Upsampling Range Data in Dynamic Environments

Jennifer Dolson Jongmin Baek Christian Plagemann Sebastian Thrun
Dept. of Computer Science
Stanford University

Abstract

We present a flexible method for fusing information from optical and range sensors based on an accelerated high-dimensional filtering approach. Our system takes as input a sequence of monocular camera images as well as a stream of sparse range measurements as obtained from a laser or other sensor system. In contrast with existing approaches, we do not assume that the depth and color data streams have the same data rates or that the observed scene is fully static. Our method produces a dense, high-resolution depth map of the scene, automatically generating confidence values for every interpolated depth point. We describe how to integrate priors on object motion and appearance and how to achieve an efficient implementation using parallel processing hardware such as GPUs.

1. Introduction

High resolution depth images are useful in computer vision applications. For example, a depth map at the resolution of a camera image simplifies image segmentation, a first step for many tracking, classification, and recognition algorithms. Depth information can also be helpful for scene exploration and visualization of image data. Currently, there are many active and passive sources of depth information. We focus on scanning laser rangefinders, since they are the only viable sensors for high-resolution range sensing in outdoor environments. Other classes of active sensors like flash lidars do not work in bright sunlight, or at long range. Passive sources of depth information, such as stereo vision, have made impressive progress, but at practical camera resolutions and baselines they do not yet provide the necessary depth accuracy at long ranges.

Although scanning laser rangefinders have become prevalent in ranging tasks, dense depth recovery at an arbitrary point in time, such as when a given camera frame was recorded, is an unsolved problem in dynamic environments. First, laser range measurements are inherently sparse. Second, the data acquisition rate is usually less than that of an optical camera. Figures 1 and 2 illustrate the problem of pairing a camera and a scanning laser. If the scene is not static, laser returns recorded at time $t + \Delta$ are hard to correspond to image pixels at time t, which leads to inaccuracies in a naively-constructed depth map. Prior work in creation of depth maps assumes the data acquisition time for the range device is negligible and/or the scene is static.

Figure 1. For any given camera frame, we can recover an accurate, dense depth map. The depth map shown in the lower panel corresponds to a single camera frame from a sequence of highway images, recorded from a mobile platform. Intensity in this visualization is proportional to the magnitude of depth at each pixel location. We can generate a depth map for an arbitrary camera frame even if the coincident depth data is sparse, or missing, as shown in the second panel, using depth data from neighboring frames.

We present a Gaussian framework that treats data as an input stream, more accurately reflecting different data rates of different sensors. Our method projects depth and image data into a high-dimensional space, and processes the data using accelerated Gaussian interpolation. Sensors may have their own, unrelated data acquisition rates, as long as data from each sensor is timestamped with a common clock. The use of our framework enables the upsampling of information spatially and temporally, potentially matching the

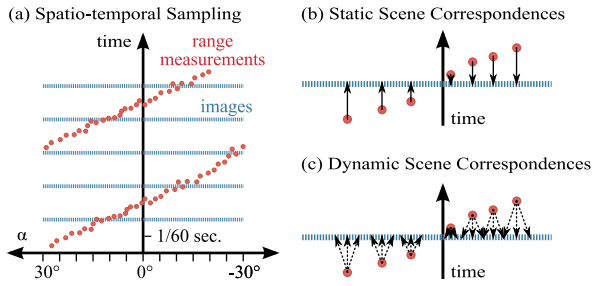

Figure 2. The fusion of image and range information is difficult because different parts of the scene are observed at different points in time. Diagram (a) visualizes the spatio-temporal sampling behavior of cameras and typical range scanning devices. The observation angle relative to the sensor is represented by the x-axis, the y-axis denotes time. Diagram (b) shows that correspondences between measurements are easy to establish for static scenes and diagram (c) shows that this is not the case for dynamic scenes, since a measured object may have moved in space.

higher resolution of a camera in both space and time while allowing for motion of the sensor platform and/or objects in the scene.

Our framework can easily be adapted to handle depth data from any input source. In this paper, we focus on the case of pairing a scanning laser with a camera. Algorithmically, other sources of range data, such as time-of-flight cameras, are easier to deal with, since they provide time-synchronous arrays of range measurements. Our approach is developed for the general case in which every individual laser beam has its own unique timestamp.

