From Point Clouds to Mesh using Regression

L'ubor Ladický
ETH Zürich
lubor.ladicky@inf.ethz.ch

Olivier Saurer
ETH Zürich
saurero@inf.ethz.ch

SoHyeon Jeong
ETH Zürich
sohyeon.jeong@inf.ethz.ch

Fabio Maninchedda
ETH Zürich
fabiom@inf.ethz.ch

Marc Pollefeys
ETH Zürich, Microsoft
marc.pollefeys@inf.ethz.ch

Abstract

Surface reconstruction from a point cloud is a standard subproblem in many algorithms for dense 3D reconstruction from RGB images or depth maps. Methods, performing only local operations in the vicinity of individual points, are very fast, but reconstructed models typically contain lots of holes. On the other hand, regularized volumetric approaches, formulated as a global optimization, are typically too slow for real-time interactive applications.

We propose to use a regression forest based method, which predicts the projection of a grid point to the surface, depending on the spatial configuration of point density in the grid point neighborhood. We designed a suitable feature vector and efficient oct-tree based GPU evaluation, capable of predicting surface of high resolution 3D models in milliseconds. Our method learns and predicts surfaces from an observed point cloud sparser than the evaluation grid, and therefore effectively acts as a regularizer.

1. Introduction

There is an increasing demand in creating compelling 3D models from 2D image sets. Current reconstruction pipelines can build 3D models from millions of images at city scale [1]. Others have optimized the reconstruction process for real-time applications and showed that the full pipeline can run on commodity smart phones [27, 14, 19, 25]. While the reconstruction process has been highly optimized one of the challenges which remains is the efficient extraction of a triangular mesh from a point cloud. While standard dense reconstruction approaches provide noisy depth measurements the approach we are seeking for needs to be robust towards noisy measurements, outliers and needs to be able to cope with varying point cloud densities.

Surface reconstruction algorithms can mainly be divided into two categories, which are volumetric approaches and surface based methods. Various algorithms have been proposed for volumetric integration of depth measurements, we refer here to the most closely related publications. Curless *et al.* [7] compute the signed distance for each point and extract an isosurface at the zero crossing using marching cubes. The method expects the input depth maps of similar scale and density, and lacks global regularization and only averages depth measurements locally, leading to holes at places where the point cloud is sparse. Real-time variants of this approach have been used in [19, 17, 11]. To extend the surface into neighboring areas with no measurement, local surface reconstruction methods such as variants of moving least squares [5] have been proposed, however they often struggle with small density of points, especially with extrapolation into holes. To overcome this problem, Fuhrmann *et al.* [10] propose a fusion algorithm that copes with depth maps of varying resolutions, allowing to simultaneously represent low resolution regions as well as high resolution details. They use an oct-tree representation to store the signed distance field at various scales. Similarly Ummenhofer *et al.* [30] proposed to globally optimize the signed distance field on a hierarchical oct-tree. While those algorithms provide very compelling results, they operate in terms of hours and are therefor not suitable for real-time applications. Zach *et al.* [34] formulates the depth integration task as a global optimization problem. They use a L_1 data fidelity term to be more robust towards noise and outliers. While the method provides impressive results it is restricted by its computation time and memory consumption. In [33] Zach proposed a robust and memory efficient approach to integrate range images. Using recent advances in optimization Savinov *et al.* [23] formulate the visibility constraints as high order potentials over rays and solves it using non-convex relaxation methods. This method is currently the state-of-the-art on the Middlebury dataset [24], however, it takes days to run on high resolution models.

Surface-based methods parametrize the problem in terms of basis functions and formulate the optimization problem, which either fits the point cloud data [4], or minimizes the difference between the gradient of indicator function of surface interior and input normals of the point in the point cloud [12]. Recently, various forms of basis functions and additional regularizations have been proposed [20, 8].

We propose to use a regression forest [3] based method, which predicts the projection of a grid point to the surface, depending on the spatial configuration of point density in the grid point neighborhood. We designed a suitable feature vector and efficient oct-tree based GPU evaluation, capable of predicting surface of higher resolution 3D models in milliseconds. Our method learns and predicts surfaces from an observed point cloud sparser than the evaluation grid and therefore effectively acts as a regularizer. To our knowledge, our method is the first learning based surface reconstruction, with the exception on the not directly related work of Firman *et al.* [9], that formulates the prediction of unobserved (occluded) structures as a regression task.

2. Method description

Our algorithm predicts the projection of a given grid point to a surface, which is eventually used to calculate the signed distance of this grid point. The final surface is estimated as a 0-isosurface of the predicted signed distance field (SDF) using the marching cubes algorithm.

The projection $\pi_S(\mathbf{x})$ of a point $\mathbf{x}$ to the surface S is defined as:

$$\pi_S(\mathbf{x}) = arg\min_{\mathbf{x}' \in S} |\mathbf{x}' - \mathbf{x}|. \tag{1}$$

