Large Scale Outdoor Scene Reconstruction and Correction with Vision

Michael Tanner¹, Pedro Piniés¹, Lina María Paz¹, Ştefan Săftescu¹, Alex Bewley¹, Emil Jonasson² and Paul Newman¹

Figure 1. This paper is about the efficient generation of dense models of multiple-km-scale environments from stereo and/or laser data. Pictured is an example reconstruction from the 1.6 km Oxford Broad Street dataset released with this paper.

Abstract
We provide the theory and the system needed to create large-scale dense reconstructions for mobile-robotics applications – this stands in contrast to the object-centric reconstructions dominant in the literature. Our BOR²G system fuses data from multiple sensor modalities (cameras, lidars, or both) and regularizes the resulting 3D model. We use a compressed 3D data structure which allows us to operate over a large scale. In addition, because of the paucity of surface observations by the camera and lidar sensors, we regularize over both the 2D (camera depth maps) and 3D (voxel grid) to provide a local contextual prior for the reconstruction. Our regularizer reduces the median error between 27% to 36% in 7.3 km of dense reconstructions with a median accuracy between 4 cm to 8 cm. Our pipeline does not end with regularization. We take the unusual step to apply a learned correction mechanism which takes the global context of the reconstruction and adjusts the constructed mesh, addressing errors that are pathological to the first pass camera-derived reconstruction.

We evaluate our system using the Stanford Burghers of Calais, Imperial College ICL-NUIM, Oxford Broad Street (released with this paper), and the KITTI datasets. These latter data sets see us operating at a combined scale and accuracy not seen in the literature. We provide statistics for the metric errors in all surfaces created compared to those measured with 3D lidar as ground truth. We demonstrate our system in practice by reconstructing the inside of the EUROfusion Joint European Torus (JET) fusion reactor, located at the Culham Centre for Fusion Energy (UK Atomic Energy Authority) in Oxfordshire.

Keywords
Dense Reconstruction, Regularization, Mapping

1 Introduction and Previous Work
This paper is about the efficient generation of dense, colored models of very-large-scale environments from stereo cameras, laser data, or a combination thereof. Better maps make for better understanding; better understanding leads to better robots, but this comes at a cost. The computational and memory requirements of large dense models can be prohibitive.

Over the past few years, the development of 3D reconstruction systems has undergone an explosion facilitated by the advances in GPU hardware. Earlier, large-scale efforts such as Pollefeys et al. (2008); Agarwal et al. (2009); Furukawa et al. (2010) reconstructed sections of urban scenes from unstructured photo collections. The ever-strengthening and broadening theoretical foundations of

¹Oxford Robotics Institute, University of Oxford, UK
{mtnner,ppini,linapaz,steфан beware,pnewman}@robots.ox.ac.uk
²RACE, Culham Science Centre, Abingdon, UK
emil.jonasson@ukaea.uk

Corresponding author:
Ştefan Săftescu, Oxford Robotics Institute, University of Oxford, 17 Parks Road, Oxford, OX1 3PJ, UK.
Email: stefan@robots.ox.ac.uk

Prepared using sage.cls [Version: 2016/06/24 v1.10]
continuous optimization (Chambolle and Pock 2011; Goldluecke et al. 2012), upon which the most advanced algorithms rely, have become accessible for robotics and computer vision applications. Together these strands – hardware and theory – allow us to build systems which create large-scale 3D dense reconstructions.

However, the state of the art of many dense 3D reconstruction systems rarely considers scalability for the practical use in mapping applications such as autonomous driving or inspection. The most general approaches are motivated by the recent mobile phone and tablet development (Klingensmith et al. 2015; Engel et al. 2015; Schöps et al. 2015) with an eye on small-scale reconstruction.

Some researchers have worked with data and sensors more applicable for autonomous vehicles. Most notable, in 2010 Google released an academic article detailing their “Street View” application that utilises laser and camera data to create dense 3D reconstructions of cities across the world (Anguelov et al. 2010). However, their algorithms overemphasize the laser data and assume all depth maps only contain piecewise planar objects. In 2013, Google presented an alternative Structure from Motion approach using only camera sensors (Klingner et al. 2013). In 2014, Xiao and Furukawa proposed a system to model large-scale indoor environments with laser and image inputs, but their modified Manhattan-world assumption restricted reconstructions from modelling anything other than vertical and horizontal planes. Bok et al. (2014) built upon the prior state-of-the-art to create large-scale 3D maps using camera and lasers. Their final reconstructions were sparse and only utilised the cameras for odometry and loop closure, not as an additional depth sensor to improve the dense reconstructions.

To this end, we propose a dense mapping system which meets the following requirements:

**R1**: Operate in multiple-kilometers-scale environments

**R2**: No range limitations for input sensor observations

**R3**: Cope with a paucity of surface observations

**R4**: Fuse data from multiple sensor modalities

These were designed to support our autonomous-driving applications. A survey of the literature found no systems currently exist which meet these requirements.

Four seminal systems (Table 1) in this area are Dense Tracking and Mapping (DTAM) by Newcombe et al. (2011b); Kinect Fusion by Newcombe et al. (2011a); Kintinuous by Whelan et al. (2012); and Voxel Hashing by Niélner et al. (2013).

Before RGB-D cameras were widely available, DTAM presented a method to produce high-quality depth maps with a monocural camera. A cost volume is constructed in front of the camera’s focal plane and continually updated with 2D-regularized depth estimates from sequential image frames. The final reconstructions provide fine details, but this system limits the range of the monocural camera to near-field reconstructions.

In 2010, the first commodity RGB-D camera was released. RGB-D cameras provide a centimeter-level-accurate depth measurement for each pixel in the image – 640x480 resolution at 30 Hz in this first device. Newcombe, et al. extended their work via the Kinect Fusion system to take advantage of this stream of high-frequency and high-quality depth maps. Leveraging the Truncated Signed Distance Function (TSDF) (Curless and Levoy 1996), depth observations are stored in a voxel grid where each voxel contains a positive number when in front of a surface and

<table>
<thead>
<tr>
<th>Technique</th>
<th>R1: Env.</th>
<th>R2: Range</th>
<th>R3: Data</th>
<th>R4: Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTAM†</td>
<td>1 m²</td>
<td>1 m</td>
<td>100+ img/m²</td>
<td>Mono</td>
</tr>
<tr>
<td>Kinect Fusion</td>
<td>7 m²</td>
<td>2 m</td>
<td>-</td>
<td>RGB-D</td>
</tr>
<tr>
<td>Kintinuous</td>
<td>∞</td>
<td>3 m</td>
<td>-</td>
<td>RGB-D</td>
</tr>
<tr>
<td>Voxel Hashing</td>
<td>∞</td>
<td>4 m</td>
<td>-</td>
<td>RGB-D</td>
</tr>
<tr>
<td>BOR²G†</td>
<td>∞</td>
<td>50 m</td>
<td>0.05 img/m²</td>
<td>Mono, Stereo or Lidar</td>
</tr>
</tbody>
</table>

It should be noted that the above BOR²G Burghers reconstruction has a median point-to-surface difference of 0.5 cm. Section 7 provides a more detailed quantitative analysis our reconstruction precision.

Figure 2. RGB-D vs KITTI reconstruction detail comparison. Both reconstructions were created with the BOR²G pipeline, but the Stanford reconstruction (Zhou and Koltun 2013) has over 3,100 times (Table 5) more depth observations per square meter than the KITTI reconstruction.
a negative number behind the surface. Solving for the zero-value level set results in a dense model of the original surface. Thus, Kinect Fusion could generate unprecedented-quality dense 3D reconstructions in real time for workspaces approximately 7 m³ in size.

In contrast to Kinect Fusion where the voxel grid is fixed to one location in space, Kintinuous sought to extend the size of reconstruction scenes by allowing the voxel grid to move with the camera. A continual stream of previously-observed regions are streamed to disk as a mesh, but can be reloaded back into the GPU if that region is observed again. This system theoretically infinitely extended the reconstruction workspace size. However, it cannot leverage sensor observations further than 3 m from the RGB-D camera because it is still fundamentally based upon a conventional fixed-size voxel grid.

Nießner’s Hashing Voxel Grids (HVG) also extended the size of reconstructions but did so instead by only allocating voxels in regions where surfaces were observed. This removed memory wasted storing free space. When combined with streaming of data to/from the GPU and hard disk when surfaces are “far” away from the sensor, there is essentially no limit on the size of reconstructions. This implementation restricts the sensor range to 4 meters as that is near the maximum effective range of the Kinect camera, however the range can be trivially extended. Solutions such as Whelan et al. (2014) leverage a rolling cyclical buffer as the volumetric-reconstruction data structure. This is an interesting approach that allows local volumetric regions to virtually translate as the camera moves through an environment. The correct choice of an efficient volumetric data structure has also gained attention in the deep learning context where it impacts the resolution of 3D tasks, including 3D object classification, orientation estimation, and point-cloud labeling Riegler et al. (2017)

Most of the above techniques focused on object-centered reconstructions. In other words, the operator selected a small scene to reconstruct (e.g., desk, courtyard, etc.) and a single sensor was moved through the environment in a way that observed all surfaces multiple times from a variety of angles. This results in a fine-detailed final reconstruction, as shown in Figure 2a. In autonomous driving scenarios where the sensor is mounted on a vehicle and the path is not known a priori, the surfaces are observed fewer times. In fact, we found that RGB-D scenarios have over 3,100 times more depth images per square meter than our autonomous vehicle applications (Table 5).

In addition, mobile-robotics sensor suites typically include lidars and forward-facing monocular or stereo cameras. Since the viewing range of RGB-D cameras is so short (5 meters) and their accuracy degrades outdoors, they are not useful when reconstructing an urban environment from a mobile-robotics platform. Cameras on mobile-robotics platforms produce significantly less accurate depth measurements. Over time, lidar sensors usually have a smaller reconstruction field of view and depth-observation density than a similarly placed camera sensor. Both camera and lidar sensors observe surfaces far fewer times than in a typical RGB-D scenario. Therefore the reconstructions inherently have less detail, as shown in Figure 2b, which requires us to apply contextual priors via a local regularizer to improve the final surface reconstructions.