Our method can interpolate depth information with a level of accuracy based on the density of the input information; through use of our framework it is possible to generate depth maps with respect to camera frames that contain sparse depth data, or even no depth data, or were not perfectly aligned with existing depth and camera data. Each interpolated depth value is also automatically assigned a confidence value, based on the availability of spatio-temporally proximate data. We can use this confidence value to threshold the information returned by a query, avoiding artifacts from extrapolation in areas of sparse input data.

Once our framework is constructed, all queries are data-parallel, making implementation in parallel both sensible and advantageous. The work of Adams *et al.* [1, 2] has shown that the speedups offered through GPU implementation of d-dimensional filters allow for high-resolution data processing at interactive rates. Greater control over the quality of data provided by a camera/range system can enable the use of such a system for real-world, mobile vision applications. We evaluate our algorithm in Sec. 5 on both real-world and synthetic data.

2. Background

Our Gaussian framework can be thought of as a d-dimensional extension of the 2D joint bilateral filter, first described by Eisemann and Durand *et al.* [10], Petschnigg *et al.* [16] and Kopf *et al.* [14], and then discussed in a d-dimensional context by Adams *et al.* [2]. Specifically, we alter the data structures of [1, 2], which have only been evaluated in denoising, for use in an interpolation/upsampling domain.

Prior work has shown that high-resolution depth information can be produced through various methods: prior-based estimation with respect to a monocular image, the implementation of stereo vision algorithms using a physical or temporal baseline, or through the pairing of a camera and a non-passive sensor, such as a laser rangefinder or depth camera. In the following paragraphs, we compare our algorithm to related work and discuss the limitations and assumptions implicit in each method.

In traditional stereo camera systems, range error increases quadratically with depth. Techniques exist to bound error [12], but accuracy is still limited in applications with constraints on the temporal or spatial stereo baseline.

The first successful attempt to upsample laser-generated depth values to match the resolution of a camera image was based on Markov Random Fields (MRFs) that used color information from a camera image, and depth information where available [8]. An inherent assumption of the method is that objects in the scene are not moving within the time it takes to complete a scan of the frame with a sweeping laser rangefinder. The terms of the MRF energy function attempt to enforce depth smoothness, but allow for depth discontinuities across color borders. The belief underlying the method is that areas of constant color are most likely areas of constant depth. Therefore, the depth at any given pixel is likely similar to that of its neighbors that are within the same color boundary.

A follow-up paper [3] compares five different interpolation methods with the original MRF method [8]. Again, the underlying assumptions are that proximity and color determine likelihood of a depth value at any pixel location, though their methods also do not incorporate motion or time.

The work of Yang *et al.* [19] compares a non-accelerated, iterative bilateral-filtering method with the MRF approach, showing that a bilateral filtering method allows for sub-pixel accuracy, in contrast with a potentially blocky MRF result. Chan *et al.* [6] extend the bilateral filtering approach to include the noise model of their specific depth sensor, constraining data in noisy regions. Both methods also show that a bilateral filtering-based approach enables a greater increase in spatial resolution than an MRF-based approach. Neither method is widely applicable to all laser types, as their algorithms assume temporally and spatially aligned

depth data at every upsampling reference frame, and do not account for the possibility of a data rate mismatch between the two sensors.

In contrast with other recent methods for high-resolution depth acquisition, our method does not require a strong prior on the geometry of the environment, such as in the work of Furukawa *et al*. [11]. Our only assumption is that motion is piecewise linear at the time scale we are considering; although this assumption does not technically hold in perspective views, it is a decent approximation, especially over short time intervals.

Schuon *et al*. [17] recover dense, denoised depth information without the use of a camera reference image. Their approach requires multiple aligned scans and therefore works only on static scenes.

A distinct feature of our approach and its GPU-accelerated implementation is its time efficiency. Previously, Chan *et al*. [6] upsampled on the order of $25\,000$ input depth points to the resolution of their camera, $48\,000$ pixels, averaging 49 ms per frame. The running time of their algorithm represented up to $5\times$ empirical improvement over other iterative methods [8, 19] (also implemented on the GPU, for fair comparison). Their method focused only on the case of one sensor, however, acquiring depth information from a time-of-flight camera. Our algorithm runs approximately 2.3x faster than their algorithm (running time normalized by number of pixels processed), as detailed in Sec. 4.