The projection difference $\Delta_S(\mathbf{x})$ is defined as a vector between the point and its projection:

$$\Delta_S(\mathbf{x}) = \mathbf{x} - \pi_S(\mathbf{x}). \tag{2}$$

The unsigned distance of a point $\mathbf{x}$ to the surface S can be calculated as:

$$d_S(\mathbf{x}) = |\Delta_S(\mathbf{x})|. \tag{3}$$

Let $\mathbf{n}_S(\mathbf{x})$ be the normal of a surface S in a point $\mathbf{x}$. The sign in the signed distance $s_S(\mathbf{x})$ can be determined based on the agreement of the projection difference $\Delta_S(\mathbf{x})$ with the normal in the point projected to the surface $\mathbf{n}_S(\pi_S(\mathbf{x}))$:

$$s_S(\mathbf{x}) = d_S(\mathbf{x}) sgn(\Delta_S(\mathbf{x}) \cdot \mathbf{n}_S(\pi_S(\mathbf{x}))), \tag{4}$$

where the signed distance is positive outside and negative inside of the surface S. Thus, we have established, that given the projection difference of a point and the normal field we can determine the signed distance. Note, that we do not need exact normals to determine the sign of the signed distance; it is sufficient to use any approximation, which does not differ from correct normal by more than 90 degrees, for example a direction of a viewing ray. The translation invariance of the projection difference (the projection difference stays the same, if we shift both a surface and a point) makes it suitable to be used as a regressed vector in learning. We discuss other alternatives in Section 2.5. The schematic description of our algorithm is depicted in Figure 1.

2.1. Learning method

Our goal is to choose the learning method and design a feature vector, which can train the following:

$$\Delta_S(\mathbf{x}) := f(\Phi(\mathbf{x})), \tag{5}$$

where $\Phi(x)$ is a yet to be chosen feature vector for a point $\mathbf{x}$ and $f(.)$ is the regression function predicting a projection difference. The advantage of using projection difference as the regressed vector is, that it is independent on orientation, and thus it can be reliably predicted using only positions of points without using normals. We are interested in the prediction over a discrete grid, thus we approximate the set of points in a point cloud by the density field $\rho(\mathbf{x})$ using trilinear interpolation of a constant 1 for each particle, and formulate the regression problem as:

$$\Delta_S(\mathbf{x}) := f(\rho(\mathbf{x} + \mathbf{x}')), \tag{6}$$

where $\mathbf{x} + \mathbf{x}' \in N(\mathbf{x})$ and $N(\mathbf{x})$ is the neighborhood of $\mathbf{x}$ of the size depended on the range of predicted projection difference. Recent advances in machine learning suggest, that the best method for learning a quantity based on spatial context of features $\rho(\mathbf{x} + \mathbf{x}')$ are convolutional neural networks (CNNs) [6]. However, due to their large computational requirements to train large 3D models [21] and the slow evaluation, we decided to use regression forests [3], which were already successfully applied to similar context-based learning problems [28, 15].

2.2. Feature vector

The main weakness of tree-based methods is that we have to hand-design a robust feature vector, such that decisions (typically based on individual dimensions of the feature vector) can rapidly decrease the variance of output quantity. A naïve density feature $\rho(\mathbf{x} + \mathbf{x}')$ in single point in the neighborhood does not have this property due to the sparseness of the points in a point cloud. Instead of a single point densities, we use more robust total densities $\Omega(\mathbf{x}+\mathbf{B})$ over boxes $B \in \mathcal{B}$ placed relative to $\mathbf{x}$:

$$\Omega(\mathbf{x} + \mathbf{B}) = \sum_{x' \in \mathbf{B}} \rho(\mathbf{x} + \mathbf{x}'). \tag{7}$$

The final feature vector $\Phi(\mathbf{x})$ is a concatenation of these individual features over a large set of randomly generated boxes $\mathcal{B}$. Intuitively, these features can rapidly decrease the variance of projection difference with a single decision - for example, all training samples with a zero total density in a

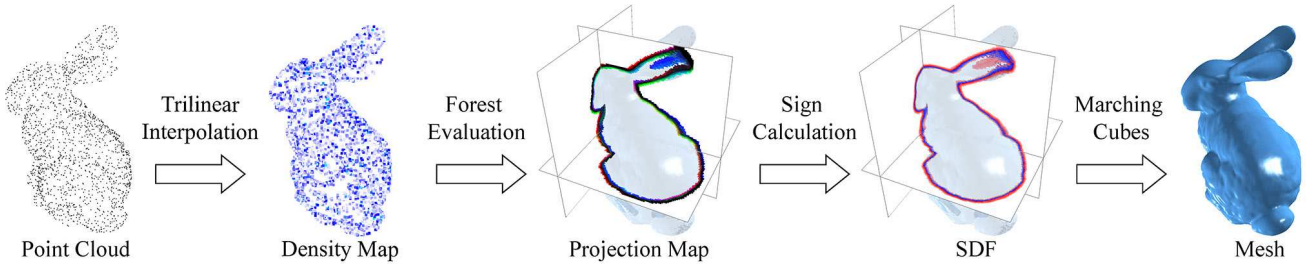

Figure 1. Schematic description of our algorithm. Given a point cloud, we generate a density map using trilinear interpolation. Using this density map, we predict the projection to the surface (RGB corresponds to XYZ coordinates of the projection difference), and calculate the signed distance field by estimating the sign from the scalar product of each projection vector with a corresponding normal (red is outside, blue is inside, the darker the closer to the surface). Finally, the marching cubes algorithm is applied to obtain the final mesh.

large box to the left from an evaluate sample, have a positive x-coordinate of a projection difference. Furthermore, this feature vector is robust to noise and can be trained to ignore uncorrelated outliers. Individual feature dimensions can be evaluated using integral volumes in a similar fashion to [31, 26, 16].