Table 1, Table 5, and Figure 2 summarize the discussion in the preceding literature review.

Our contributions are as follows:

1. We present in Sections 2–6 the theory required to construct a state-of-the-art dense-reconstruction pipeline for mobile-robotics applications (i.e., satisfies R1–R4 previously defined in this section). Our implementation includes a compressed data structure (R1/R2) of a sensor-agnostic voxel grid (R4). Since the input data may be noisy and the observations per m² are very low, we utilize a regularizer in 2D and 3D to both serve as a prior and reduce the noise in the final reconstruction (R3).

2. We present in Sections 5.3–5.4 a method to regularize 3D data stored in a compressed volumetric data structure thereby enabling optimal regularization of significantly larger scenes (fulfills R1 and R3). The key difficulty (and hence our vital contribution) in regularizing within a compressed structure is the presence of many additional boundary conditions introduced between regions which have and have not been directly observed by the range sensor. Accurately computing the gradient and divergence operators, both of which are required to minimize the regularizer’s energy, becomes a non-trivial problem. We ensure the regularizer only operates on meaningful data and doesn’t extrapolate into unobserved regions.

3. We present in Section 6 a method to adjust reconstructions and correct gross errors with priors learnt from high-fidelity historical data (e.g. roads generally don’t have holes in them, cars are of certain shapes, etc.) We show that appearance and geometry features can be extracted from a 3D reconstruction, and better depth maps can be produced with a neural network that applies these priors. We illustrate how this step can be included in our dense-reconstruction pipeline to produce more comprehensive meshes.

4. We provide, by way of our quantitative results in Section 7 and the models released in the supplementary materials, a new KITTI benchmark for the community to compare dense-map reconstructions. In addition, we include the ground truth (GT) data used to evaluate our system. This includes the optimized vehicle trajectory, consolidated GT pointcloud, and final dense reconstructions with different sensor modalities.

5. We release (Section 7.3) the 1.6 km Oxford Broad Street dataset (partial example reconstruction in Figure 1) as a real-world reconstruction scenario for a mobile-robotics platform with a variety of camera and lidar sensors.

Our hope is that our dataset and benchmarks become valuable tools of comparison within the community.

This article builds upon our prior work (Tanner et al. 2015, 2018) by enabling regularization in a compressed 3D data structure; evaluating the relative reconstruction quality based on using combinations of camera and laser sensors; documenting the performance of SDF and histogram data
2 System Overview

This section provides a brief overview of our system’s architecture (Figure 3) before we proceed to a detailed discussion in Sections 3–6.

At its core, our system consumes a stream of stereo camera images (I) or laser scans (L) and produces a 3D model. The pipeline consists of a Localization module which provides sensor pose estimates. We place no requirements on the trajectory of the source sensors; indeed, we illustrate our method using images captured from a forward-facing camera on a road vehicle – an ill-conditioned use-case which is challenging yet likely given the utility of forward-facing cameras in navigation and perception tasks. The design of this module is versatile allowing us to pick any arbitrary SLAM derivative; we use ORB-SLAM2 (Mur-Artal and Tardós 2015) as it is a widely used, open source benchmark tool for localization and loop closure with monocular, stereo, and RGB-D cameras.

The Depth-Map Estimation module (Section 4) processes a stream of stereo or monocular frames into a stream of 2D regularized depth maps, D. Because of the nature of a passive camera and the large scale at which we operate, we utilize a sophisticated regularizer – specifically, the Total Generalized Variation (TGV) regularizer – to improve the accuracy of these depth maps.

The Dense Fusion module (Section 5) merges the regularized depth maps into a compressed data structure as a Signed Distance Function (SDF), \( f : \Omega_3 \rightarrow \mathbb{R} \), where \( \Omega_3 \subset \mathbb{R}^3 \) represents a subset of points in 3D space. New incoming data can be added at any time from any suitable sensor source. Even with the regularized input depth maps, the resulting reconstruction tends to contain spurious and noisy surfaces. Our 3D Total Variation (TV) regularizer operates over \( \Omega_3 \) to generate the optimized, denoised SDF field, \( u : \Omega_3 \rightarrow \mathbb{R} \). This regularizer also serves as a prior to aid in the reconstruction even with a paucity of direct surface observations (Figure 2).

The CNN Correction step (Section 6) further improves reconstructions created from low quality or scarce data. Our approach corrects the 3D geometry based on prior knowledge of what certain scenarios are expected look like when reconstructed. We achieve this by projecting various features in training 3D reconstructions into 2D images and using a CNN to learn the differences between high and low quality data. Learnt error correction, along with regularization in both 2D and 3D are the salient features of our system.

A final surface model is extracted to be processed in parallel by a separate application (e.g., segmentation, localization, planning, or visualization).

3 Optimization with Total Variation

In this paper we address the inverse problem of estimating the 3D dense structure of an environment given a set of 2D images and/or 3D sparse laser scans. Inverse problems deal with the estimation of unknown quantities \( u \) (structure in our case) given their effects \( f \) (images or laser data). By their nature, these problems are ill-posed which means that at least one of the following three requirements is not met: existence of a solution, uniqueness of the solution, or stability of the solution (i.e., small changes in input data produce small changes in the result). Clearly the problem we try to solve is ill-posed. As an extreme example, if we want to reconstruct a white wall from a set of images, we cannot guarantee uniqueness and stability (walls at different distances and orientations will produce the same observations).

The properties of the problem can improve if we limit the space of meaningful solutions by imposing regularity assumptions or prior knowledge about the solution. Energy-based minimization approaches model these requirements by using an energy regularization term that favours certain solutions and a data term that models how well the searched solution \( u \) explains the set of observations \( f \). Equation 1 synthesizes this approach,

\[
\min_u E_{\text{reg}}(u) + E_{\text{data}}(f, u) \tag{1}
\]

where \( E_{\text{reg}}(u) \) is the regularisation or “smoothness” term and \( E_{\text{data}}(f, u) \) is the data-fidelity term.

In this paper, we use a first order \((\alpha = 1)\) and a second order \((\alpha = 2)\) Total Generalised Variation (TGV\(^\alpha\)) regularization term, respectively, in the optimizations.
performed in the Depth-Map Estimation and the 3D Dense Fusion blocks in Figure 3. As we will discuss later, these regularizers allow us to incorporate prior information about the type of surfaces we expect to encounter in the traversed environment.

Let us now present a simple denoising problem that will help us illustrate the Primal-Dual algorithm Chambolle and Pock (2011) implemented to solve Equation 1 in the following sections. For simplicity, we employ a first order TV$^1$ regularizer also known as Total Variation (TV). Figure 4 left shows the effect of the TV on a noisy 1D signal defined in the bounded domain $\Omega_1 = [t_0, t_n]$ where $\Omega_1 \subset \mathbb{R}$. The energy to be minimized is given by,

$$E(u) = \int_{\Omega_1} |\nabla u| d\Omega_1 + \frac{\lambda}{2} \int_{\Omega_1} ||u - f||_2^2 d\Omega_1$$

where $\int_{\Omega_1} |\nabla u| d\Omega_1$ is the Total Variation of the solution and, in this case, we use a quadratic penalty for the data term. Notice that the TV accumulates the magnitude of local variations in the signal and therefore measures its total length. Intuitively the TV term tends to reduce signal’s ripples and therefore filters out high frequency noise while maintaining big discontinuities present in the data term. In the extreme case when no data term is available, TV would produce a constant solution—maintaining big discontinuities present in the data term. rippled and therefore filters out high frequency noise while preserving sharp discontinuities. We impose zero Neumann boundary conditions that assume the signal remains constant outside the left and right boundaries. (Right) The original non-differentiable energy minimization is transformed into a smooth Primal-Dual problem by applying the Legendre-Fenchel transform to the TV term. The resulting saddle point problem can be solved by alternating variants of gradient descent and gradient ascent steps for the primal $u$ and dual $p$ variables, respectively.

$$\nabla u_i = \begin{cases} u_{i+1} - u_i & \text{if } 0 \leq i < n \\ 0 & \text{if } i = n \end{cases}$$

To simplify the explanation let us minimize Equation 3 w.r.t. the $i\text{th}$ sample in the summation,

$$\min_{u_i} || \nabla u_i ||_2 + \frac{\lambda}{2} ||u_i - f_i||_2^2$$

Although this problem is convex and therefore its global minimum can be computed, standard optimization algorithms like gradient descent or Newton and quasi-Newton methods cannot be applied directly because they require smooth cost functions to assure convergence (Nocedal and Wright 2006). In this case, the gradient is not defined at the kink of the absolute value. The Legendre-Fenchel transform (Rockafellar 1970) provides an elegant way to convert the L1 norm into a maximization problem that, although more complex, is differentiable (see Figure 5 for an intuitive explanation),

$$|| \nabla u_i || = \sup_{|p_i| \leq 1} < \nabla u_i, p_i >$$

where the operator $< \bullet, \bullet >$ computes the inner product of its operands and $p_i$ is known as the dual variable which represents for each $u_i$ the sub-gradient of the L1 norm. Substituting Equation 6 into Equation 5 we obtain,

$$\min_{|p_i| \leq 1} \sup_{u_i} < \nabla u_i, p_i > + \frac{\lambda}{2} ||u_i - f_i||_2^2$$

which is a saddle point problem with equivalent primal and dual solutions since strong duality holds (Boyd and Vandenberghe 2004). Figure 4 right shows an example of a saddle point problem in $u$ and $p$ with zero duality gap.

