# HyperDepth: Learning Depth from Structured Light Without Matching Supplementary Material

Sean Ryan Fanello* Christoph Rhemann* Vladimir Tankovich
Adarsh Kowdle Sergio Orts Escolano David Kim Shahram Izadi

Microsoft Research

## Abstract

*This document accompanies the paper: HyperDepth: Learning Depth from Structured Light Without Matching. We first provide an analysis of the label space that aids in understanding the design choices in our algorithm. In particular we show that our choice of using one classifier per line does not increase the model complexity and simplifies the recognition problem. Furthermore, we present additional experimental results. Note that information given in this document is not necessary to understand the content of the main paper.*

## 5. Label Space Analysis

In this Section we provide an analysis of the label space to model all possible disparities with a precision of $0.05$ pixels. Considering a $1280 \times 1024$ image $I$, each pixel in the image has a particular class label $c$, totaling $1280$ classes per line. If we want to achieve $0.05$ subpixel accuracy, we need $20$ additional classes between each class $i$ and $i+1$. So in total we have $C = 1280 \times 20 = 25.600$ possible labels per line. The minimum depth of the tree required to model all the labels within a line is $15$, which gives us $2^{15} = 32.768$ possible assignment. For our HyperDepth algorithm we use one classifier for each line. Therefore we need additional $1024$ classifiers, which leads of $2^{25}$ nodes for all classifiers. In practice, due to imaging noise and pattern variability, we need a higher number of nodes. Multiple trees help to mitigate these effects.

Note that using one classifier per line simplifies the problem because less classes have to be disambiguated. Moreover it handles repetition of the same the local structure of the pattern in different lines. Lets assume that there are no repetition involved and we can use a single tree to model the whole image. The total number of labels is $1280 \times 1024$, one for each pixel, therefore we have $1.310.720$ possible labels. To reach the same subpixel accuracy ($0.05$), we have to increase the label space by a factor of 20, reaching $26.214.400$ possible labels. The minimum depth of the tree required to model all the possible labels is $25$, giving $2^{25} = 33.554.432$ possible assignments. This shows that using one single classifier for the whole image would require the same amount of memory

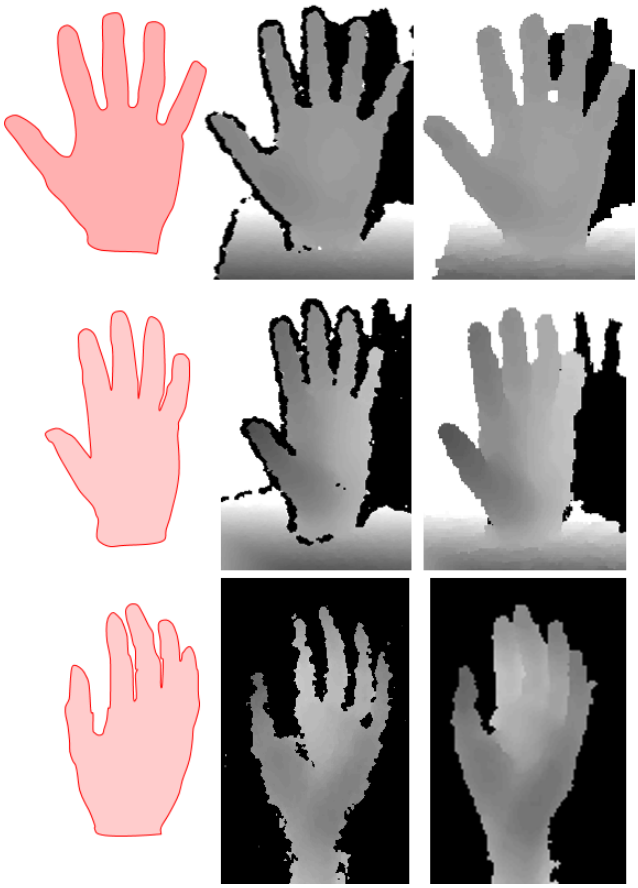

Figure 1. **Edge Fattening Comparison**. We compare our method (**middle**) and Kinect (**right**) to the hand-labeled ground truth (**left**). Our method better invalidates edges (first two rows). However in some cases we shrink the object (bottom images).

*Authors equally contributed to this work.