3. The Gaussian Framework

In this section we present an overview of our proposed algorithm. We will discuss the general framework for d-dimensional filtering and show how it applies to the creation of high resolution depth maps or color-depth images.

All methods for upsampling range data using camera information, as mentioned in Sec. 2, rely to some degree on the assumption that areas of similar color or appearance in the camera image will have similar depth values. We also rely on this assumption, but, as discussed in Sec. 3.2, our method is general and can use any prior on depth values that can be encoded as Euclidean distance between vectors.

3.1. d-dimensional Filtering

Many image operators such as *blurring*, *bilateral filtering*, *non-local means denoising* [5] or *denoising of image volumes* [4] can be grouped into a general class of d-dimensional filters, formalized as

$$\hat{v}_i = \sum_{j=1}^{n} f(|p_i - p_j|) \cdot v_j \ . \tag{1}$$

Here, each color value v_i in the input is replaced by $\hat{v}_i$, a weighted combination of neighboring color values v_j. Each weight is determined by a function f, considering the difference between position in some d-dimensional space of the point whose value will be replaced, p_i, and the neighbor's position p_j. In the most general case, f can be any kernel. The function most commonly used in the case of denoising is $f(x) = e^{-|x|^2/2\sigma^2}$, a Gaussian with standard deviation σ.

Recently, many algorithms [1, 2, 15] have accelerated filters based on the general formulation above through explicitly representing data in the d-dimensional position space, where d is the dimensionality of the position vectors, and approximating a Gaussian kernel through a three-stage weighted resampling process. Before filtering, a d-dimensional data structure is constructed. Data in the original image manifold is then *sampled* at the resolution of the data structure, and the value at each node in the data structure is set to a weighted sum of nearby input values. The value at each node is then *blurred* with values at neighboring nodes. Finally, the data structure is *queried* at locations along the original manifold in position-space, and an output image is constructed based on the weighted averages of values stored at nodes proximate to the query.

This pipeline greatly accelerates a naive bilateral filter, for example, if one sets each position vector p_i to the (r, g, b, u, v) values at each point; u and v represent a pixel's position in 2D image space. For non-local means, position vectors become (ρ, u, v), where ρ is a vector encoding a description of an image patch around a given pixel.

Adams *et al*. [2], show that a 5D representation is more accurate than other accelerations [7, 18] that treat bilateral filtering as a 3D volume and only consider distance in luminance. Also, in other proposed data structures [1, 2], space is represented only along the 2D data manifold to increase the memory efficiency with respect to approaches that maintain a dense grid [15]. The computational and memory efficiency of such sparse data structures makes blurring with respect to d-dimensional feature vectors tractable.

3.2. Joint d-dimensional Interpolation

Instead of blurring and/or denoising, our goal is interpolation in a dynamic system where both depth information and camera information may be available. In cases where depth data comes from a laser rangefinder, we assume that the extrinsic parameters of both the camera and laser are known (though in practice we have found that these values can be slightly incorrect, and our method still performs well).

Consider the laser and the camera as both sampling a 4D time-space volume, each sensor with a different sampling rate and density. Taking a perspective transform of the scene for every time t results in a 3D (u, v, t) volume, representing a continuous space of camera images that could be returned from the scene, given the perspective of the camera. This 3D image-time volume is sampled at discrete time

intervals dictated by the camera's frame rate. Applying the same perspective transform to the laser returns (after correction for the physical offset of the two sensors), results in the projection of range values into the 3D image-time volume. After this projection, some depth values might be outside the camera image plane. For the full version of our algorithm that uses (ρ, u, v, t) values as position vectors, these range values should be discarded, as each depth value at position (u, v, t) cannot reliably be assigned ρ, an image-based descriptor.

To create a high-resolution depth map, the general d-dimensional filtering formulation given in Eq. 1 remains the same, but v_i and v_j represent depth values instead of color values. We also constrain our position vectors to be of the form (ρ, u, v, t), where ρ represents a descriptor based on color information from the camera, u and v represent the 2D position of the descriptor on the image plane, and t is the time the data was captured. Should camera information not be available, the position vector for each depth point can reduce to (u, v, t). Whereas Adams *et al*. [2] illustrated the utility of adding t to the position vector for temporal coherence in video or image burst denoising, we rely on time measurements to constrain the temporal interpolation that allows us to construct a depth map for an arbitrary camera frame, possibly taken at a time when no depth information was available. If the scene is static, our method still works reliably, though t can be dropped from the position vector, as it is irrelevant (equivalently, the temporal dimension of the Gaussian can be given infinite extent).