2.3. Regression forest training

Given a set of training feature vectors $\Phi(\mathbf{x}^i)$ and their ground truth projection differences $\Delta(\mathbf{x}^i)$, we aim to find a function $f(.)$, minimizing the expected squared loss function:

$$L = \sum_{i \in D} (f(\Phi(\mathbf{x}^i)) - \Delta(\mathbf{x}^i))^2, \tag{8}$$

where D is the set of training samples. The cost function is optimized greedily [3] for each node N over set of data points D^N, that ended up in this node based on previous decisions in a tree. In each step the most discriminative decision stump $\Phi(\mathbf{x}^i)_j \geq \theta_j$ for each node N is found by brute-force search for j and θ_j by minimizing:

$$L^N = \sum_{i \in D_R^N} (\mu(D_R^N) - \Delta(\mathbf{x}^i))^2 + \sum_{i \in D_L^N} (\mu(D_L^N) - \Delta(\mathbf{x}^i))^2, \tag{9}$$

where D_R^N is the set of samples satisfying $\Phi(\mathbf{x}^i)_j \geq \theta_j$, $D_R^N = D^N \setminus D_L^N$ is its complement, and $\mu(D')$ is the mean of a set D'. This procedure is iteratively applied either till the depth of a tree reaches predefined limit, or the number of training sampled in a given node is below a chosen threshold. The final prediction of the projection difference for a given leaf node N is the mean $\mu(D^N)$ over data D^N, that ended up in this node. Final regression forest predictions are calculated as an average over multiple trees, trained for different subsets of data or using randomly sampled subset of dimensions for each decision.

2.4. Adaptive evaluation of the regressor

An evaluation starts with trilinear interpolation of points and normal into a density grid and a normal grid, required to correctly determine the sign of the signed distance. To avoid the possibility of no particle in a projected point to the surface, the normal grid is calculated in lower resolution ($4 \times 4 \times 4$ sub-sampling) than the point density. The approximation does not cause any problems in practice (except very thin surfaces with different densities on both sides), because the sign calculation does not require exact normals. The next step is the calculation of an integral volume for density, which can be done in linear time with respect to the number of grid cells.

In practice, we avoid any unnecessary calculation far from the surface. First, we split the volume into boxes of the size $8 \times 8 \times 8$ cells. We extend each box by K cells every direction and check that there is no point in this box, we have a guarantee, that the surface is at least K cells far. This test can be done with a single box evaluation using preprocessed integral images and for $K = 8$ we are typically able to avoid 90% of forest evaluations. For the remaining occupied boxes, we iteratively evaluate the forest (tree) over an oct-tree for $4 \times 4 \times 4$, $2 \times 2 \times 2$ and $1 \times 1 \times 1$ scales. We determine, whether it is necessary to evaluate the next subdivision of an $n \times n \times n$ cell, based on the values of the maximum unsigned distance $d_{max} = \max_{\mathbf{x}_k \in C} |\Delta_S(\mathbf{x}_k)|$ and the minimum unsigned distance $d_{min} = \min_{\mathbf{x}_k \in C} |\Delta_S(\mathbf{x}_k)|$ of corners of a cell, where C is the set of corners of a cell. Using triangular inequality we show, that there can not be a point inside the cell that belongs to the surface, if $d_{max} + d_{min} > \sqrt{3} n |c|$, where $|c|$ is the cell size. We relax this condition (to accommodate the error of the regressed projection difference) by adding a slack ϵ. If the condition is satisfied, we skip further evaluation on a lower level in the pyramid. In practice, GPU implementation gets faster by skipping the $2 \times 2 \times 2$ step. For the remaining cells, we evaluate the signed distance from the projection difference using equation (4). We avoid extrapolation of the surface too far from the data point, we flip the sign of the distance of a cell only if the unsigned distance is less than a threshold (we used an 8 cells threshold in experiments). Eventually, we smooth the obtained signed distance field with a gaussian filter with $\sigma = 0.5|c|$ and evaluate marching cubes to get the final surface mesh.

2.5. Alternative formulations

Regressing the projection difference was not the only alternative. We considered two other options - the direct regression of absolute distance and the regression of signed distance from the reconstructed surface.

The absolute distance can be successfully regressed using regression forests, because proposed decision stumps do provide strong cues about the distance leading to a rapid decrease of variance during training. Intuitively, no points in a large box $[-K, K]^3$ give a strong cue, that the surface is at least K cells. Otherwise, a sufficient number of points above the noise level in such a box suggest, the surface is closer than $\sqrt{3}K$. Combining such cues one could quickly determine the correct absolute distance (similarly to projection) after a few decision. However, the use of absolute distance suffers from two main weaknesses. First, the isosurface has to be extracted at a non-zero ϵ value. Theoretically, this can be fixed by shifting all points by ϵ in the negative normal direction, but this does not work well in practice with only approximate estimates of normals. Second, the extraction of surface from an unsigned distance would generate two surfaces - one outside and one inside.

Another option we considered was a direct prediction of signed distance, which can not be predicted without the inclusion of normals in the feature vector. Construction of separate normal-based feature vector would lead to a significant slowdown of a method. The approaches directly incorporating normals in density (such as a point $\mathbf{x}$ placed with a positive weight to $\mathbf{x} + \mathbf{n}\delta$ and with a negative weight to $\mathbf{x} - \mathbf{n}\delta$) turned out to lose the ability to predict sub-cell signed distances. Intuitively, it is very hard to think of a simple decision stump, that can rapidly decrease the variance, by giving a strong cue to distinguish between the signed distance higher and lower than given threshold. Despite several other attempts (such as regressing both the signed and the unsigned distance together), we failed to reproduce the quality of the results, we were able to achieve by regressing the projection difference or absolute distance.