An equivalent energy expression, that will be useful in the derivation of the optimization algorithm, can be obtained by calculating the adjoint $\text{Adj}(\bullet)$ of the linear operator $\nabla$. When $\nabla u$, and therefore $p$, is $n$-dimensional (with $n > 1$), the adjoint of the gradient is equal to the negative of the divergence operator: $\text{Adj}(\nabla)p = -\nabla \cdot p = -(\nabla |p|^1 + \nabla |p|^2)$.
Equation 7 as to divergence – the value of the variable at the borders is differences and Dirichlet boundary conditions for the boundary conditions for the gradient become backward solution | non-differentiable expression By definition a linear operator and its adjoint fulfill, algorithms such as projected gradient ascent. the last component of \( p \) will use as well (\( \nabla \cdot p \)) = \( \nabla_j p_j \). By definition a linear operator and its adjoint fulfill, \[
\langle \nabla u_i, p_i \rangle \geq - \langle u_i, \nabla \cdot p_i \rangle \quad (8)
\]
To satisfy this equation, forward differences and Neumann boundary conditions for the gradient become backward differences and Dirichlet boundary conditions for the divergence – the value of the variable at the borders is, \[
(\nabla \cdot p_i) = \begin{cases} 
\frac{p_i}{p_{i-1}} & \text{if } 0 < i < n \\
\frac{p_i}{p_{i-1}} & \text{if } i = 0 \\
-p_{i-1} & \text{if } i = n
\end{cases} \quad (9)
\]
With a little abuse of notation, we will use the divergence operator for the current 1D example (\( j \) can only be equal to 1). Using the adjoint of the gradient we can rewrite Equation 7 as
\[
\min_{u_i} \sup_{|p_i| \leq 1} - \langle u_i, \nabla \cdot p_i \rangle + \frac{\lambda}{2} ||u_i - f_i||^2 \quad (10)
\]
The Primal-Dual optimization algorithm (Chambolle and Pock 2011) to solve Equation 7 or its equivalent Equation 10 consists of the following steps:

1. \( p_i, u_i, \) and \( \hat{u}_i \) are initialised to 0. \( \hat{u}_i \) is a temporary variable which reduces the number of optimization iterations required to converge.
2. To solve the maximisation we use a projected gradient ascent algorithm. The gradient of Equation 7 with respect to \( p_i \) is just \( \nabla \hat{u}_i \) (as we explained before we will use \( \hat{u}_i \) instead of \( u_i \) to up the convergence of the algorithm, more details in Chambolle and Pock 2011). Therefore \( p_i^{\text{new}} = p_i + \sigma \nabla \hat{u}_i \) where \( \sigma \) is the dual variable’s gradient-ascent step size. Since \( p_i^{\text{new}} \in [-1, 1] \) we project the solution into the feasible set using a projection operator \( \pi(p_i^{\text{new}}) \). Therefore the maximization step is given by,
\[
p_i = \pi(p_i + \sigma \nabla \hat{u}_i) \quad (11)
\]
3. The minimization is based on a look-ahead gradient descent algorithm. In this case it is easier to calculate the gradient with respect to \( u_i \) from Equation 10. With look-ahead we mean that the gradient is not evaluated at the current solution but at the new estimate \( u_i^{\text{new}} \) after the step is taken. Deriving Equation 10 w.r.t. \( u_i \) and evaluating at \( u_i^{\text{new}} \) the gradient is given by: \(-\nabla \cdot p_i + \lambda(u_i^{\text{new}} - f_i)\). Applying a gradient descent step with step size \( \tau \) we obtain,
\[
u_i^{\text{new}} = u_i - \tau(-\nabla \cdot p_i + \lambda(u_i^{\text{new}} - f_i)) \quad (12)
\]
which gives an implicit equation in \( u_i^{\text{new}} \). Rearranging terms, the primal step is,
\[
u_i^{\text{new}} = \frac{u_i + \tau \nabla \cdot p_i + \tau \lambda f_i}{1 + \tau \lambda} \quad (13)
\]
4. To reduce the number of required iterations we apply the following “relaxation” step,
\[
\hat{u}_i = u_i + \theta(u_i - \hat{u}_i) \quad (14)
\]
where \( \theta \) is a parameter to adjust the relaxation step size.
5. Finally, we repeat steps 2–4 until convergence.
In the following two sections we will make use of the Primal-Dual algorithm to regularize both a stream of 2D depth maps (Section 4) and the final 3D dense reconstruction (Section 5).

4 Depth-Map Estimation

Given a pair of rectified images captured from a stereo camera, we implement an algorithm based on Ranftl...
et al. (2012) that allows us to calculate very high quality disparity maps. Although many other dense stereo approaches exist in the literature (a nice survey is presented by Scharstein and Szeliski 2002), most of the solutions require explicit strategies to handle occlusions. Instead, we implement an energy minimization approach with an L1-fidelity data term that provides some robustness against outliers from occlusions while handling varying illumination and radiometric differences between the cameras. For regularization we use a second-order regularizer term that favours piecewise planar disparity maps and allows sub-pixel accurate solutions. The energy to be minimized is,

\[ E(d) = E_{\text{reg}}(d) + E_{\text{data}}(d; I_L, I_R) \]

where \( d \) is the searched disparity image defined in \( \Omega_2 \subset \mathbb{R}^2 \).

### 4.1 Data Term

To improve robustness against outliers we implement an L1-norm penalty term that penalizes photo-consistency differences between matched pixels in the images,

\[ E_{\text{data}}(d; I_L, I_R) = \int_{\Omega_2} |\nabla C_W(I_L(x + d, y), I_R(x, y))| \, d\Omega_2 \]

In particular \( \nabla C_W(\bullet) \) stands for the difference between the Census Transform Signature (CTS) (Zabih and Woodfill 1994) of two pixels calculated in a window of size \( W \) for a candidate disparity \( d \). This metric has been shown to be both illumination invariant and fast to compute in comparison to other metrics such as the Sum of Absolute Distances (SAD), the Sum of Square Distances (SSD) or the Normalized Cross Correlation (NCC) (Zabih and Woodfill 1994). Its computation is simple: given a pixel, the CTS computes a bit string by comparing the chosen pixel with the local window \( W \) centered around it. A bit is set to 1 if the corresponding pixel has a lower intensity than the pixel of interest. The difference between two windows \( \nabla C_W(\bullet) \) is then given by the Hamming distance between the two bit strings defining them.

### 4.2 Affine Regularization

As we explained in the previous section, for ill-posed problems, like depth-map estimation, good and apt priors are essential – whether the prior is task-specific and bespoke (Güney and Geiger 2015) or more general. A common choice is to use TV regularization as a prior to favor piecewise-constant solutions. However, its use lends itself to poor depth-map estimates over outdoor sequences because it assumes fronto-parallel surfaces. Figure 6 shows some of the artifacts created after back-projecting the point cloud for planar surfaces not orthogonal to the image plane (e.g. the roads and walls which dominate our urban scenes). Thus we reach for a second order Total Generalized Variation (TGV\(^2\)) regularization term (Bredies et al. 2010) which favours affine surfaces,

\[ \min_{d \in \mathbb{R}, w \in \mathbb{R}^2} \alpha_1 \int_{\Omega_2} |\nabla d - w| \, d\Omega_2 + \alpha_2 \int_{\Omega_2} |\nabla w| \, d\Omega_2 \]  

where \( w \) allows the disparity \( d \) in a region of the depth map to change at a constant rate and therefore creates planar surfaces with different orientations. \( \alpha_1 \) and \( \alpha_2 \) are free parameters (Table 2).

To minimize this problem using the Primal-Dual algorithm explained in the previous section, we apply the Legendre-Fenchel transform to both summands in Equation 17 so that
both terms become differentiable,

\[
|\nabla d - w| = \sup_{|p|_w \leq 1} <\nabla d - w, p > \\
|\nabla w| = \sup_{|q|_w \leq 1} <\nabla w, q >
\]

(18)

Notice that, this time, two additional dual variables \(p \in \mathbb{R}^2\) and \(q \in \mathbb{R}^2\) are added to the optimization.

### 4.3 Leveraging Appearance

A common problem that arises during the energy minimization is the resulting tension between preserving object discontinuities while respecting the smoothness prior. Ideally the solutions preserve intra-object continuity and inter-object discontinuity. For example, in Figure 6 we desire to accurately estimate the depths of both the rubbish bin and automobile (intra-object consistency), but we also must account for their position relative to one another (inter-object discontinuity).

One may mitigate this tension by using an inhomogeneous diffusion coefficient \(c(|\nabla I|)\) in the energy as an indicator of boundaries between objects,

\[
\begin{align*}
\min_{d \in \mathbb{R}, w \in \mathbb{R}^2} & \quad \alpha_1 \int_{\Omega_2} c(|\nabla I|) \nabla d - w \, d\Omega_2 + \alpha_2 \int_{\Omega_2} |\nabla w| \, d\Omega_2 \\
\text{subject to} & \quad w_n \leq d_n \\
\end{align*}
\]

where \(c(|\nabla I|)\) is defined as

\[
c(|\nabla I|) = \exp(-\gamma |\nabla I|^\beta)
\]

However, though this aids the regularizer, it does not contain information about the direction of the border between the objects. To take this information into account we adopt an anisotropic diffusion tensor,

\[
\begin{align*}
\min_{d \in \mathbb{R}, w \in \mathbb{R}^2} & \quad \alpha_1 \int_{\Omega_2} |T \nabla d - w| \, d\Omega_2 + \alpha_2 \int_{\Omega_2} |\nabla w| \, d\Omega_2 \\
\text{subject to} & \quad w_n \leq d_n \\
\end{align*}
\]

with,

\[
T = \exp(-\gamma |\nabla I|^\beta) n T + n^\top n^T
\]

Equation 22 is to decompose the components of the disparity gradient \(\nabla d\) in directions aligned with the gradient of the image \(n\) and orthogonal to it \(n^\perp\). We do not smooth depth discontinuities aligned with \(n\) but we penalize large image gradient components aligned with \(n^\perp\). In other words, if there is a discontinuity visible in the color image, then it is highly probable that there is a discontinuity in the depth image. The benefit of this tensor term are visually depicted in Figure 6.

### 5 Dense 3D Fusion

The core of the 3D dense mapping system consists of a Dense Fusion module which integrates a sequence of depth/range observations (depth maps from Section 4 or laser scans) into a volumetric representation. To smooth noisy surfaces and remove uncertain surfaces, our system carries out an energy optimization on the data volume together with 3D regularization.

...
observations represented by their corresponding Truncated Signed Distance Function (TSDF), \( \nu_{TSDF} \) (Curless and Levoy 1996). The TSDF value of each voxel is computed such that one can globally solve for the zero-crossing level set (isosurface) to find a continuous surface model. Even though the voxel grid is a discrete field, because the TSDF value stores the precise distance to the nearest surface, the surface reconstruction is even more precise than the voxel size.