3.3. Data Processing and Motion Priors

Selection of useful data points in the data stream and determination of motion priors are important steps in initializing our framework and selecting the correct parameters for Gaussian interpolation. In this section we detail the technique for selecting input data and motion priors in the scanning laser/camera case.

Referring again to Figure 2, we see that for each frame in a scanning laser/camera system only certain points returned by the laser can be considered coincident with pixels in a camera frame. Here we define "coincident" in terms of the difference between a camera frame timestamp and a laser return timestamp, Δt, and the location of the laser return after it is projected into image space. In Sec. 3.2, we discussed discarding points that do not fall on the image plane; now, we also consider whether the amount of object motion in the scene relative to the image plane during Δt could exceed the spatial extent of an image pixel. Coincident depth points can be assigned a descriptor using information from the camera frame, creating a (ρ, u, v, t) point, while non-coincident points cannot, and will be discarded under the full version of our algorithm.

To set the Gaussian standard deviations that constrain interpolation, we must calculate the maximum bound on the distances we expect objects to move between observations. If the field of view of a camera is θ, an object moves relative to the camera with a velocity s, and s_u is the component of velocity parallel to the image plane, the maximum bound on the lateral distance p^*, in pixels, that an object can move during Δt is

$$p^* = (s_u \cdot \Delta t) \cdot \frac{r_u}{(2d)\ \text{atan}(\frac{\theta}{2})}\ , \tag{2}$$

where r_u is the resolution of the sensor in the horizontal dimension, and d is the minimum distance from the camera to the object plane. Setting the standard deviation in the u dimension proportional to p^*, where Δt is the time interval between camera frames, is a principled way to constrain interpolation of depth values. One could similarly solve for p^* in the v dimension, should expected vertical velocity be different from horizontal in a given application.

Given a perfect descriptor ρ, however, the above u and v dimension constraints would not be necessary since related points across time would have exact correspondence in the ρ dimension of the position vector. Choosing the correct descriptor for a specific application and computing the descriptor efficiently is still an area of active research, and beyond the scope of this work. We have found that setting the u and v standard deviations as discussed above and setting $\rho = (r, g, b)$ leads to accurate depth reconstructions for most natural scenes. In our experiments, we selected the standard deviations for our descriptor ρ based on grid search across non-test data sets.

For memory and processor efficiency, the window of time considered by the algorithm should also be constrained. The standard deviation in time and the the size of the input buffer to the algorithm should depend on the data rate of the sensor providing depth information. In Sec. 5 we provide a concrete example of how to set these parameters.

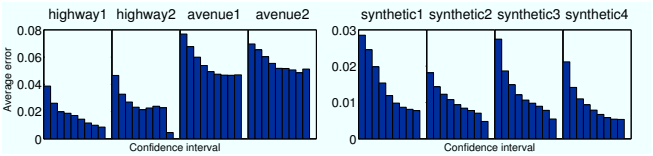

Figure 3. Each plot above shows how the prediction error correlates with the confidence values. For each data set, the reconstructed depth values were binned according to their corresponding confidence values, and the average prediction error was calculated for each bin. As the plots show, higher confidence estimates lead to lower actual prediction errors on average.

3.4. Confidence Weighting

With each query to our d-dimensional data structure, we calculate a weighted average of the values stored at

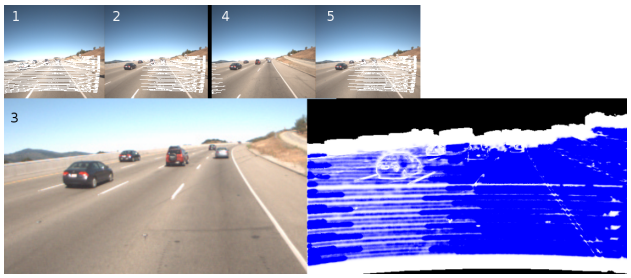

Figure 4. In this visualization, each camera frame is overlaid with laser data that was recorded within 0.02 seconds of the frame capture, i.e. laser returns that can be assigned a color with minimal uncertainty. We hold out frame 3, and wish to reconstruct a depth map using only the data available from the row above. The bottom, right image illustrates the weights returned by our algorithm at each (u, v) location of the (r, g, b, u, v, t) query for every pixel of frame 3. We have zero confidence in areas with no data (black regions), very low confidence in areas with sparse data (white regions), and highest confidence in areas of proximate data (blue regions).