3. Data generation

While there are many ways to capture high quality scans using LiDAR, Kinect etc. it turned out to be sufficient to use synthetic models to train the proposed regressor. Since the rendered point clouds do not exactly represent the noise distribution and the point density of the real data, which is hard to replicate due to different noise sources and varying textureness of the scene. Typically stereo algorithms provide high point densities in texture rich areas while no reconstruction is possible in textureless regions. However, as we show, the quality of the synthetic data is sufficient to train a regressor, that can generalize to the real-world test data, captured using a mobile phone.

3.1. Synthetic data generation

To train the regression tree we make use of the *ModelNet40* [32] dataset, where the following object classes are used: *Flower Pot, Lamp, Plant, Sink, Toilet* and *Vase*. Models, which have inconsistent face normal orientations, intersecting meshes, only a single side of a thin object (leaf), degenerate triangles, multiple overlapping triangles lying on each other etc, were pruned from the dataset. The pruning was not done due to the difficulty of the data. The final dataset consists of a total of 405 models. The examples of 3D meshes from the training set are depicted in Figure 2. We generated sparse point clouds from the meshed 3D models by uniformly sampling points over the surface of each model. Given three vertices $\{v_1, v_2, v_3\} \in \mathbb{R}^3$ of a triangle, a point $p \in \mathbb{R}^3$ is uniformly sampled using the convex combination of the vertices:

$$p = (1 - \sqrt{r_1})v_1 + (1 - r_2)\sqrt{r_1}v_2 + \sqrt{r_1}r_2v_1, \quad (10)$$

where $r_1, r_2 \sim U[0, 1]$. On average we sample 40k points per model. Additional Gaussian noise is added to the point set with a σ of half a grid cell. Eventually, we added 2% of uniformly distributed outliers.

3.2. Real-world data capture

For real data acquisition, we use a similar methodology as was proposed by Tanskanen *et al.* [27] to capture 3D models with a mobile phone. For sake of completeness we briefly outline the mobile 3D reconstruction pipeline here. It consists of two main building blocks, which are the *camera pose estimation* and the *dense reconstruction* parts.

3.2.1 Camera pose estimation

The input to our 3D reconstruction pipeline is a continuous image stream and the intrinsic parameters of the camera. At first the 3D reconstruction process is initialized by tracking 2D feature points between consecutive images and estimating the camera pose between the current frame and the first camera frame using the 5-point algorithm [18] embedded in a RANSAC [2] framework. Once the baseline between the two frames is large enough the scene map is initialized by triangulating 3D points. The map is used to estimate the camera pose relative to the scene map by projecting the map points into the current frame and optimizing a photo metric error [13, 27]. To be more robust towards large camera motions, the optimization is performed over multiple image scales. While the image registration process provides fast frame by frame mapping a more accurate camera pose is required for dense reconstruction. We perform a sliding window bundle adjustment [29] over a subset of images, which we refer to as keyframes. After each newly selected keyframe we extend the scene map by extracting new keypoints using fast corner [22] detection

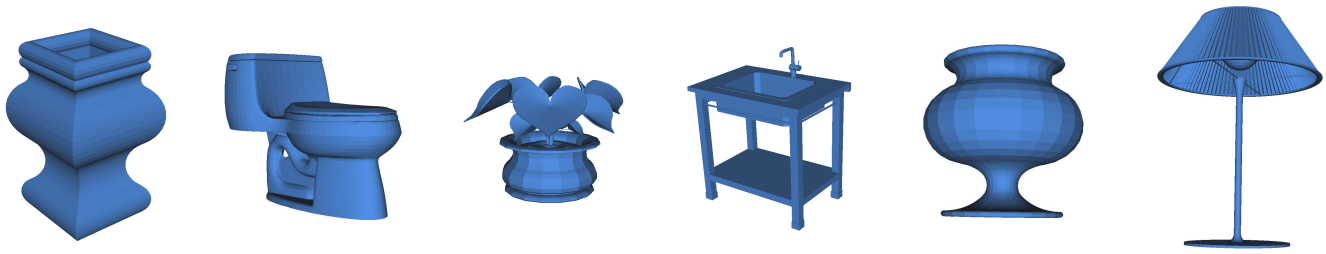

Figure 2. Examples of synthetic 3D models from ModelNet40 dataset. All models are of the following object classes: *Flower Pot, Lamp, Plant, Sink, Toilet* or *Vase*.

and searching for keypoint correspondences in neighboring views. Correspondences search is done by matching an affine transformed 8×8 pixel patch across views. The optimized keyframes are then passed on to the dense reconstruction part of our pipeline. To achieve real-time performance, the computations are performed at an image resolution of 640×480 pixels.

3.2.2 Dense reconstruction

For each pair of keyframes we estimate the depth by performing a pixel wise correspondence search along the epipolar line. To be robust towards noise, outliers and small specularities, the search is done over multiple resolutions, starting at the lowest resolution and refining the correspondence match by moving down the image pyramid. In our experiments we use the Sum of Squared Differences (SSD) as correlation cost with a kernel size of 5×5 pixels and an image scale space of 3 levels. The result is a depthmap which is typically corrupted by noise due to image sensor noise, motion blur, rolling shutter effect or lack of texture. We apply an efficient filtering scheme to remove noisy measurements by enforcing depth consistency across neighboring views. If a depth measurement obtains sufficient support from surrounding views the measurement is retained otherwise it is marked as invalid. The pre-filtered depthmaps are then smoothed using an edge preserving bilateral filter (kernel size 5×5), to improve the depth measurements. Finally different depth estimates are put into the agreement or discarded using a confidence based fusion approach, similar to Kolev *et al.* [14]. Here each depth sample is assigned a confidence weight. If a new depth measurement is in accordance with the existing depth (e.g., within a given range) the confidence is updated. Only depth samples which exceed a certain confidence threshold are kept and used in the proposed regression approach.