Due to memory constraints, only a small subset of space (a few cubic meters) can be reconstructed using a conventional approach where the voxel grid is fixed in space (Nießner et al. 2011a, b). This presents a particular problem in mobile robotics applications since the exploration region of the robot would be restricted to a prohibitively small region. In addition, long-range depth/range sensors (e.g., laser, stereo-camera-based depth maps) cannot be fully utilised since their range exceeds the size of the voxel grid (or local voxel grid if a local-mapping approach is used) (Whelan et al. 2012), as shown in Table 1.

In recent years, a variety of techniques have been proposed to remove these limits (Whelan et al. 2012; Nießner et al. 2013; Chen et al. 2013). They leverage the fact that the overwhelming majority of voxels do not contain any valid TSDF data since they are never directly observed by the range sensor. A compressed data structure only allocates and frees memory on the GPU (via CUDA or OpenCL) as, once the memory is properly allocated, no atomic operations are required to fuse data into the voxel grid. Each voxel can be allocated in memory, perform the following operations on every new depth map, \( D \):

1. Calculate the voxel’s global-frame center \( \mathbf{p}_g = [x_g, y_g, z_g]^T \) with respect to the camera coordinate frame as \( \mathbf{p}_c = \mathbf{T}_{ac}^{-1}\mathbf{p}_g \), where \( \mathbf{T}_{ac} \in \mathbb{SE}(3) \) is the camera-to-global coordinate frame transformation.
2. Project \( \mathbf{p}_c \) into \( D \) to determine the nearest pixel \( d_{x,y} \).
3. If the pixel \((x, y)\) lies within the depth map, evaluate \( u_{TSDF} = d_{x,y} - z_c \), \( u_{TSDF} > 0 \) indicates the voxel is between the surface and the camera, \( u_{TSDF} < 0 \) for voxels occulted from the camera by the surface, and \( u_{TSDF} = 0 \) for voxel centroids which exactly intersect the surface.
4. Update the voxel’s current \( f \) (TSDF value) and \( w \) (weight or confidence in \( f \)) at time \( k \),

\[
\begin{align*}
  w_k &= \begin{cases} 
    w_{k-1} + 1 & \text{if } u_{TSDF} \geq -\mu \\
    w_{k-1} & \text{if } u_{TSDF} < -\mu 
  \end{cases} \\
  f_k &= \begin{cases} 
    \frac{uw_{k-1} + f_{k-1}}{w_k} & \text{if } u_{TSDF} \geq -\mu \\
    f_{k-1} & \text{if } u_{TSDF} < -\mu 
  \end{cases}
\end{align*}
\]

where \( w_{k-1} \) and \( f_{k-1} \) are the previous values of \( f \) and \( w \) for that voxel.

Note we initially elected to use the SDF rather than the TSDF for \( f \) as this allows us to simultaneously fuse data from different sensors, each with distinct \( \mu \) values based on the sensor’s precision, into the same voxel grid. For example, \( \mu \) might be large for points far away from the stereo camera but smaller for laser scans or close stereo observations.

5.1.2 Laser Fusion Fusing laser data into the voxel grid is accomplished via an efficient ray casting implementation (Amanatides et al. 1987). Consider the case of a single laser ray (\( \vec{p} \)) with an origin at \( \mathbf{p}_o \) and a termination (i.e., where the laser struck a surface) point at \( \mathbf{p}_t \). \( \vec{p} \) is fused into the voxel grid by tracing the ray and updating each voxel with the signed distance to \( \mathbf{p}_t \). Specifically, the following operations are performed on all voxels which intersect the ray from \( \mathbf{p}_o \) to \( \mathbf{p}_t \):

1. Calculate the voxel’s global-frame center \( \mathbf{p}_g = [x_g, y_g, z_g]^T \)
2. Compute the vector from the laser scan’s origin to the current voxel center: \( \mathbf{v}_1 = \mathbf{p}_g - \mathbf{p}_o \)
3. Compute the vector from the current voxel center to the laser scan’s termination point: \( \mathbf{v}_2 = \mathbf{p}_t - \mathbf{p}_g \)
4. Compute the voxel’s SDF value:

\[
u_{TSDF} = \text{sgn}((\mathbf{v}_1 \cdot \mathbf{v}_2))||\mathbf{v}_2||_2 \]
If \( u_{SDF} > 0 \), the voxel is between the surface and the laser sensor, \( u_{SDF} < 0 \) indicates the surface occludes the laser sensor’s view of the voxel, and \( u_{SDF} = 0 \) indicates the voxel’s centroid exactly intersects the surface.

5. Update the voxel’s current \( f \) (SDF value) and \( w \) (weight) via Equation 23.

As with the depth-map fusion, these calculations are highly data-independent and thus suitable for parallel processing. However, in contrast to depth-map fusion, one must ensure the memory update operations are atomic since multiple laser rays may simultaneously intersect any given voxel.

To conserve space, voxel blocks are allocated in memory for the region \( \pm \mu \), but voxels between the sensor and observed surface are updated (if they exist). This removes free-space violations without increasing the memory requirements. In other words, in addition to the memory allocated to store the observed surface, we ray trace between the surface and the sensor to update any voxel blocks which may have previously been allocated when fusing depth or range data. For example, if an automobile appeared in a previous depth map but it has since moved, the full ray trace will update the SDF values in those now-empty voxels (i.e., “delete” the now-spurious automobile reconstruction).

5.2 Energy for 3D Reconstruction

Both inputs to our system pipeline, stereo-image-based depth maps and laser scans, are noisy, especially when compared to the centimeter-level-accuracy of RGB-D systems. As explained in Section 3, we pose the noise-reduction problem as a continuous energy minimization

\[
E(u) = E_{reg}(u) + E_{data}(u, f)
\]

where \( f \) represents noisy SDF data and \( u \) is the optimally denoised SDF, both of which are defined in the 3D domain \( \Omega \subset \mathbb{R}^3 \).

For the regularization term, we first implemented a second order 3D TGV\(^2 \) regularizer, – TGV\(^2 \) provided the high-quality 2D depth-map results in Section 4 – but we found that the simpler first order TV regularizer produces nearly identical results. We believe this is because the size of the voxels is very small relative to the size of the surfaces being reconstructed. The frontal-parallel and piece-wise discontinuity effects are therefore minimal, so we selected the 3D TV regularizer as it has significantly lower computational and memory requirements while producing similar results. In practice, the TV norm has a two-fold effect: (1) smooths out the reconstructed surfaces, and (2) removes surfaces which are “uncertain” – i.e., voxels with high gradients and few direct observations. This includes removing surfaces with very small surface area, but also serves to smooth out larger surfaces.

We propose two implementations for the data term: the first method (Section 5.2.1) averages all depth measurements, whereas the second method (Section 5.2.2) records the depth measurements as samples from a probability-density function in a histogram data structure.

5.2.1 SDF Data Term In the method originally proposed by Curless and Levoy (1996) and popularized by Newcombe et al. (2011a), data is fused into the voxel grid by storing in each voxel a weighted average of all depth observations. To produce good results, this method usually requires the noise to either be small or Gaussian.

The end result of fusion (Equation 23) are two numbers stored in each voxel: \( f \), the signed metric distance to the nearest surface, and \( w \), the number of observations used to compute \( f \). One may also think of \( f \) and \( w \) as storing the mean \( f \) and information \( w \) (inverse of variance) to characterize the sampled distribution. As the energy minimization seeks to solve for a denoised-version of \( f \), a simple L2 norm data term ensures the final \( u \) is consistent with the observed depths:

\[
E(u) = \int_{\Omega_3} |\nabla u| d\Omega_3 + \frac{\lambda}{2} \int_{\Omega_3} \|f - u\|^2 d\Omega_3
\]

5.2.2 Histogram Data Term In general, the more meaningful information provided to an optimizer, the better the final results. The SDF data-term method summarized all depth measurements for a voxel as two numbers: \( f \) and \( w \). At the other extreme, one could store the entire set of \( N \) observed SDFs within each voxel (Zach et al. 2007),

\[
E(u) = \int_{\Omega_3} |\nabla u| d\Omega_3 + \frac{\lambda}{2} \int_{\Omega_3} \frac{1}{N} \sum_{i=1}^{N} \|f_i - u\| d\Omega_3
\]

The main drawback with this approach is that we cannot just sequentially update a single \( f \) and \( w \) when a new depth map arrives. Instead, all previous \( u_{SDF} \) values must be stored in each voxel. This greatly limits the number of depth maps that can be fused due to memory constraints.

A more practical approach is to store depth measurements as a histogram representing the probability-density function (PDF) for that voxel rather than explicitly storing the entire history of observations or the mean and variance (Zach 2008). This histogram approach is desirable since it uses a fixed amount of memory.

With a histogram representation, each voxel contains an array with \( n_{bins} \) elements where each element, \( h_b \), stores the number of depth observations within the \( b^{th} \) histogram bin. The fusion process described in the previous section is thus augmented by linear scale-and-clamp \( f \) in the interval \([-1, 1]\). The relationship between SDF, TSDF, and histogram bins is graphically depicted in Figure 8.

The energy minimization becomes,

\[
E(u) = \int_{\Omega_3} |\nabla u| d\Omega_3 + \frac{\lambda}{2} \int_{\Omega_3} \sum_{b=1}^{n_{bins}} h_b \|c_b - u\| d\Omega_3
\]

where the center of the bins are computed as,

\[
c_b = \frac{2b}{n_{bins}} - 1
\]

This method has been demonstrated by Zach (2008) on a small-scale, object-centered environment where the final reconstruction was within a millimeter of the ground-truth laser scan with 99% completeness.
5.3 Omega Domain

Since we are moving at an a priori-unknown trajectory through the world, we only observe surfaces in a subset of all allocated voxels and voxel blocks. We therefore present a new technique to prevent the unobserved voxels from negatively affecting the regularisation results of the observed voxels — i.e., to ensure the regularizer doesn’t extrapolate into unobserved regions. In order to achieve this, as illustrated in Figure 9, we define the complete voxel grid domain as \( \Lambda \) and use \( \Omega_3 \) to represent the subset of voxels which have been directly observed and which will be regularised. The remaining subset, \( \overline{\Omega} \), represents voxels which have never been observed. By definition, \( \Omega_3 \) and \( \overline{\Omega} \) form a partition of \( \Lambda \) and therefore \( \Lambda = \Omega_3 \cup \overline{\Omega} \) and \( \Omega_3 \cap \overline{\Omega} = \emptyset \).