nodes within three standard deviations of our query point, as shown in Equation 1, where f is a Gaussian kernel. The weights returned at each query point are a good indicator of the density and proximity of values relative to the query point. Figure 4 visualizes the confidence value at each pixel, which is equal to the sum of weights returned from a query at that pixel.

We also evaluated the ability of these confidence values to predict interpolation error. Figure 3 shows a graph of confidence values vs. average relative error. We evaluated the error at each frame containing ground truth in each of our data sets (which will be discussed in Sec. 5), using our full algorithm with color information.

4. GPU Implementation

We implemented our d-dimensional Gaussian framework on the GPU using the filtering algorithm of [1]. In this work, space is represented by an $(d + 1)$-dimensional lattice called the permutohedral lattice. The nodes of the data structure correspond to the vertices of this lattice, and the vertices are allocated lazily as the data in the original d-dimensional manifold is sampled. One important feature of this data structure is that the nodes have predictable, deterministic locations, making it fully data-parallel. Consequently, querying points in the d-dimensional space is ideal for the GPU. In comparison, the other state-of-the-art d-dimensional bilateral filter [2] has comparatively more expensive queries, as they require multiple traversals of a KD-tree. As opposed to applying a bilateral filter on an image, in which the number of points used to construct the d-dimensional data structure equals the number of queries afterwards, our task of upsampling is naturally dominated by the querying stage, since our input depth information is sparse compared to our high-resolution output. Therefore, the importance of efficient queries makes the permutohedral lattice an ideal choice.

The GPU algorithm operates on a stream of color images each of which is coupled with a its relevant, sparse depth information. For each pair, the GPU algorithm constructs a d-dimensional data structure using the pair itself and two previous and two subsequent pairs, for a total of five pairs. It then blurs the d-dimensional space and makes a query for each pixel of the color image. For our data sets, this amounts to roughly 27 000 depth points available in our buffer and 1024x768 queries made per frame (the resolution of our image; one query per pixel). Assuming that the data is prepared with zero latency on the CPU, copying the data onto the graphics card device memory[1] and applying the GPU algorithm runs at 29.2 fps.

With respect to the fields of view of the specific camera and laser that we use for our error analysis in Sec. 5, slicing a full 1024x768 frame is inefficient, as we do not return depth information for most of the camera image (e.g. in the sky). Therefore, one could optimize to the specific case of a given camera and sensor by only querying at image points corresponding to the field of view of the sensor. In our case, this optimization increases the frame rate to 43.3 fps.

5. Evaluation

We evaluated our algorithm on eight data sets. Our data sets span the range of noisy and imperfect data with shadows and lighting changes, to perfect ground truth with no lighting changes. Our goal was to select a variety of practically relevant situations where long-range depth is important, including highway and city scenes. Four data sets are synthetically generated, such that we have ground truth depth at every frame. The remaining data sets were recorded from a moving vehicle using a camera and a scanning laser: two are highway scenes and two scenes are recorded in a downtown main-street setting. All data sets are available online (as well as source code) [9]. The non-synthetic data sets were collected using a Velodyne HDL-64E scanning lidar sensor and a Ladybug2 video camera. Note that uncertainties in timestamping and differences in field of view led to slight misalignment of depth and color information in some frames.

We evaluated accuracy only on frames which have full coverage of depth points, such as frames 1, 4, and 7 in Figure 5. To evaluate the errors introduced by each method, we leave out the depth points that are coincident with the frame at which we generate a depth map, so that we can cross-validate the depth estimates using the omitted points. See Figure 5 for a visualization of our input buffer. Our camera and laser frame rates are approximately 30Hz and

[1] We use a desktop graphics card: NVIDIA GTX260 Core 216

Figure 5. Here we show the input buffer used in our non-synthetic evaluations. The white points indicate pixels where both depth and color information is available. We hold out depth information from frame 4, bordered in red, and use only the surrounding depth information to generate a depth map at that frame. The depth information actually recorded at the time frame 4 was captured is then used to evaluate the quality of our generated depth map.