4. Experiments

We trained a regressor on the synthetic data, generated as explained in section 3.1. We used 102 synthetic 3D models discretized to the resolution $400 \times 400 \times 400$ for training. Each model was randomly scaled by a value in the range $[\frac{2}{3}, \frac{3}{2}]$ to get a higher variance of densities in the training set. Remaining 303 synthetic models were used as a test set. As training samples we used only grid points with smaller distance to the surface than 10 cells. Points with projection distances to two different surfaces within a small margin of difference of distances below 1 cell and with the angle of projection differing by more than 45 degrees were ignored during training. In total, 20 million samples were used for training. We trained a forest consisting of 4 trees of maximum depth 20, however, the performance was comparable using only one tree and all reported results are obtained under this setting. The feature vector consisted of total densities in randomly generated 5000 boxes, surrounding each grid point. During test time only boxes required to reach the leaf node are evaluated (at most 20 comparisons).

The qualitative results on the synthetic test set are shown in Figure 3. Our method managed to obtain surface reconstructions comparable to the ground truth. Slight errors occurred for very thin objects (such as leaves of plants), where the noise made the 3D reconstruction thicker, or for concave parts with high curvature, which are rare in this data set, and thus hard to predict during test time. The quantitative results for varying level of spherical gaussian noise from $\sigma = 0$ to $\sigma = 2|c|$, where $|c|$ is the size of the cell, for varying ranges $(0 - 2|c|, .., 8 - 10|c|)$ of ground truth distances are reported in table 2. The mean error of projection near the surface was $0.44|c|$, which can be considered reasonable for a given density of input points. The inclusion of 2% outliers in the artificial test set had (almost) no effect on the performance. It was very unlikely they were near the surface (statistically 1-2 points per model), or that they formed a structure, that was reconstructed. The robustness to very large noise can be only achieved by re-training for each particular setting. This would have a different set of applications and is beyond the scope of this paper. The method already works well ($0.66|c|$ error with no noise), when we trained the regressor on only 4 handpicked models, consisting of one flat object (cup), one object with lots of details (plant), one object with both convex and concave parts (vase) and one object with varying level of details (toilet). Note that our real data test set is of a completely different kind, thus we do not consider training data to be a

	boy	mask	statue	elephant	face
points	306,018	299,403	246,941	244,167	91,730
resolution	$201 \times 646 \times 172$	$228 \times 326 \times 89$	$403 \times 526 \times 349$	$396 \times 160 \times 455$	$215 \times 289 \times 182$
triangles	1,185,092	936,260	1,361,032	1,542,828	720,688
density	2.8ms	2.7ms	2.7ms	2.6ms	0.8ms
normals	1.1ms	1.2ms	3.5ms	1.0ms	0.3ms
integral volume	8.0ms	1.6ms	25.5ms	8.1ms	4.4ms
tree evaluation	13.0ms	4.9ms	26.0ms	17.2ms	10.3ms
smoothing	2.6ms	1.7ms	7.9ms	2.9ms	1.8ms
total	27.5ms	12.1ms	65.6ms	31.8ms	17.6ms
marching cubes	19.8ms	13.5ms	36.9ms	22.5ms	16.2ms
total with MC	47.3ms	25.6ms	102.5ms	54.3ms	33.8ms

Table 1. *Comparison of running times of the GPU implementation of our method on GTX1080 for the* 5 *models from Figure 5. The evaluation of our regressor and calculation of signed distance field takes comparable time to the marching cubes algorithm, which is part of most alternative approaches, including the fastest TSDF method.*

	mean	0-2$\lvert c\rvert$	2-4$\lvert c\rvert$	4-6$\lvert c\rvert$	6-8$\lvert c\rvert$	8-10$\lvert c\rvert$
no noise	0.754	0.440	0.686	0.816	0.941	1.128
$\sigma = 0.5\lvert c\rvert$	0.851	0.455	0.783	0.961	1.109	1.221
$\sigma = 1.0\lvert c\rvert$	1.129	0.578	0.986	1.281	1.541	1.658
$\sigma = 1.5\lvert c\rvert$	1.511	0.684	1.335	1.783	2.104	2.210
$\sigma = 2.0\lvert c\rvert$	1.910	0.803	1.647	2.284	2.742	2.839
ratio	100%	25.8%	23.2%	19.8%	16.8%	14.4%

Table 2. *The average error of the projection in the cell units* $\lvert c\rvert$ *for varying range of ground truth absolute distances (columns) and varying level of spherical gaussian noise (rows). Note that a noise in the tangent direction has no influence on the error. Ratio corresponds to the percentage of samples in each range. Higher ratio of smaller distances is due to a large number of thin flat objects in the dataset, such as cups, pots, or vases. In that case, smaller distance samples are from both sides of both outer and inner boundaries, higher distance samples are only from outside of each boundary.*

serious issue.

We captured the real-world 3D models using a mobile phone as explained in Section 3.2. The comparison of our results with the alternative methods - TSDF, Poisson reconstruction and TV-L1 can be found in Figure 5. Our method achieved comparable results to the state-of-the-art. TSDF models the signed distance field only locally, which results in a large number of holes in the resulting 3D model. Poisson reconstruction is not very robust to outliers and the extrapolated surfaces fitting the noise are often not visually pleasant. TV-L1 regularization tries to find a minimal surface, that fits all points. This often leads to filling the holes, that should not be filled. In general this method gives the best results, however, for high resolution models it takes several hours to compute (CPU implementation). Our method often led to smoother results due to averaging in the regression forest training. This property is desired for lower quality point clouds. Additional results for more challenging objects with high noise and non-Lambertian specular surfaces are shown in Figure 4.