To the authors’ knowledge, all prior works rely on a fully-observed conventional voxel grid before regularization and they implicitly assume that \( \Lambda = \Omega_3 \) — i.e., every voxel in the voxel grid is directly observed by the sensor. In our mobile-robotics platform, this assumption is not valid. The robot motion results in unobserved regions caused by object occlusion, field-of-view limitations, and trajectory decisions. Therefore, \( \Omega_3 \subset \Lambda \) as Figure 9 illustrates. In practice, we found failure to account for the \( \Omega_3 - \overline{\Omega} \) boundaries causes the regulariser to spuriously extrapolate surfaces into undesired regions of the reconstruction.

Our approach introduces a new state variable, \( I_{\Omega_3} : \Lambda \rightarrow \{0, 1\} \), in each voxel indicating whether or not it was directly observed by a range sensor. \( \Omega_3 \) is therefore the set solely upon which the regularizer is constrained to operate. In our previous work, we found that failure to account for these boundary conditions creates spurious surfaces in non-valid voxels (\( \Omega_3 \)) (Tanner et al. 2015; Tanner et al. 2016). Note that all voxel blocks which are not allocated are in \( \overline{\Omega} \). A graphical depiction of how we augmented the HVG data structure with \( \Omega_3 \) and \( \overline{\Omega} \) is provided in Figure 7.

Since compressed voxel grid data structures are not regular in space, the proper method to compute the gradient (and its dual: divergence) in the presence of the additional boundary conditions (caused by the \( \Omega_3 \) domain and voxel-block boundaries) is not straightforward.

We define the gradient as:

\[
\nabla_x u_{i,j,k} = \begin{cases} 
  u_{i+1,j,k} - u_{i,j,k} & \text{if } 1 \leq i < V_x \\
  0 & \text{if } i = V_x \\
  0 & \text{if } u_{i,j,k} \in \Omega_3 \\
  0 & \text{if } u_{i+1,j,k} \in \overline{\Omega} 
\end{cases}
\]

(29)
where \( u_{i,j,k} \) is a voxel’s TSDF value at the 3D world integer coordinates \((i, j, k)\), and \( V_x \) is the number of voxels in the \( x \) dimension. The gradient term in the regularizer encourages smoothness across neighbouring voxels which explains why this new gradient definition excludes \( \Omega_3 \) voxels — they have not been observed.

To solve the Primal-Dual optimization (Section 3), we also need to define the corresponding divergence operator:

\[
(\nabla \cdot p_{i,j,k})^x = \begin{cases} 
  p^x_{i,j,k} - p^x_{i-1,j,k} & \text{if } 1 < i < V_x \\
  p^x_{i,j,k} & \text{if } i = 1 \\
  -p^x_{i-1,j,k} & \text{if } i = V_x \\
  0 & \text{if } u_{i,j,k} \in \Omega_3 \\
  p^x_{i,j,k} & \text{if } u_{i-1,j,k} \in \Omega_3 \\
  -p^x_{i-1,j,k} & \text{if } u_{i+1,j,k} \in \Omega_3
\end{cases} \tag{30}
\]

Each voxel block is treated as a voxel grid with boundary conditions which are determined by its neighbours’ \( \Omega_3 \) indicator function, \( I_{\Omega_3} \). The regularizer operates only on the voxels within \( \Omega_3 \), the domain of integration, and thus it neither spreads spurious surfaces into unobserved regions nor updates valid voxels with invalid SDF data.

Note that both equations are presented for the \( x \)-dimension, but the \( y \) and \( z \)-dimensions equations can be obtained by variable substitution between \( i \), \( j \), and \( k \).

### 5.4 3D Energy Minimization

In the following two subsections, we describe the algorithm to solve Equations 25 and 27. We point the reader to Piniés et al. (2015); Chambolle and Pock (2011); Rockafellar (1970); Handa et al. (2011) for a detailed derivation of these steps. We vary from their methods in our new definition for the gradient and divergence operators, as described in Section 5.3.

#### 5.4.1 SDF Optimization Implementation

The optimization described in Equation 25 is a 3D version of the Primal-Dual TV algorithm explained in Section 3. We just need to use the new definitions of the gradient and the divergence operator defined previously.

#### 5.4.2 Histogram Optimization Implementation

Again, the implementation to minimize the histogram-based approach closely mirrors the implementation described Section 3. The only difference is that the primal update (Equation 13) becomes

\[
\begin{align*}
  u_k &= \text{median}(c_1, ..., c_{n_{bins}}, b_0, ..., b_{n_{bins}}) \\
  \bar{u} &= u_k - \tau \nabla \cdot p \\
  b_i &= \bar{u} + \tau \lambda W_i \\
  W_i &= -\frac{1}{n_{bins}} \sum_{j=1}^{n_{bins}} h_j + \sum_{j=i+1}^{n_{bins}} h_j \\
  i &\in [0, n_{bins}]
\end{align*}
\tag{31}
\]

where, \( u_k \) is the primal solution at the \( k \)th optimization iteration, and \( W_i \) and \( b_i \) are the the optimal weight and gradient-ascent solution for the \( i \)th histogram bin.

For both the SDF and histogram optimization implementations, the operations in each voxel are independent. Therefore, our implementations leverage massively-parallel GPU processing with careful synchronization between subsequent primal and dual variable updates.

### 5.5 Data Structures

To summarize the key data tracked during the data fusion and 3D regularization stages our pipeline, we define the following three voxel datatypes:

```c
struct VoxelSdf {
  float sdf;
  uint8_t weight;
  uint8_t color[3];
  bool is_omega;
};

struct VoxelHistogram {
  uint8_t histogram[N_BINS];
  uint8_t color[3];
  bool is_omega;
};

struct VoxelTotalVariation {
  float u; // primal
  float p[3]; // dual
  float u_relax; // relaxation step
};
```

Sensor data (stereo-camera-based depth maps or laser) is continually fused into either \texttt{VoxelSdf} or \texttt{VoxelHistogram} voxel grid. In practice, we found the laser data works best with \texttt{VoxelSdf} and the camera-based depth maps work well (in the tested scenarios) with either \texttt{VoxelSdf} or \texttt{VoxelHistogram}. Section 7 provides a more detailed analysis on why this is the case.

Depth-map fusion color is read directly from the RGB reference image. When fusing lidar scans, the color is the grayscale reflectance intensity value. If both lidar and depth images are fused into the same voxel grid, color data always takes priority over grayscale data as to make the final model visually appealing (i.e., surfaces viewed by the camera are colored correctly while surfaces viewed only by the laser are grayscale).

Once all data is fused, the energy minimization data is stored in a voxel grid of type \texttt{VoxelTotalVariation}. The final surface may be extracted directly from the \texttt{VoxelTotalVariation} voxel grid via Marching Cubes (Lorensen and Cline 1987) or Ray Casting (Amanatides et al. 1987) to solve for the zero-crossing level set of \( u \).

### 6 Learnt Corrections and Prior Application

The final step in our pipeline is a system that uses prior knowledge of scene appearance and geometry to detect and correct missing data in the 3D reconstructions (e.g., holes in the road). To achieve this, we leverage a deep neural network. The subsystem described in this section is illustrated in Figure 10 and operates as follows:
Figure 10. Machine Learning Data-Flow Pipeline. To train our network, we create separate stereo-camera and laser dense reconstructions, using the techniques discussed in Sections 3–5. We create ground-truth data by subtracting the rasterized inverse-depth images from each reconstruction. The neural network trains on rasterized feature images (see Figure 11) to learn to generalize the ground-truth error in new scenes.

(a) RGB Image  (b) Inverse-depth
(c) Triangle Area  (d) Triangle Surface Normal
(e) Triangle Edge Length Ratios  (f) Surface to Camera Angle

Figure 11. Example Input Features. As our dense reconstructions provide a 3D model of our operating environment, we can extract a variety of features. Pictured above are an example frame from KITTI-VO (a), and the corresponding features our GLSL pipeline currently extracts (b)–(f). The network has the potential to use these low level features in its intermediate representation.

1. Generate a sequence of mesh features (including inverse-depth maps) from an existing reconstruction;
2. Use a CNN trained on high-quality reconstructions to predict the residual error in inverse-depth;
3. Correct the initial inverse-depth maps by subtracting the predicted errors;
4. Create a new reconstruction using the other modules described in this paper.

6.1 Mesh Features

Many works in the deep learning literature deal with RGB images and thus operate in 2D. However, as our problem formulation assumes that a dense 3D model is available, we can extract a much richer set of features from the reconstruction. Our system builds on the method proposed by Tanner et al. (2018), where mesh features are projected into sequences of 2D images. This method enables the use of established CNN architectures, while still leveraging the geometric information that meshes contain.

6.1.1 Feature Creation To extract dense 2D features suitable for a CNN, we begin by using GLSL to “fly” a virtual camera through the 3D models. Our vertex shader transforms the 3D model’s vertices into the camera reference frame to compute the depth of each vertex and the normalised camera vector. These camera-frame points are then grouped into triangles and passed along to a geometry shader that computes the triangle’s area, surface normal, and edge lengths. Finally, the fragment shader processes each of the preceding feature values and rasterises them into individual pixels in feature images, as shown in Figure 11. This GLSL...
The architecture we employ in this paper, like the one proposed by Tanner et al. (2018), is based on the Fully Convolutional Residual Network (Laina et al. 2016) with concatenated ReLU activation functions on the intermediate layers (Shang et al. 2016). This network was designed to infer per-pixel depth from a single RGB image, a task highly correlated with the aim to compute estimated depth error given a set of feature inputs. In this work, the input layer is generalized to accept $F$ input channels dependant on the number of active features used from the previous section. A notable addition is the U-Net-style (Ronneberger et al. 2015) skip connections between the encoder and the decoder. Table 3 gives a summary of the architecture. The encoder uses a series of residual blocks based on the ResNet-50 architecture (He et al. 2016), and the decoder uses the up-projection blocks proposed by Laina et al. (2016). The outputs of the residual blocks are padded and concatenated with the inputs to the up-projection blocks of corresponding scale. This architecture allows for high-frequency information (such as edges), to be more easily localized, since it is not compressed all the way through the encoder. The final layer is a simple $3 \times 3$ convolutional layer which outputs real-valued estimates corresponding to the inverse-depth error at each pixel location.

