbf{x}^*)$ that (a) meets criteria for being adversarial and (b) ensures that $b_t(y_i^{(k)}(\mathbf{a}))$ takes a prescribed value (either one or zero). The feasible set for this constraint can be disconnected (e.g.a sum of the bump functions of Figure 4.4 (right)), and so need not be convex, implying that the optimization problem is intractable. As a simple example, the following constraint set is disconnected for $\epsilon < 1/2$

$$\{\mathbf{x} \mid b_t(b(\mathbf{x}; 1, \mathbf{0}, \epsilon) + b(\mathbf{x}; 1, \mathbf{1}, \epsilon)) = 1\}.$$

4.7 DISCUSSION

We have described a method to produce a classifier that identifies and rejects adversarial examples. Our SafetyNet is able to reject adversarial examples that come from attacking methods not seen in training data. We have shown that it is hard to produce an example that (a) is mislabeled and (b) is not detected as adversarial by SafetyNet. We have sketched one possible reason that SafetyNet works, and is hard to attack. Many interesting problems are opened by our work, and we provides lots of insights into the mechanism that neural network works.

SaferNet: There might be some better architecture than our SafetyNet, whose objective function is harder to optimize. The ideal case would be an architecture that forces the attacker to solve a hard discrete optimization problem which does not naturally admit smoothing.

Neural network pruning: Our work suggests that networks behave poorly for input

space regions where no data has been seen. We speculate that this behavior could be discouraged by a post-training pruning process, which removes neurons, paths or activation patterns not touched by training data.

Explicit management of overshoot during training: we have explained adversarial examples using p-domains, which is the result of poor damping of weights during training. We speculate that constructing adversarial examples during training, by identifying locations where this damping problem occurs and exploiting structural insights into network behavior, could control the adversarial sample problem (rather than just using adversarial examples as training data).

CHAPTER 5: CONCLUSION

The thesis focuses on using images and depths for high resolution, low latency sensing, and then using these sensing techniques to build security applications. High quality depth sensing is the key to various types of applications, and it is very difficult to acquire such depth stream via pure hardware approach.

The thesis first proposes a sensor fusion approach, which combines depth camera and color camera. The method produces detailed high resolution depth maps from aggressively subsampled depth measurements. Our method fully uses the relationship between image segmentation boundaries and depth boundaries. It uses an image combined with a low resolution depth map. 1) The image is segmented with the guidance of sparse depth samples. 2) Each segment has its depth field reconstructed independently using a novel smoothing method. 3) For videos, time-stamped samples from near frames are incorporated. The thesis shows reconstruction results of super resolution from x4 to x100, while previous methods mainly work on x2 to x16. The method is tested on four different datasets and six video sequences, covering quite different regimes, and it outperforms recent state of the art methods quantitatively and qualitatively. We also demonstrate that depth maps produced by our method can be used by applications such as hand trackers, while depth maps from other methods have problems.

After improving the spatial resolution of the depth stream, the thesis proposes an approach to combine off-the-shelf hardware to create a high frame rate, low latency configurable depth stream. Hybrid HFR Depth is more powerful than a purely software-based approach, and less demanding than a customized hardware approach. We presented detailed specifications of Hybrid HFR Depth to demonstrate the capability and flexibility of the system needed to address a variety of applications. Our evaluation of Hybrid HFR Depth shows that the quality of the generated depth image is good enough to benefit three demonstrated interactive applications.

With the combined depth and image, we build a SceneProof system that can reliably detect whether an image is a picture of a real scene or not. It applies to images captured with depth maps (RGBD images) and checks if a pair of image and depth map is consistent. It relies on the relative difficulty of producing naturalistic depth maps for images in post processing. We demonstrate that our system is robust to adversarial examples built from currently known attacking approaches, such as DeepFool. The system depends on our novel network architecture SafetyNet, which utilizes a RBF-SVM inspector to classify adversarial examples based on the internal activations of the neural networks. Our construction suggests

some insights into how deep networks work. We provide a reasonable analyses that our construction is difficult to defeat, and show experimentally that our method is hard to defeat using several standard networks and datasets.