The running time largely depends on the resolution of the space, the number of points, ratio of occupied space and amount of noise. The running times on GTX1080 of the examples from the Figure 5 with all their parameters are shown in Table 1. For a model of the size $200 \times 200 \times 200$ it takes approximately 10 milliseconds to predict the projection difference and signed distance field. That is comparable to the evaluation of marching cubes (approx 10 milliseconds), which is part of most other competing approaches including the fastest TSDF [7] (approx 2 + 10 ms). In comparison, Poisson reconstruction for such model takes approximately 5 seconds and TV-L1 30 seconds on GPU. We implemented our method on the mobile phone, where it can generate high resolution 3D models in interactive time.

5. Conclusion

We proposed a new surface reconstruction method using regression forests, capable to reconstructing high resolution meshes in milliseconds with the quality comparable to the state-of-the-art. Because the predictions depend only on the local context, the method can be used to the arbitrary large point cloud, by splitting the space into a set of regions, overlapping by as many cells as the range of the regressor. In the future work we would like to find a better way of generating training point clouds, more similar to real-world noisy data, which seems to be the main problem in our pipeline. A promising direction is to use high quality 3D reconstructions obtained by fusing data from multiple depth sensors, or to generate textured synthetic images with more realistic rendering.

Acknowledgements L'ubor Ladický is funded by the Max Planck Center for Learning Systems Fellowship.

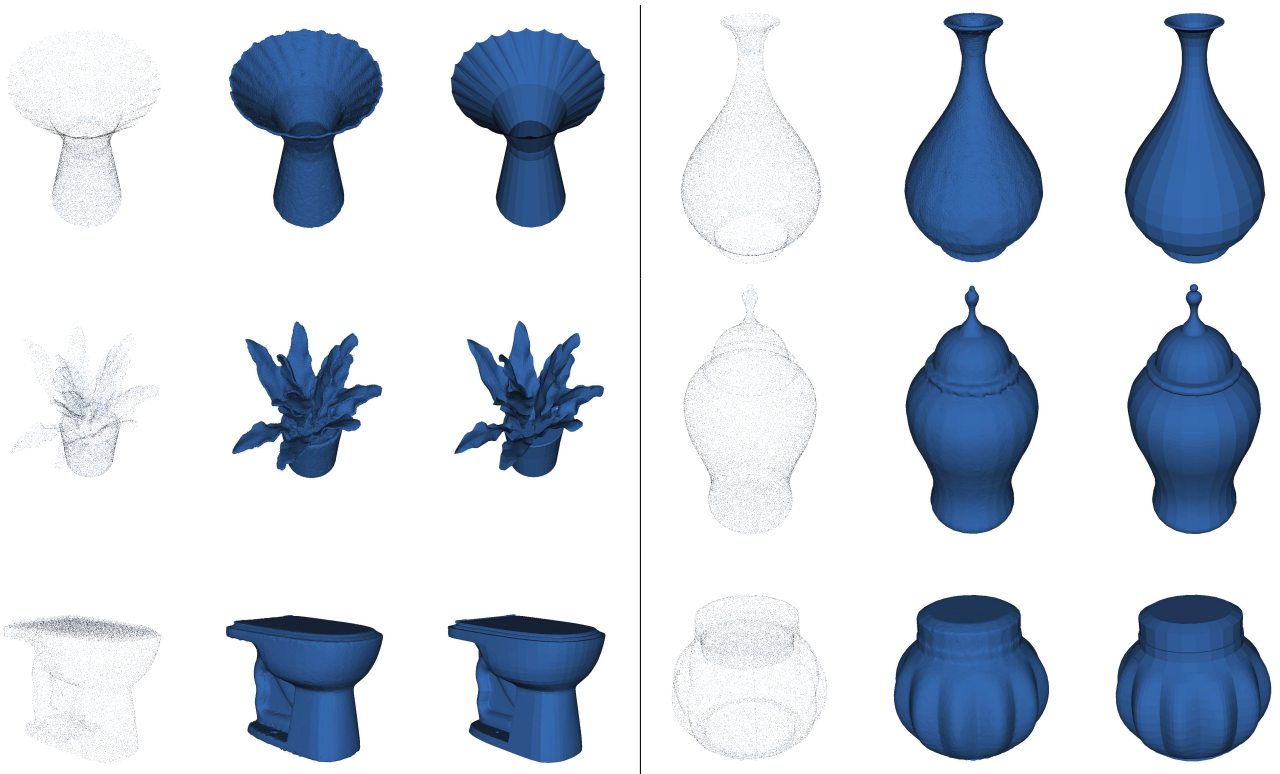

Figure 3. Results of our method on synthetic test data. Our method managed to obtain surface reconstructions comparable to the ground truth. In general, our algorithm performed better on the synthetic than on the real-world data, due to the same sampling method of the input point clouds in the synthetic training and test sets. Slight errors occurred for very thin objects (such as leaves), or for concave parts with high curvature (right middle example), which are rare in this data set, and thus hard to predict during test time.