Table 3. Overview of the CNN architecture for error prediction.

<table>
<thead>
<tr>
<th>Block Type</th>
<th>Filter Size/Stride</th>
<th>Output Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>-</td>
<td>$150 \times 590 \times F$</td>
</tr>
<tr>
<td>Convolution</td>
<td>7x7/2</td>
<td>74 $\times$ 295 $\times$ 32</td>
</tr>
<tr>
<td>Max Pool</td>
<td>3x3/2</td>
<td>38 $\times$ 148 $\times$ 32</td>
</tr>
<tr>
<td>Convolution</td>
<td>3x3/1</td>
<td>38 $\times$ 148 $\times$ 64</td>
</tr>
<tr>
<td>Residual, Residual, Projection</td>
<td>3x3/2</td>
<td>19 $\times$ 74 $\times$ 256</td>
</tr>
<tr>
<td>Residual, Residual, Projection</td>
<td>3x3/2</td>
<td>10 $\times$ 37 $\times$ 512</td>
</tr>
<tr>
<td>Residual, Residual, Projection</td>
<td>3x3/2</td>
<td>5 $\times$ 19 $\times$ 1024</td>
</tr>
<tr>
<td>Residual, Residual</td>
<td>3x3/1</td>
<td>5 $\times$ 19 $\times$ 1024</td>
</tr>
<tr>
<td>Up-projection</td>
<td>3x3/3/2</td>
<td>10 $\times$ 38 $\times$ 512</td>
</tr>
<tr>
<td>Up-projection</td>
<td>3x3/3/2</td>
<td>20 $\times$ 76 $\times$ 256</td>
</tr>
<tr>
<td>Up-projection</td>
<td>3x3/3/2</td>
<td>40 $\times$ 152 $\times$ 128</td>
</tr>
<tr>
<td>Up-projection</td>
<td>3x3/3/2</td>
<td>80 $\times$ 304 $\times$ 64</td>
</tr>
<tr>
<td>Up-projection</td>
<td>3x3/3/2</td>
<td>160 $\times$ 608 $\times$ 32</td>
</tr>
<tr>
<td>Convolution, Resize</td>
<td>3x3/1</td>
<td>150 $\times$ 590 $\times$ 1</td>
</tr>
</tbody>
</table>

where $d_{lq}$ and $d_{hq}$ are pixels in the depth-map features for the low-quality and high-quality reconstruction, respectively. An illustration of this is shown in Figure 12c.

6.2 Reconstruction Error Prediction

6.2.1 Network Architecture

An illustration of the edge-based loss weighting used to train our neural network. When learning to predict error for a scene (a,b), edges are extracted from the ground-truth error label (c) using the Canny edge detector, and a weighting based on the distance transform is computed (d). Brighter areas represent higher weights. The per-pixel losses (berHu, bilateral smoothness regularization) are scaled by this weight, increasing penalty especially around sharp edges. The black areas correspond to a weight of 0, where ground-truth data is missing. Note the ground-truth error (c) is signed: blue represents negative values (missing parts) and red represents positive values (extra parts).
6.2.2 Generalisation Capacity To be able to deploy a learnt CNN to new meshes created without a high-fidelity sensor, we need our network to generalize to unseen data. To this end, three techniques are employed: cropping, downsampling, and weight regularization. Firstly, we randomly perturb and crop all input feature images before providing them to the network. After each epoch of training (i.e. the network has viewed all the training images), the next epoch will receive a slightly different cropped region of each input image. This prevents the network from associating a specific pixel location in the training data with its corresponding output.

Secondly, the feature maps are gradually downsampled via projection blocks (from ResNet-50) creating a bottleneck, thus reducing representational capacity. This also provides greater context to the convolutional filters in the later layers of the network by increasing the size of their receptive field.

Thirdly, we implement an $L_2$ weight regularizer to further constrain the representational capacity by preventing the network from over-relying on the cost function at the expense of generalization performance.

6.2.3 Loss Function Several loss functions are widely used in machine learning applications. The $L_2$ norm is traditionally popular because it heavily penalises large errors and is smooth. However, unlike the $L_1$ norm, it has a near zero gradient for small errors, thus is often unable to drive the error completely to zero. Over the years, researchers have proposed alternative norms which combine the “best” characteristics (based on application) of each of these norms. The Huber norm uses $L_1$ near the origin and $L_2$ elsewhere (Owen 2007). We choose berHu as our cost function on the networks error prediction since the $L_2$ norm is more favourable to capture high gradient edges around objects (Newcombe et al. 2011b). Furthermore, inspired by the work of Ronneberger et al. (2015) on U-Nets, we use a loss-weighting mechanism based on the Euclidean Distance Transform (Felzenszwalb and Huttenlocher 2004) to give more importance to edge pixels when regressing to the error in depth (Figure 12). We first extract Canny edges (Canny 1986) from the ground-truth labels. Based on these edges, we then compute the per-pixel weights as:

$$d^{ij} = \ln(1 + \text{EDT}(i, j))$$

$$w^{ij} = (w_{\text{max}} - w_{\text{min}}) \cdot \left(1 - \frac{d^{ij}}{\text{max}_{s, i} d^{ij}}\right) + w_{\text{min}},$$

where $w^{ij}$ is the loss weight for pixel $ij$, $\text{EDT}(i, j)$ is the Euclidean Distance Transform at pixel $ij$, and $w_{\text{max}}$ and $w_{\text{min}}$ are the desired range of the per-pixel weight. We use $w_{\text{min}} = 0.1$ and $w_{\text{max}} = 5$.

Finally, since the ground-truth reconstructions might be missing some data, we allow for a per-pixel mask on the loss, to ignore pixels with unavailable data.

The final loss function can be written as:

$$L = w_{\alpha} L_{\text{reg}} + \sum_{ij} w^{ij} \cdot (w_{\beta} L_{\text{berHu}}^{ij} + w_{\gamma} L_{\text{sm}}^{ij}),$$

where $L_{\text{reg}}$ is the weight regularization loss. We use $w_{\alpha} = 10^{-6}$, $w_{\beta} = 1$, $w_{\gamma} = 1$.

7 Results

This section provides an extensive analysis of our system, the parameters for which are provided in Table 2. Some

![Figure 13. GPS trace for the Oxford Broad Street dataset. The dataset includes data from 1x Bumblebee XB3, 1x Bumblebee2, 4x Grasshopper2, 2x Velodyne HDL-32E lidar, and 3x SICK LMS-151 lidar sensors.](image)
Table 6. SDF vs. Histogram Regularized Reconstruction Errors

<table>
<thead>
<tr>
<th>Fusion Method</th>
<th>Median</th>
<th>75%</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed Distance Function (SDF)</td>
<td>5.9 cm</td>
<td>21.0 cm</td>
<td>28.0 cm</td>
</tr>
<tr>
<td>Histogram</td>
<td>5.9 cm</td>
<td>21.5 cm</td>
<td>27.9 cm</td>
</tr>
</tbody>
</table>

parameters \( (\sigma, \theta, \tau) \) are standard constants from the literature (Pock and Chambolle 2011). The \( \mu \) and \( \lambda \) were empirically tuned. As discussed in Section 5.1.1, we found a constant \( \mu \) value based on the sensor accuracy produced better results across a range of scenarios. When using multiple sensor modalities, we selected \( \mu \) based on the least accurate sensor. We found \( \lambda \) to be dependent on the sensor observation density (number of observations per voxel) and accuracy. Selecting too large \( \lambda \) resulted in minimal changes to the quality of the reconstruction, but a very low \( \lambda \) deletes too many surfaces. If the same platform is used with a similar sensor suite and vehicle velocity between data collections, as is the case in the KITTI dataset, then the same \( \lambda \) is valid for all data collections.

In the first portion of our results, we present how our system’s reconstruction quality compares with prior work (Stanford Burghers of Calais and Imperial College ICL-NUIM). Next, we demonstrate its performance on our Oxford Broad Street dataset, that we release with this paper. Then, we use the publicly available KITTI dataset (Geiger et al. 2012) to evaluate the reconstruction performance of stereo-only, laser-only, and multi-sensor fusion methods. Finally, we use the KITTI dataset to present additional improvements that can be made to stereo-only reconstructions, when using a CNN trained on laser data.

Three KITTI sequences (00, 05, and 06) were selected based on their length, number of loop closures, and urban structure visible throughout the sequences. A summary of the physical scale of each is provided in Table 5.

The dense fusion and reconstruction experiments were performed on a GeForce GTX TITAN with 6 GB memory. The CNN mesh correction experiments were performed on a Tesla K80 with 12 GB memory.