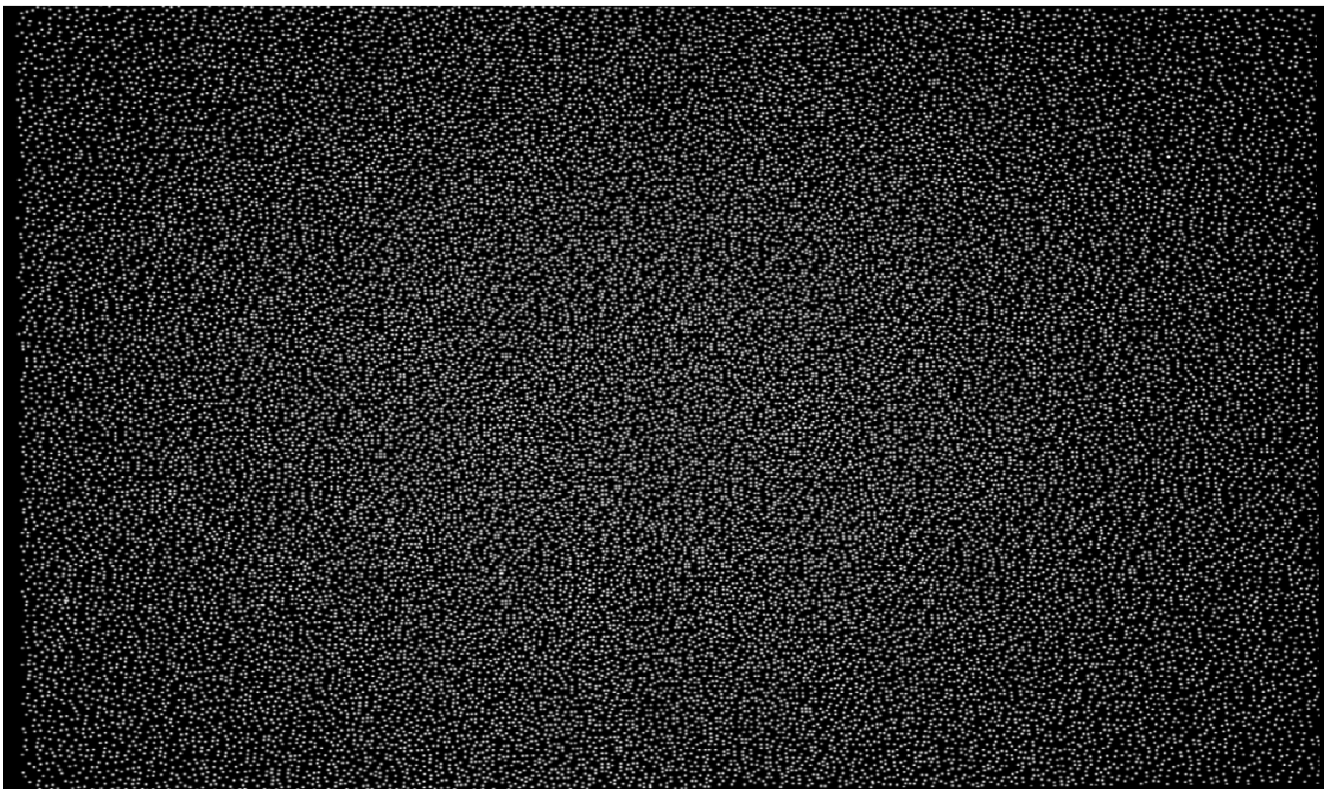

Figure 2. **Recovered Reference Pattern**. Recovered Reference Pattern with the calibration procedure described in Section 2.3.

of a classifier per line. Moreover this per-image classifier has to learn to disambiguate $26.214.400$ labels, versus $25.600$ for the per-scanline classifier, making the problem much harder. This shows that our choice of using one classifier per line does not increase the model complexity and simplifies the recognition problem.

We also tried to group multiple adjacent lines together, i.e. a random forest predicts the output for $l$ consecutive lines. However due to the high class intra-variability we noticed worse results given the same model complexity of a classifier per line.

We note that if the repetitions in the pattern are known they can be exploited to reduce the label space. In particular, let us assume that the reference pattern repeats exactly the same structure every $l$ lines, we reduce the output space to $1280 \times 20 \times l$, which saves us a factor of $\frac{H}{l}$ in the model complexity, where $H$ is the height of the image.

## 6. Disparity Direct Regression Analysis

One may argue that direct regression of the disparity $d$ may be more efficient. In the following we show that the complexity of direct regression is higher compared to our model. We recall that in our framework the disparity $d$ can be obtained via the mapping $d = c - x$, where $c$ is the continuous class label and $x$ is the $x$ coordinate of the pixel that we classify. In order to predict all the possible disparities, a direct regression has to learn this mapping as well. Assuming we have $211$ disparities in total (like in the Kinect reference pattern), with a subpixel precision of $0.05$ we have to predict $4220$ possible labels. Moreover, these disparities depend on the current $x$ position of the pixel, leading to a total number of labels of $211 \times 20 \times 1280 = 5.401.600$ which requires a tree model with 23 levels. This assumes no repetition in the reference pattern, in reality, due to repetitions, the same structure of the pattern at the same coordinate $x$ can lead to two different disparities $d_1$ and $d_2$. In other words, there exists a class label $c$ and a $x$ coordinate such that $d_1 = c - x$ and $d_2 = c - x$ with $d_1 \neq d_2$, which leads to inconsistent results. To resolve this ambiguity, we have to consider also all the possible $y$ coordinates, increasing the label space from $5.401.600$ to $211 \times 20 \times 1280 \times 1024 = 5.531.238.400$, which results in an unfeasible model complexity. This shows that leveraging the knowledge $d = c - x$ is more efficient than attempting to learn directly mapping directly.