REFERENCES

[1] B. Jones, R. Sodhi, D. Forsyth, B. Bailey, and G. Maciocci, "Around device interaction for multiscale navigation," in *MobileHCI*, 2012.

[2] R. Sodhi, B. Jones, D. Forsyth, B. Bailey, and G. Macciocci, "Bethere: 3d mobile collaboration with spatial input," in *SIGCHI*, 2013.

[3] I. Oikonomidis, N. Kyriazis, and A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect," in *BMVC 2011*. BMVA, 2011.

[4] N. Kyriazis, I. Oikonomidis, and A. Argyros, "A gpu-powered computational framework for efficient 3d model-based vision," ICS-FORTH, Tech. Rep. TR420, July 2011.

[5] D. Ferstl, C. Reinbacher, R. Ranftl, M. Ruether, and H. Bischof, "Image guided depth upsampling using anisotropic total generalized variation," in *Proceedings International Conference on Computer Vision (ICCV), IEEE*, December 2013.

[6] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[7] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.

[8] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2574–2582.

[9] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against deep learning systems using adversarial examples," *arXiv preprint arXiv:1602.02697*, 2016.

[10] K. He, J. Sun, and X. Tang, "Guided image filtering," in *Proceedings of the 11th European Conference on Computer Vision: Part I*, ser. ECCV'10. Berlin, Heidelberg: Springer-Verlag, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1886063.1886065 pp. 1–14.

[11] J. Diebel and S. Thrun, "An application of markov random fields to range sensing," in *NIPS*. Cambridge, MA: MIT Press, 2005.

[12] J. Lu, D. Min, R. S. Pahwa, and M. N. Do, "A revisit to MRF-based depth map super-resolution and enhancement." in *ICASSP'11*, 2011, pp. 985–988.

[13] J. Zhu, L. Wang, R. Yang, and J. Davis, "Fusion of time-of-flight depth and stereo for high accuracy depth maps."

[14] J. Park, H. Kim, Y.-W. Tai, M. S. Brown, and I. Kweon, "High quality depth map upsampling for 3d-tof cameras," *Computer Vision, IEEE International Conference on*, vol. 0, pp. 1623–1630, 2011.

[15] O. Mac Aodha, N. D. Campbell, A. Nair, and G. J. Brostow, "Patch based synthesis for single depth image super-resolution," in *ECCV (3)*, 2012, pp. 71–84.

[16] S. Schuon, C. Theobalt, J. Davis, and S. Thrun, "Lidarboost: Depth superresolution for tof 3d shape scanning," in *CVPR'09*, 2009, pp. 343–350.

[17] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '11. New York, NY, USA: ACM, 2011. [Online]. Available: http://doi.acm.org/10.1145/2047196.2047270 pp. 559–568.

[18] S. A. Gudmundsson, H. Aanaes, and R. Larsen, "Fusion of stereo vision and time-of-flight imaging for improved 3d estimation," *Int. J. Intell. Syst. Technol. Appl.*, vol. 5, no. 3/4, pp. 425–433, Nov. 2008. [Online]. Available: http://dx.doi.org/10.1504/IJISTA.2008.021305

[19] M. Mahmoudi and G. Sapiro, "Sparse representations for range data restoration," *IEEE Transactions on Image Processing*, vol. 21, no. 5, pp. 2909–2915, 2012.

[20] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Comput. Graph. Appl.*, vol. 22, no. 2, pp. 56–65, Mar. 2002. [Online]. Available: http://dx.doi.org/10.1109/38.988747

[21] I. Tosic and S. Drewes, "Learning joint intensity-depth sparse representations," *IEEE Transactions on Image Processing*, vol. 23, no. 5, pp. 2122–2132, 2014.

[22] M. Kiechle, S. Hawe, and M. Kleinsteuber, "A joint intensity and depth co-sparse analysis model for depth map super-resolution," *Computer Vision, IEEE International Conference on*, vol. 0, pp. 1545–1552, 2013.