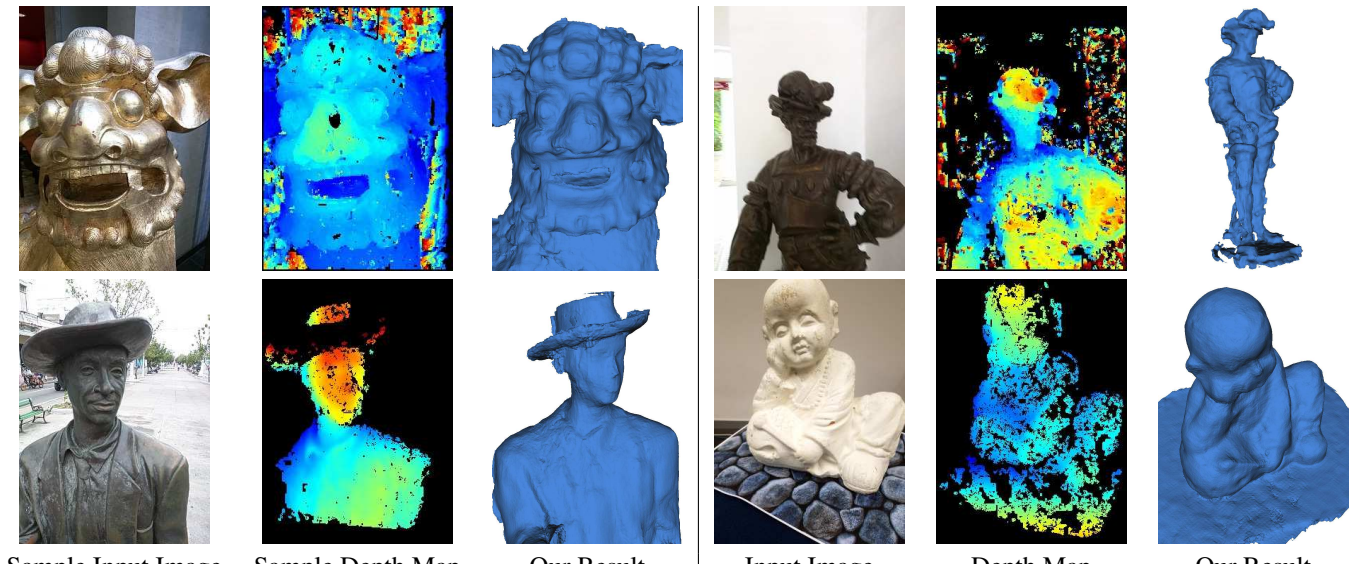

Figure 4. Results of our method on challenging real-world mobile-captured data, that includes non-Lambertian specular surfaces. Despite a high level of noise and low density of points in reflective areas(the hat of a statue), our method managed to obtain reasonable results.