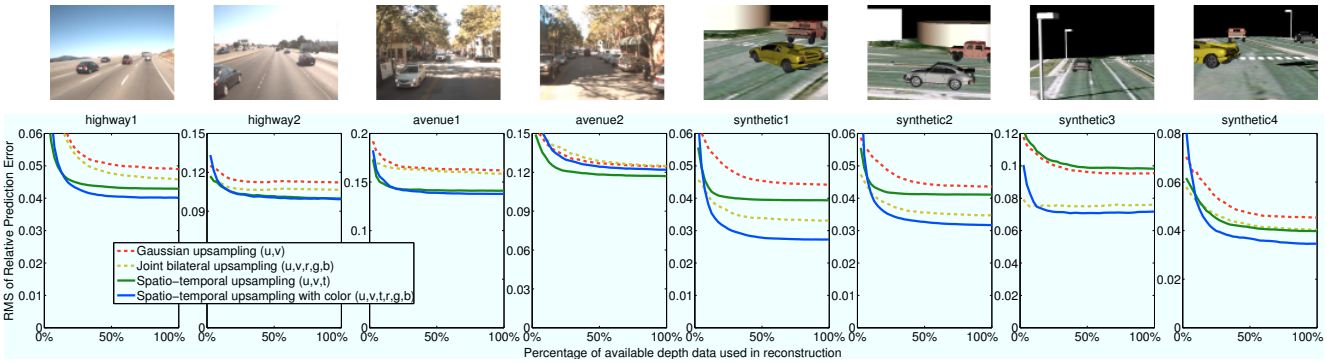

Figure 6. In this figure we show the RMS of relative prediction error for our generated depth maps for four methods: Gaussian upsampling, joint bilateral upsampling, spatio-temporal upsampling, and spatio-temporal upsampling using color information. The x-axis of each plot represents the percentage of available input data used to generate the depth map, as discussed in the text below. In the first two methods, we use only depth data from the previous full-coverage frame, approximating prior work that expects full depth coverage of static scenes.

10Hz respectively, and given the 72 degree field of view of our camera, we expect full coverage[2] of a camera frame by depth values at approximately every 3 camera frames. In practice, this means that we could interpolate using a buffer of 2 camera frames worth of depth points on either side of our query time. However, since we wish to cross-validate our method by leaving out frames with full depth coverage, we expand our buffer to include depth points corresponding to 3 adjacent camera frames in either direction.

Given the sparse ground truth information in the sequences described above, we also wanted to test our algorithm in a fully-controllable setting free of noise. We created our set of four synthetic data sets by rendering dynamic street scenes using OpenGL, in the same format as above, with a color image, depth map, and timestamp at each frame. Unlike the static scenes of the Middlebury data sets [13], commonly used for benchmarking depth map generation, our data sets are sequences in which objects and the sensing platform are moving.

Figure 6 shows the average RMS relative error across different input data densities. We compare 4 different methods of upsampling our sparse depth data. The first method is a naive bilinear interpolation on the nearest full depth information upsampled with respect to the current frame. The second method approximates the various prior work; we perform a joint bilateral upsample using the nearest full depth frame as input and (r, g, b) color values of the current frame to constrain the interpolation (see the discussion in Sec. 2). The third method is our algorithm with position vectors set to (u, v, t), i.e. spatio-temporal interpolation without camera information. The fourth method is the full algorithm, with position vectors (r, g, b, u, v, t). The standard deviations of the Gaussians were fixed across all methods.

To produce the curves in Figure 6, we selected four evaluation frames from each test sequence which coincided with full range coverage (e.g. Frame 4 visualized in Figure 5) and removed those from the sequence. The task was then to match the held-out range values given the surrounding frames and range measurements.