[23] L.-F. Yu, S.-K. Yeung, Y.-W. Tai, and S. Lin, "Shading-based shape refinement of rgb-d images," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[24] Y. Han, J.-Y. Lee, and I. S. Kweon, "High quality shape from a single rgb-d image under uncalibrated natural illumination," in *ICCV*, 2013.

[25] J. Li, Z. Lu, G. Zeng, R. Gan, and H. Zha, "Similarity-aware patchwork assembly for depth image super-resolution," June 2014.

[26] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vision*, vol. 59, no. 2, pp. 167–181, Sep. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000022288.19776.77

[27] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient hierarchical graph based video segmentation," *IEEE CVPR*, 2010.

[28] T. Brox and J. Malik, "Large displacement optical flow: descriptor matching in variational motion estimation," pp. 500–513, 2011. [Online]. Available: http://lmb.informatik.uni-freiburg.de//Publications/2011/Bro11a

[29] K. Karsch, C. Liu, and S. B. Kang, "Depthtransfer: Depth extraction from video using non-parametric sampling," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2014.

[30] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374.

[31] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering the spatial layout of cluttered rooms," 2009.

[32] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vision*, vol. 47, no. 1-3, pp. 7–42, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1014573219977

[33] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *CVPR*, ser. CVPR'03. Washington, DC, USA: IEEE Computer Society, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=1965841.1965865 pp. 195–202.

[34] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[35] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)*, ser. Part IV, LNCS 7577, A. Fitzgibbon et al. (Eds.), Ed. Springer-Verlag, Oct. 2012, pp. 611–625.

[36] Q. Yang, R. Yang, J. Davis, and D. Nistér, "Spatial-depth super resolution for range images." in *CVPR*, 2007.

[37] S. Hawe, M. Kleinsteuber, and K. Diepold, "Analysis operator learning and its application to image reconstruction," *Image Processing, IEEE Transactions on*, vol. 22, no. 6, pp. 2138–2150, 2013.

[38] K. Karsch, K. Sunkavalli, S. Hadap, N. Carr, H. Jin, R. Fonte, M. Sittig, and D. Forsyth, "Automatic scene inference for 3d object compositing," *ACM Trans. Graph.*, vol. 33, no. 3, June 2014.

[39] H. Xia, R. Jota, B. McCanny, Z. Yu, C. Forlines, K. Singh, and D. Wigdor, "Zero-latency tapping: using hover information to predict touch locations and eliminate touchdown latency," in *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 2014, pp. 205–214.

[40] J. Knibbe, H. Benko, and A. D. Wilson, "Juggling the effects of latency: motion prediction approaches to reducing latency in dynamic projector-camera systems."

[41] K. Kitani, K. Horita, and H. Koike, "Ballcam!: dynamic view synthesis from spinning cameras," in *Adjunct proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 2012, pp. 87–88.

[42] M. Schmidt, K. Zimmermann, and B. Jähne, "High frame rate for 3d time-of-flight cameras by dynamic sensor calibration," in *Computational Photography (ICCP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–8.

[43] J. Stuhmer, S. Nowozin, A. Fitzgibbon, R. Szeliski, T. Perry, S. Acharya, D. Cremers, and J. Shotton, "Model-based tracking at 300hz using raw time-of-flight observations," in *ICCV 2015 - International Conference on Computer Vision*. IEEE Institute of Electrical and Electronics Engineers, October 2015. [Online]. Available: https://www.microsoft.com/en-us/research/publication/model-based-tracking-at-300hz-using-raw-time-of-flight-observations/

[44] S. Ryan Fanello, C. Rhemann, V. Tankovich, A. Kowdle, S. Orts Escolano, D. Kim, and S. Izadi, "Hyperdepth: Learning depth from structured light without matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5441–5450.

[45] G. Papadakis, K. Mania, and E. Koutroulis, "A system to measure, control and minimize end-to-end head tracking latency in immersive simulations," in *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*. ACM, 2011, pp. 581–584.