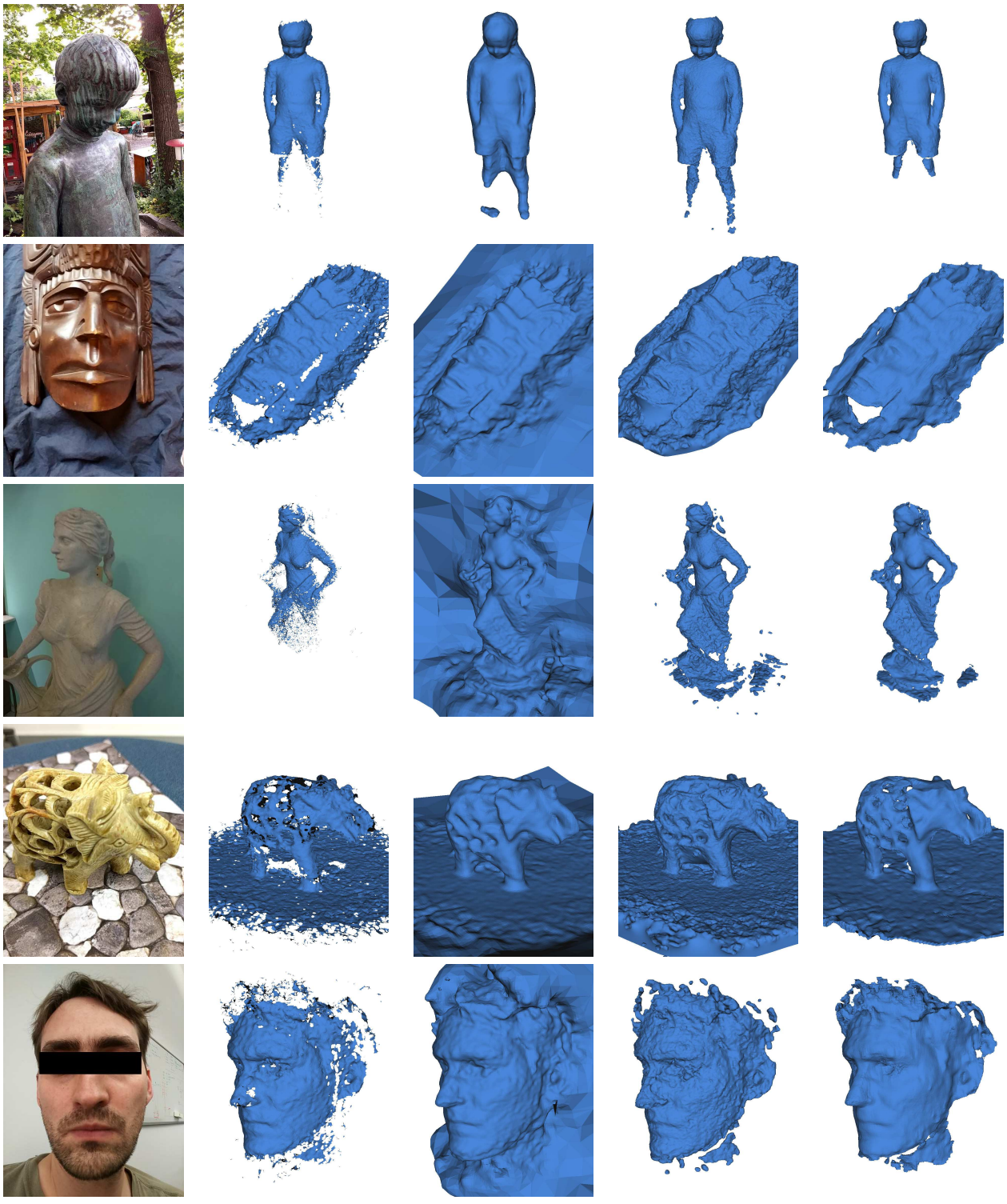

Figure 5. Qualitative comparisons of our method with other approaches. TSDF [7] models the signed distance field only locally, which results in large number of holes in the resulting 3D model. Poisson reconstruction [12] is not very robust to outliers and the extrapolated surfaces fitting the noise are often not visually pleasant. TV-L1 [34] regularization tries to find a minimal surface, that fits all points. This often leads to filling the holes, that should not be filled (for example in the elephant model). In general this method gives the best results, however, for high resolution models it takes several hours to compute. Our method often led to smoother results, probably due to averaging in the regression forest training. This property is desired for lower quality point clouds and improved the result for the face reconstruction example, but on the other hand led to the loss of details in the mask 3D model.

References

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *2009 IEEE 12th international conference on computer vision*. IEEE, 2009. 1

[2] R. C. Bolles and M. A. Fischler. A ransac-based approach to model fitting and its application to finding cylinders in range data. In *IJCAI*, volume 1981, pages 637–643, 1981. 4

[3] L. Breiman. Random forests. In *Machine Learning*, 2001. 2, 3

[4] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 67–76, New York, NY, USA, 2001. ACM. 2

[5] Z.-Q. Cheng, Y.-Z. Wang, B. Li, K. Xu, G. Dang, and S.-Y. Jin. A survey of methods for moving least squares surfaces. In *VGTC Conference on Point-Based Graphics*, 2008. 1

[6] Y. L. Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems 2*, 1990. 2

[7] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996. 1, 6, 8

[8] V. Estellers, M. Scott, K. Tew, and S. Soatto. Robust poisson surface reconstruction. In *Scale Space and Variational Methods in Computer Vision: 5th International Conference, SSVM 2015, Lège-Cap Ferret, France, May 31 - June 4, 2015, Proceedings*, 2015. 2

[9] M. Firman, O. Mac Aodha, S. Julier, and G. J. Brostow. Structured prediction of unobserved voxels from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 2

[10] S. Fuhrmann and M. Goesele. Fusion of depth maps with multiple scales. In *ACM Transactions on Graphics (TOG)*, 2011. 1

[11] 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*, 2011. 1

[12] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006. 2, 8

[13] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 83–86. IEEE, 2009. 4

[14] K. Kolev, P. Tanskanen, P. Speciale, and M. Pollefeys. Turning mobile phones into 3d scanners. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3946–3953. IEEE, 2014. 1, 5

[15] L. Ladicky, S. Jeong, B. Solenthaler, M. Pollefeys, and M. Gross. Data-driven fluid simulations using regression forests. *Transactions on Graphics*, 34(6), 2015. 2

[16] L. Ladicky, C. Russell, P. Kohli, and P. H. S. Torr. Associative hierarchical CRFs for object class image segmentation. In *International Conference on Computer Vision*, 2009. 3

[17] M. Niessner, M. Zollhofer, S. Izadi, and M. Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 2013. 1

[18] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):756–770, 2004. 4

[19] P. Ondrúška, P. Kohli, and S. Izadi. Mobilefusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones. *IEEE transactions on visualization and computer graphics*, 21(11):1251–1258, 2015. 1

[20] R. Poranne, C. Gotsman, and D. Keren. 3D Surface Reconstruction Using a Generalized Distance Function. *Computer Graphics Forum*, 2010. 2

[21] D. J. Rezende, S. M. A. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess. Unsupervised learning of 3d structure from images. *CoRR*, 2016. 2

[22] E. Rosten. Fast corner detection. *Engineering Department, Machine Intelligence Laboratory, University of Cambridge. Available from:< http://mi. eng. cam. ac. uk/er258/index. html*, 2006. 4

[23] N. Savinov, L. Ladicky, C. Hane, and M. Pollefeys. Discrete optimization of ray potentials for semantic 3d reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, 2015. 1

[24] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 2002. 1

[25] T. Schöps, T. Sattler, C. Häne, and M. Pollefeys. 3d modeling on the go: Interactive 3d reconstruction of large-scale scenes on mobile devices. In *3D Vision (3DV), 2015 International Conference on*, pages 291–299. IEEE, 2015. 1

[26] J. Shotton, J. Winn, C. Rother, and A. Criminisi. *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision*, 2006. 3

[27] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. Live metric 3d reconstruction on mobile phones. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 65–72, 2013. 1, 4

[28] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *Conference on Computer Vision and Pattern Recognition*, 2012. 2

[29] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustmentâĂŤa modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999. 4

[30] B. Ummenhofer and T. Brox. Global, dense multiscale reconstruction for a billion points. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015. 1

[31] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004. 3

[32] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 4

[33] C. Zach. Fast and high quality fusion of depth maps. In *Proceedings of the international symposium on 3D data processing, visualization and transmission (3DPVT)*, volume 1, page 2. Citeseer, 2008. 1

[34] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust tv-l 1 range image integration. In *2007 IEEE 11th International Conference on Computer Vision*. IEEE, 2007. 1, 8