## 7. Edge Fattening

Let us consider Fig. 1 which illustrates the robustness of our approach against edge fattening. We visually compare the results of our method (middle columns) and Kinect (right column) against the ground truth outline on the left (generated by manually tracing the boundary of the hand in

the IR images). Notice that in the first two rows HyperDepth nicely preserves the object boundaries, whereas Kinect fattens and merges the fingers. This is the case because our invalidation is are based on probabilities, which lead to more invalidation on object boundaries. A failure cases for our approach is shown in the bottom row where we shrink the object contours too much.

## 8. Additional Evaluations

In Fig. 2, we show the IR reference pattern recovered via the calibration procedure described in Section 2.3. We also provide additional evaluations for the object scanning scenario. We followed the same protocol of Section 3.3. Results are depicted in Fig. 3. HyperDepth and PatchMatch reconstructions show higher level of details. Notice that for some objects RealSense F200 was not able to recover the whole structure. For example the ”head” object was too shiny and most of the depthmaps were invalidated. The ”phone” object instead was very dark and, in order to reconstruct it, we had to place the RealSense F200 camera at a very short distance (20 cm away), which gives an advantage to this sensor.

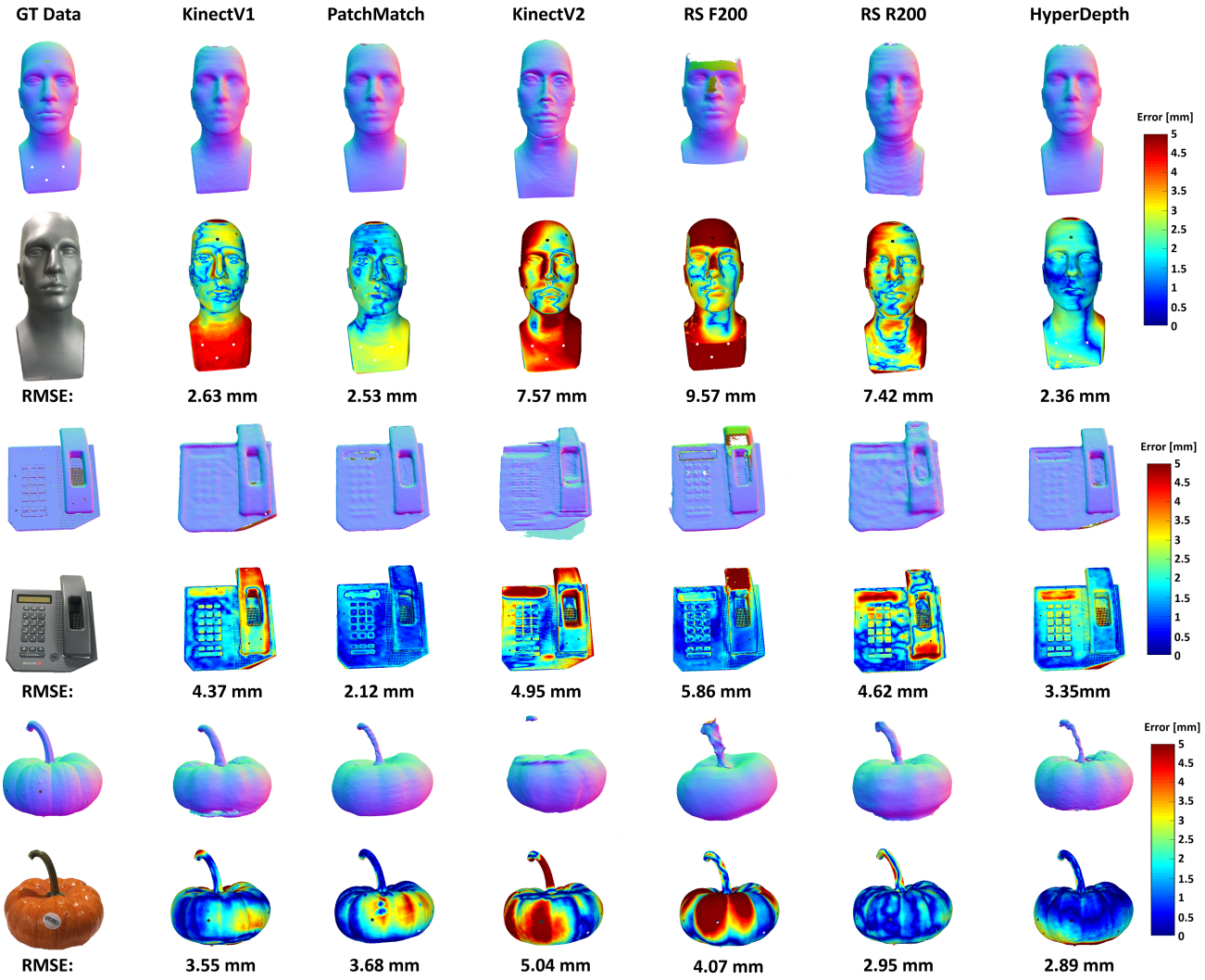


Figure 3. **Object Scanning - Additional Results**. We show here additional comparisons for the object scanning task. The same sensors of Section 3.3 have been compared.