[46] H. Koike and H. Yamaguchi, "Lumospheres: real-time tracking of flying objects and image projection for a volumetric display," in *Proceedings of the 6th Augmented Human International Conference*. ACM, 2015, pp. 93–96.

[47] K. Okumura, H. Oku, and M. Ishikawa, "High-speed gaze controller for millisecond-order pan/tilt camera," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 6186–6191.

[48] J. Lu and D. Forsyth, "Sparse depth super resolution," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2245–2253.

[49] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *European conference on computer vision*. Springer, 2004, pp. 25–36.

[50] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect."

[51] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei et al., "Accurate, robust, and flexible real-time hand tracking," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 3633–3642.

[52] J. O. Wobbrock, A. D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes," in *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM, 2007, pp. 159–168.

[53] B. R. Jones, H. Benko, E. Ofek, and A. D. Wilson, "Illumiroom: peripheral projected illusions for interactive experiences," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2013, pp. 869–878.

[54] B. Jones, R. Sodhi, M. Murdock, R. Mehra, H. Benko, A. Wilson, E. Ofek, B. MacIntyre, N. Raghuvanshi, and L. Shapira, "Roomalive: magical experiences enabled by scalable, adaptive projector-camera units," in *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 2014, pp. 637–644.

[55] H. Benko, A. D. Wilson, and F. Zannier, "Dyadic projected spatial augmented reality," in *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 2014, pp. 645–655.

[56] R. Jota, A. Ng, P. Dietz, and D. Wigdor, "How fast is fast enough?: a study of the effects of latency in direct-touch pointing tasks," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2013, pp. 2291–2300.

[57] T. Kroeger, R. Timofte, D. Dai, and L. Van Gool, "Fast optical flow using dense inverse search," in *European Conference on Computer Vision*. Springer, 2016, pp. 471–488.

[58] H. Farid, "How to detect faked photos," *American Scientist*, 2017.

[59] H. Farid, "Exposing photo manipulation with inconsistent reflections." *ACM Trans. Graph.*, vol. 31, no. 1, p. 4, 2012.

[60] E. Kee, J. F. O'Brien, and H. Farid, "Exposing photo manipulation with inconsistent shadows," *ACM Transactions on Graphics (ToG)*, vol. 32, no. 3, p. 28, 2013.

[61] H. Farid, "Photo forensics," *MIT Press*, 2016.

[62] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *CVPR*, 2015.

[63] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[64] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 372–387.

[65] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *arXiv preprint arXiv:1608.04644*, 2016.

[66] J. Su, D. V. Vargas, and S. Kouichi, "One pixel attack for fooling deep neural networks," *arXiv preprint arXiv:1710.08864*, 2017.

[67] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[68] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 15–26.

[69] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially transformed adversarial examples," *arXiv preprint arXiv:1801.02612*, 2018.

[70] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *arXiv preprint arXiv:1611.02770*, 2016.

[71] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.

[72] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," *arXiv preprint arXiv:1610.08401*, 2016.

[73] D. Krotov and J. J. Hopfield, "Dense associative memory is robust to adversarial inputs," *arXiv preprint arXiv:1701.00939*, 2017.

[74] J. Lu, H. Sibai, E. Fabry, and D. Forsyth, "Standard detectors aren't (currently) fooled by physical adversarial stop signs," *arXiv preprint arXiv:1710.03337*, 2017.

[75] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 582–597.

[76] N. Carlini and D. Wagner, "Defensive distillation is not robust to adversarial examples."

[77] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," *arXiv preprint arXiv:1711.00117*, 2017.

[78] J. Lu, H. Sibai, and E. Fabry, "Adversarial examples that fool detectors," *arXiv preprint arXiv:1712.02494*, 2017.

[79] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing with virtual adversarial training," *arXiv preprint arXiv:1507.00677*, 2015.

[80] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *German Conference on Pattern Recognition*. Springer, 2014, pp. 31–42.

[81] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff, "On detecting adversarial perturbations," *arXiv preprint arXiv:1702.04267*, 2017.

[82] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *ICLR 2016*, 2016.