We show the convergence behavior of the different algorithms with respect to available range measurement densities by additionally subsampling the range measurements. Range points were randomly sampled from a uniform distribution. The horizontal axes of the diagrams in Figure 6

[2]by "full," we mean depth values are distributed across the horizontal extent of the frame, though the resolution of the HDL-64E sensor is much less than the camera as shown in Figure 1

Figure 7. This figure compares depth maps generated by our algorithm with color information and without. We zoom in on the area of the depth map near the closest car; the first two columns show the depth map in that area, and the second two are 3D visualizations where each image pixel is offset by its corresponding value in the depth map. Without color information, blending occurs across depth discontinuities.

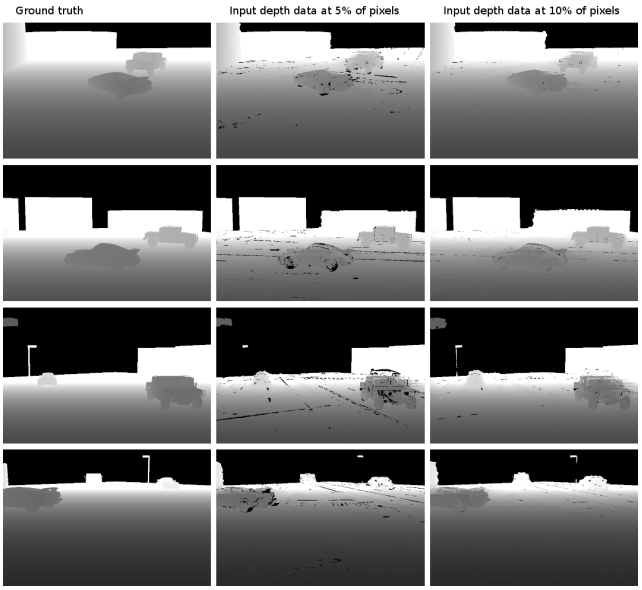

Figure 8. In this figure, we compare the depth maps generated by our algorithm with the dense ground truth depth for the synthetic data sets. In our evaluation, we leave out all depth information at the current frame and use only a small fraction of depth data from neighboring frames. The second and third columns show the result of using 5% and 10%, respectively, of the depth data from two preceding and two subsequent frames. Note that some holes appear in the second column depth maps due to the inadequate density of depth data in some places, given our selected Gaussian standard deviations.

Figure 9. Color is not a good descriptor when shadows and specular highlights cause depth information to blend into regions that do not belong to the object.

give the amount of range measurements visible to the algorithms. Note that even a value of 1, that is, the case in which all range measurements are available, means that only 5% of all image pixels were within one pixel of a range measurement. The experiments were run 10 times for each sequence and the average relative RMS prediction errors are plotted in Figure 6. The standard deviations of the errors over the 10 trials were between 1% and 5% of each average, assuring that the average error shown is indicative of the performance of a typical execution. Error bars were omitted for better readability of the graphs. Also, in scenes with very high relative RMS (L-2 norm) error, the error measurement is significantly affected by outliers, belying the quality of the resulting depth map; the average relative error (L-1 norm) is 2-4%.

Figures 7 and 9 analyze the use of color information from the camera frame in constraining interpolation. An area of constant color is usually a good indication of an area of constant depth; however, that assumption is sometimes violated, as shown in Figure 9. Bright lighting and dark shadows in our avenue 1 and 2 sequences most likely account for the decrease in performance of the algorithm when using color information. Another factor is alignment of the camera and laser, in terms of field of view and extrinsic calibration. The avenue datasets illustrate the effect of slight misalignment, lighting, and appearance changes. In cases of gross misalignment (due to inaccuracies in time-stamping, vibration, or other factors) and differences in occlusion boundaries, depth points are assigned an erroneous color value, possibly leading to artifacts and errors in interpolated depth. With worse alignment, parallax due to perspective differences between the sensors, and more variation in lighting, performance of all evaluated algorithms declines.

When the scene does not contain much motion relative to the total amount of pixels in an image, static joint bilateral upsampling outperforms temporal interpolation without color, as it better preserves object boundaries. Preserving object boundaries is especially important in terms of error evaluation in our synthetic data sets where we have full

ground truth depth along all object edges. In our data sets, whenever the scene is one in which color is a useful indication of depth boundaries, and the scene contains motion, using the full (r, g, b, u, v, t) descriptor always produces a more accurate depth map than static bilateral upsampling, given depth data at greater than 2.5% of camera pixels.