Pose estimation was processed in real time (20 Hz); depth-map estimates (1 Hz), data fusion (5 Hz deep maps, 10 Hz Velodyne), and CNN post-processing (12 Hz) were at interactive rates; while the regularisation could only be achieved via an off-line process since it required multiple minutes to converge in each scenario (see Table 7). With these timing requirements, we use this reconstruction (depth
Table 7. Summary of Stereo-Camera-Only Reconstruction Quality and Error Statistics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Type</th>
<th>Res.</th>
<th>50%</th>
<th>75%</th>
<th>GPU Memory</th>
<th>Surface Area</th>
<th># Voxels</th>
<th>Time Per Iter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>KITTI-VO 00</td>
<td>Raw</td>
<td>20 cm</td>
<td>11.4 cm</td>
<td>38.3 cm</td>
<td>1,222 MiB</td>
<td>126,251 m²</td>
<td>107·10⁶</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>10 cm</td>
<td>10.7 cm</td>
<td>37.0 cm</td>
<td>7,565 MiB</td>
<td>126,251 m²</td>
<td>661·10⁶</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Regularized</td>
<td>20 cm</td>
<td>7.8 cm</td>
<td>25.9 cm</td>
<td>1,222 MiB</td>
<td>93,546 m²</td>
<td>107·10⁶</td>
<td>1:46 (mm:ss)</td>
</tr>
<tr>
<td></td>
<td>10 cm</td>
<td>7.3 cm</td>
<td>25.1 cm</td>
<td>7,565 MiB</td>
<td>93,546 m²</td>
<td>661·10⁶</td>
<td>11:20 (mm:ss)</td>
<td></td>
</tr>
<tr>
<td>KITTI-VO 05</td>
<td>Raw</td>
<td>20 cm</td>
<td>14.2 cm</td>
<td>45.0 cm</td>
<td>756 MiB</td>
<td>76,880 m²</td>
<td>66·10⁶</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>10 cm</td>
<td>12.7 cm</td>
<td>41.5 cm</td>
<td>4,533 MiB</td>
<td>76,880 m²</td>
<td>396·10⁶</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Regularized</td>
<td>20 cm</td>
<td>9.1 cm</td>
<td>30.2 cm</td>
<td>756 MiB</td>
<td>55,909 m²</td>
<td>66·10⁶</td>
<td>1:00 (mm:ss)</td>
</tr>
<tr>
<td></td>
<td>10 cm</td>
<td>8.1 cm</td>
<td>27.6 cm</td>
<td>4,533 MiB</td>
<td>55,909 m²</td>
<td>396·10⁶</td>
<td>3:45 (mm:ss)</td>
<td></td>
</tr>
<tr>
<td>KITTI-VO 06</td>
<td>Raw</td>
<td>20 cm</td>
<td>6.1 cm</td>
<td>23.8 cm</td>
<td>356 MiB</td>
<td>32,241 m²</td>
<td>31·10⁵</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>10 cm</td>
<td>5.5 cm</td>
<td>21.3 cm</td>
<td>2,591 MiB</td>
<td>32,241 m²</td>
<td>226·10⁶</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Regularized</td>
<td>20 cm</td>
<td>4.5 cm</td>
<td>12.4 cm</td>
<td>2,591 MiB</td>
<td>23,585 m²</td>
<td>31·10⁵</td>
<td>0:34 (mm:ss)</td>
</tr>
<tr>
<td></td>
<td>10 cm</td>
<td>4.0 cm</td>
<td>10.6 cm</td>
<td>356 MiB</td>
<td>23,585 m²</td>
<td>226·10⁶</td>
<td>3:44 (mm:ss)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 16. Comparison of reconstruction quality with stereo-camera only, laser only, and stereo-with-laser fusion. The stereo camera has a higher field of view than the laser sensor (i.e., the building/trees are cut half way up in the laser reconstruction), but the laser sensor is much more accurate and can see into regions which were occluded for the stereo camera (e.g., behind the automobiles). Fusing data from both sensors into the same voxel grid produces a more comprehensive (Table 8) and higher-quality result than either sensor can achieve alone.

Table 8. Summary of Surface Area Coverage by Sensor Type

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Image Only</th>
<th>Laser Only</th>
<th>Image &amp; Laser</th>
</tr>
</thead>
<tbody>
<tr>
<td>KITTI-VO 00</td>
<td>93,546 m²</td>
<td>117,087 m²</td>
<td>176,624 m²</td>
</tr>
<tr>
<td>KITTI-VO 05</td>
<td>55,909 m²</td>
<td>77,292 m²</td>
<td>111,188 m²</td>
</tr>
<tr>
<td>KITTI-VO 06</td>
<td>23,585 m²</td>
<td>29,617 m²</td>
<td>41,401 m²</td>
</tr>
</tbody>
</table>

7.1 Imperial College ICL-NUIM

We first evaluated the performance of our dense reconstruction pipeline (fusion and regularization) with the augmented Imperial College ICL-NUIM dataset Choi et al. (2015). The original dataset (Handa et al. 2014) provided a set of indoor scenarios with sequences of ground-truth 3D models, camera poses, and RGB-D images. In 2015, Choi et al. augmented the dataset by applying lens and depth distortion models to make the input more similar to that of real-world data.

A summary of our reconstruction results are presented in Table 4. The median reconstruction error of the raw fused model ranged from 4.35 cm to 10.10 cm. The regularizer reduced error by 24% to 29%, with final median errors between 3.17 cm and 7.64 cm. The absolute error metrics are not as important as the fact that the regularizer consistently reduced the reconstruction errors – a theme which repeats throughout the quantitative results results presented in this section.
Figure 17. A summary of the stereo-camera-only dense reconstruction quality for three scenarios from the KITTI-VO public benchmark dataset. The left side are the results before regularization and the right side are after regularization. Above each histogram of point-to-surface errors are the top-view, colored reconstruction errors corresponding to the same colors in the histogram. The regularizer reduces the reconstruction’s error by approximately a third, primarily by removing uncertain surfaces – as can be seen when you contrast the raw (left) and regularized (right) reconstruction errors.
Figure 18. A few representative sample images for various points of view (offset from the original camera’s position) along each trajectory. All sample images are of the final regularized reconstruction with 10 cm voxels using only stereo images as the input.
Figure 19. Comparison of reconstruction quality with laser only, stereo-camera only, and stereo-camera after CNN Correction. The correction step results in meshes with more coverage, though some areas (such as behind the parked cars) are occluded from the camera and therefore are not be corrected. A less obvious fix can be noticed in the road, where the uneven surface in (b), due to the shadows in the original images, is smoothed in (c).

7.2 Stanford Burghers of Calais

We first validate that our dense-fusion system creates high-quality reconstructions with the standard Stanford Burghers of Calais RGB-D dataset (Zhou and Koltun 2013). We performed a point-to-surface comparison between BOR$^2$G’s (0.5 cm voxels) and Zhou and Koltun’s (unknown voxel size) reconstructions. BOR$^2$G’s median reconstruction difference was 0.5 cm with a 1.0 cm 75-percentile difference. Note that this is a “difference” not an “error” metric as there is no ground-truth data against which we can compare.

As can be seen in Figure 2, our level of detail compares favorably with the reconstructions by Zhou and Koltun (2013). We accurately depict folds in clothing, muscle tone, and facial features.

However, this does not reveal the full capabilities of our framework as the low-noise, high-observation, and small-scale RGB-D dataset does not require sophisticated 2D and 3D regularization to create high-quality reconstructions. We are not aware of any other system which both operates at scale and accepts a variety of sensor inputs – hence the remaining quantitative analysis compares only against ground-truth data.

7.3 Oxford Broad Street Dataset

In practice, we found dense reconstructions are the most complete and highest quality (with our mobile robotics platform) when fusing data from multiple Velodyne, SICK LMS-151, and stereo cameras using our own datasets. We are releasing such a dataset with this paper to provide a realistic mobile-robotics platform with a variety of sensors – a few of which are prime candidates for ground truth when testing the accuracy of reconstructions with other sensors. For example, the push-broom laser sensor (which excels at 3D urban reconstructions) can be used to compare the reconstruction quality of monocular vs. stereo camera vs. Velodyne vs. a combination thereof.

This dataset is ideal to benchmark and evaluate large-scale dense reconstruction frameworks. It was collected in Oxford, UK at mid-day, thus it provides a representative urban environment with numerous pedestrians, bicycles, and vehicles visible to all sensors throughout the 1.6 km trajectory (Figure 13).

It includes data from the following sensors which collectively provide a continuous 360° view around the vehicle:

- 1x Point Grey Bumblebee XB3 Stereo Camera (Color)
- 1x Point Grey Bumblebee2 Stereo Camera (Grayscale)
- 4x Point Grey Grasshopper2 Monocular Cameras (Color, Fisheye Lens)
- 2x Velodyne HDL-32E 3D lidars
- 3x SICK LMS-151 2D lidars

In addition, the following is provided to aid in processing the raw sensor data:

- Optimized $\mathbb{SE}(3)$ vehicle trajectory
- Undistorted, rectified stereo image pairs
- Undistorted mono images
- Camera intrinsics
- Extrinsic $\mathbb{SE}(3)$ transforms for all sensors

Finally, we provide example depth maps – using the techniques described in this paper – for the Bumblebee XB3 to enable users to rapidly utilize the dataset with their existing dense reconstruction pipelines.

All data is stored in a similar format as KITTI along with MATLAB development toolkit. Videos of the
included data are available at http://ori.ox.ac.uk/dense-reconstruction-dataset/.

For brevity, we provide more detailed analysis of the stereo-only, laser-only, and multi-sensor fusion performance on the KITTI dataset in the following subsections. However, an example Velodyne-based reconstruction of a small segment of this dataset is shown in Figure 1 and a larger qualitative analysis is included in the Supplemental Material video.

### 7.4 KITTI Stereo-Only Reconstruction Analysis

In addition to traditional qualitative analysis, KITTI stereo-only reconstructions may be quantitatively analyzed by using laser data as ground truth. We consolidated all Velodyne HDL-64E laser scans into a single reference frame using poses provided by ORB-SLAM2. We found that KITTI’s “ground-truth” GPS/IMU poses were not accurate enough to produce high-quality dense reconstructions — in particular when revisiting regions. There was as much as 3 meters of drift from one pass of a region to the second pass with the KITTI ground-truth poses, while ORB-SLAM2 poses were accurate within a few centimeters.

The following three subsections perform analysis on the stereo-only reconstruction performance for SDF vs. Histogram data terms, comparing multiple stereo camera passes through the same environment, and the quantitative quality of 7.3 km of stereo-only reconstructions.

#### 7.4.1 SDF vs. Histogram Data Terms

Zach (2008) demonstrated the superior histogram data term performance in small-scale, object-centered scenarios. We compare SDF and histogram optimization performance so dense-reconstruction system developers may make an informed decision based on their target application.

After fusing stereo-based depth images into the voxel grid, the resulting reconstruction is indeed noisy, as shown in Figure 14. The 2D TGV² regularizer significantly reduced the noise in the input depth maps, but passively-generated depth maps inherently must infer depth of large portions of the image.