The quality of the reconstructed depth map is dictated by the density of available depth data, and depending on the standard deviations of the d-dimensional blur, regions of the image in which there are no depth data available will result in holes, as shown in Figure 8. The holes can be removed either by having more depth data available to our algorithm, by increasing the Gaussian standard deviations, or by using multi-scale techniques. Using large standard deviations to fill areas, however, constitutes a tradeoff of accuracy for continuity of interpolated information.

6. Conclusions and Future Work

Sensor systems typically differ significantly in their spatial and/or temporal resolution. Previous upsampling algorithms were limited to static scenes, and possibly limited in scope to a single sensor. Our framework is flexible and can generate depth maps at increased spatial and temporal resolution for both static and dynamic scenes.

Use of our method could enable the generation of depth maps at the spatial and temporal resolution of a camera, even if given camera frames have no coincident data. Depth maps at every camera frame provide good input to many tracking and classification algorithms. One could perform unsupervised segmentation of video sequences given sparse depth information, for example.

Through the study of depth upsampling, we also hope to have illustrated the applicability of a d-dimensional denoising framework to general d-dimensional joint filtering tasks. Because our framework is general, one could also interpolate multiple values concurrently; for example, lidar intensity returns could be added and upsampled in addition to depth values. In the future we would like to explore use of this data structure to automatically assign labels and confidence values derived from depths values, or possibly hand-labeled semantic data, for a semi-supervised way of labeling image data.

Acknowledgements

This work was supported by an NDSEG Graduate Fellowship from the United States Department of Defense. Thanks to Andrew Adams for helpful discussions, and Mike Sokolsky for assistance with data collection.

References

[1] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum (EG 2010 Proceedings)*, 29(2), 2010.

[2] A. Adams, N. Gelfand, J. Dolson, and M. Levoy. Gaussian kd-trees for fast high-dimensional filtering. *ACM Transactions on Graphics (Proc. SIGGRAPH 2009)*, 28(3):1–12, 2009.

[3] H. Andreasson, R. Triebel, and A. J. Lilienthal. *Non-iterative Vision-based Interpolation of 3D Laser Scans*, volume 76 of *Studies in Computational Intelligence*, pages 83–90. Springer, Germany, Aug 14 2007.

[4] E. P. Bennett and L. McMillan. Video enhancement using per-pixel virtual exposures. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005)*, 2005.

[5] A. Buades, B. Coll, and J.-M. Morel. Nonlocal image and movie denoising. *International Journal of Computer Vision*, 76(2):123–139, 2008.

[6] D. Chan, H. Buisman, C. Theobalt, and S. Thrun. A noise-aware filter for real-time depth upsampling. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, 2008.

[7] J. Chen, S. Paris, and F. Durand. Real-time edge-aware image processing with the bilateral grid. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*. ACM, 2007.

[8] J. Diebel and S. Thrun. An application of markov random fields to range sensing. In *Proceedings of Conference on Neural Information Processing Systems (NIPS)*, Cambridge, MA, 2005. MIT Press.

[9] J. Dolson, J. Baek, C. Plagemann, and S. Thrun. Data sets and source code available on the project website http://graphics.stanford.edu/papers/upsampling_cvpr10.

[10] E. Eisemann and F. Durand. Flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics*, 23(3):673–678, 2004.

[11] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. Manhattan-world stereo. In *CVPR*, 2009.

[12] D. Gallup, J.-M. Frahm, P. Mordohai, and M. Pollefeys. Variable baseline/resolution stereo. In *CVPR*, 2008.

[13] H. Hirschmüller and D. Scharstein. Evaluation of cost functions for stereo matching. In *CVPR*, 2007.

[14] J. Kopf, M. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2007)*, 26(3), 2007.

[15] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. *International Journal of Computer Vision*, 81:24–52, 2009.

[16] G. Petschnigg, R. Szeliski, M. Agrawala, M. F. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics*, 23(3):664–672, 2004.

[17] S. Schuon, C. Theobalt, J. Davis, and S. Thrun. High-quality scanning using time-of-flight depth superresolution. In *Time of Flight Camera based Computer Vision*, 2008.

[18] Q. Yang, K.-H. Tan, and N. Ahuja. Real-time o(1) bilateral filtering. In *CVPR*, 2009.

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