Both the SDF and Histogram optimizers smooth the noisy surfaces (e.g., buildings) in the reconstruction, but the SDF regularizer subjectively appears to do a better job. When compared against ground-truth in Table 6, the SDF and histogram optimizers are nearly indistinguishable: both reconstructions have a 5.9 cm median point-to-surface error with a 28 cm standard deviation.

Based on Zach’s previous results, we initially expected the histogram to perform better. However, it appears that when a camera travels through the environment (rather than observing a single object from many angles) there are not enough observations of surfaces to provide a complete PDF in each voxel. Since the SDF regularizer is both simpler to implement, faster to execute, and provides similar results, we believe it can be a better choice for mobile-robotics platforms and we use it exclusively for the remainder of our experiments.

#### 7.4.2 Multiple Passes

In theory, revisiting a region should result in a more complete and accurate reconstruction, assuming one has accurate localization and loop closures. Noisy depth maps from the stereo camera preclude the traditional point-cloud alignment approaches used in Kinect Fusion-based approaches. However, ORB-SLAM2 provides accurate loop closures and locally-consistent pose estimates.

In fact, ORB-SLAM2 performs better in our applications than did the GPS/INS ground truth – the latter resulted in trees and walls being observed in the middle of a road when revisiting a location.

In Figure 15, we compare the quality of reconstruction between a single and two passes of a region. The second pass noticeably improves the detail in previously-observed surfaces and increases the surface area reconstructed by 11% (6,044 m² → 6,713 m²).

#### 7.4.3 Full-Length KITTI-VO Error Metrics

Using only the stereo camera as input, we first processed all three KITTI-VO with 10 cm voxels and compared the dense reconstruction model, both before and after regularization, to the laser scans. In these large-scale reconstructions, the compressed voxel grid structure provides efficient fusion performance while vastly increasing the size of reconstructions. For the same amount of GPU memory, the conventional voxel grid was only able to process 205 m, in stark contrast to the 1.6 km reconstructed with the HVG – though that may be extended to an infinite-size reconstruction by utilizing bidirectional GPU-CPU streaming.

As shown in Table 7, the SDF regularizer reduced the median error by 27% to 36%, the 75-percentile error by 32% to 50%, and the surface area by 26%. The final median error for the scenarios varied between 4.0 cm and 8.1 cm.

In Figure 17, it becomes clear that errors in the initial “raw” fusion largely come from false surfaces created in the input depth map caused by sharp discontinuities in the original image. The SDF regularizer removes many of these surfaces, which dominate the tail of the histogram plots and are visible as red points in the point-cloud plots.

When processed at 20 cm voxel resolution, the results are similar, though with slightly higher error metrics, as shown in Table 7. However, even though the spatial resolution was reduced by a factor of two (and memory requirements by a factor of 8), the final reconstruction accuracy was only reduced by about 10%. This is because the urban environments are largely dominated by planar objects (e.g., roads, building façades) which, by the very nature of SDF, can be nearly-perfectly reconstructed with coarse voxels. Smaller voxels are only beneficial in regions with fine detail (e.g., automobiles, steps, curb).

Figure 18 shows the bird’s-eye view of each sequence with representative snapshots of the 10 cm stereo-camera-only reconstructions. To illustrate the quality, we selected several snapshots from camera viewpoints offset in both translation and rotation to the original stereo camera position, thereby providing a depiction of the 3D structure. Overall, the reconstructions are quite visually appealing; however, some artifacts such as holes are persistent in regions with poor texture or with large changes in illumination. This is an expected result since, in these cases, no depth map can be accurately inferred.
7.5 KITTI Multi-Sensor Reconstruction Analysis

Many mobile platforms have a variety of different sensors, yet conventionally much dense reconstruction work is isolated to a single sensor, and almost exclusively a camera of some sort (RGB-D, monocular, or stereo). The KITTI dataset provides both camera images and Velodyne laser scans, so we decided to see what quality of reconstructions could be achieved by fusing data from both sensors.

A snapshot of our qualitative results are shown in Figure 16, but the supplemental video provides a fly-through of each sequence to visualize the quality of our final regularized 3D reconstructions.

The stereo-camera-only reconstructions were moderately detailed, but a number of false surfaces were created between neighbouring objects (e.g., automobiles, buildings) due to the limitations of stereo-based depth maps. The camera reconstructions included the full height of the buildings, in contrast to the laser reconstructions which could only reconstruct the bottom 2.5 m of objects. However, the laser-only reconstruction was much more detailed and could see surfaces which were occluded to the camera (e.g., behind automobiles) because the Velodyne was mounted higher on the data-collection vehicle and it has a continuous 360° field of view.

Figure 16c shows the combination of both sensors produced a reconstruction that was higher quality and more comprehensive. The comprehensiveness is measured in Table 8, where the multi-sensor fused reconstruction has 40% to 50% more surface area coverage than the best sensor was able to reconstruct by itself.

7.6 Mesh Correction with CNNs

Correcting meshes post-hoc is particularly useful when dealing with reconstructions obtained in mobile robotics scenarios, where the amount of data is small relative to the surface area of the reconstruction.

In our experiments, we used a regularized laser reconstruction with 20 cm$^3$ voxels as our high-quality prior, along with 20 cm$^3$ regularized stereo-camera-only reconstructions to train the CNN described in Section 6. We generate mesh features along the original trajectories from each of the three KITTI sequences (00, 05, and 06), and train three models, one on each pair of sequences. When evaluating our approach on a sequence, we use the model trained on the other two sequences.

The network module was implemented in Python using the TensorFlow framework. The network weights were optimized using the ADAM solver (Kingma and Ba 2015) with a learning rate of 10$^{-4}$ for 300 epochs over two of the sequences. At this point, the value of the loss function appears converged with variance coming from the stochasticity of the mini-batch sampling. To reduce this variance and confirm convergence, the optimization is run a further 100 epochs with the learning rate reduced to 10$^{-5}$. All training parameters are documented in Table 9.

The reconstructions obtained by fusing laser and stereo-camera data are the most complete we can obtain given the sensor setup. As shown in Table 10, our learnt correction closes the coverage gap between the stereo-camera reconstruction and the combined reconstruction by 18–36%, without increasing the metric error.

7.7 EURofusion Joint European Torus Reconstruction

We also demonstrate a very different practical application of our work at the Joint European Torus (JET), which is operated by the UK Atomic Energy Authority (UKAEA) under contract from the European Commission, and exploited by the EURofusion consortium of European Fusion Laboritories.

Together with the UKAEA remote handling division, RACE (Remote Applications in Challenging Environments), an inspection was carried out inside the JET device using the remote handling system (Figure 20), in order to collect lidar and camera data. Our inspection platform (the “NABU”) is equipped with two push-broom lidars that are used in the

![Figure 20. Schematic view of the Joint European Torus (JET) fusion device, showing the Remote Handling systems (Boom and Mascot) deployed inside the torus. Image: EFDA-JET.](image-url)
reconstruction (2.5 cm voxels), and a stereo camera used for visual odometry. We ran our pipeline on some of the data obtained, and Figure 21 shows the reconstruction. The recovered surface has a 287 m² area.

The inside of the JET reactor vessel contains many reflective surfaces, which induce considerable noise in the laser measurements. This kind of scenario is particularly challenging, and benefits from our fusion and regularization components.

8 Conclusion

We presented our state-of-the-art BOR²G dense mapping system for large-scale dense reconstructions. When compared to object-centered reconstructions, mobile-robotics applications have 3,100 times fewer depth observations per square meter, thus we utilized regularizers in both 2D and 3D to serve as priors which improve the reconstruction quality. We overcame the primary technical challenge of regularizing voxel data in the compressed 3D structure by redefining the gradient and divergence operators to account for the additional boundary conditions introduced by the data structure. This both enables regularization and prevents the regularizer from erroneously extrapolating surface data.

We know of no other dense reconstruction system which is quantitatively evaluated at such a large scale. When qualitatively compared to the Stanford Burghers of Calais RGB-D dataset, BOR²G reconstructed the same fine detail. For our large-scale reconstructions, we used the 3D lidar sensor as ground truth to evaluate the quality of our reconstructions, for different granularities, against 7.3 km of stereo-camera-only reconstructions. Our regularizer consistently reduced the reconstruction error metrics by a third, for a median accuracy of 7 cm over 173,040 m² of reconstructed area. Qualitatively, fusing both stereo-camera-based depth images with lidar data produces reconstructions which are both higher quality and more comprehensive than either sensor achieves independently.

We demonstrated our system on a practical inspection task by producing a reconstruction of the EUROfusion Joint European Torus.

We showed how a CNN can be trained on historical high-fidelity data to learn a prior that can correct gross errors in meshes, such as holes in the road or cars. We overcome the challenge of processing meshes with CNNs by rasterizing appearance and geometric features and applying 2D techniques. The learnt CNN prior closes the gap in coverage by 18–36% without increasing the metric error.

Finally, we hope the release of our ground-truth KITTI pointclouds, ground-truth KITTI 3D models, and the comprehensive Oxford Broad Street dataset will become helpful tools of comparison for the community.

Supplemental material
- Oxford Broad Street dataset: http://ori.ox.ac.uk/dense-reconstruction-dataset/
- Stanford, Oxford, and KITTI reconstructions video: https://youtu.be/QHrQUlUpwOs
- Ground truth trajectories and Velodyne pointclouds
- Final colored dense reconstruction models

Notes
1. https://www.euro-fusion.org/

Acknowledgements

The authors would like to acknowledge the support of the UKs Engineering and Physical Sciences Research Council (EPSRC) through the Centre for Doctoral Training in Autonomous Intelligent Machines and Systems (AIMS) Programme Grant EP/L015897/1, Programme Grant EP/M019918/1, and the Doctoral Training Award (DTA). Additionally, the donation from Nvidia of a Titan Xp GPU used in this work is also gratefully acknowledged.

Part of this work has been carried out within the framework of the Contract for the Operation of the JET Facilities and has received funding from the European Union’s Horizon 2020 research and
innovation programme. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References


