2016 IEEE International Conference on Robotics and Automation (ICRA)
Stockholm, Sweden, May 16-21, 2016

Fast Depth Edge Detection and Edge Based RGB-D SLAM

Laurie Bose[1] and Arthur Richards[2]
University of Bristol, UK

Abstract— **This paper presents a method of occluding depth edge-detection targeted towards RGB-D video streams and explores the use of these and other edge features in RGB-D SLAM. The proposed depth edge-detection approach uses prior information obtained from the previous RGB-D video frame to determine which areas of the current depth image are likely to contain edges due to image similarity. By limiting the search for edges to these areas a significant amount of computation time is saved compared to searching the entire image. Pixels belonging to both the depth and colour edges of an RGB-D image can be back projected using the depth component to form 3D point clouds of edge points. Registration between such edge point clouds is achieved using ICP and we present a real-time RGB-D SLAM system utilizing such back projected edge features. Experimental results are presented demonstrating the performance of both the proposed depth edge-detection and SLAM system using publicly available datasets.**

I. INTRODUCTION

Autonomous navigation and exploration using simultaneous localization and mapping (SLAM) is an area of great activity and interest in robotics. The capability of conducting such tasks with cameras and computer vision is particularly desirable due to camera sensors generally being low cost and in readily available. However the necessity of real-time performance along with many persistent issues such as camera blur, varying lighting levels, dynamic objects and potential lack of visual features in a scene make this a highly challenging area. RGB-D sensors, providing both visual and geometric depth information, have seen increasingly widespread use in the field of SLAM. The additional geometric information such sensors provide makes the problem of SLAM more tractable, and as such RGB-D SLAM is an area of increasing activity. This paper presents a fast depth edge-detection approach for RGB-D video outlined in Section III, exploiting both the temporal image similarity between video frames and the ordered structure of the images themselves.

The edges present in an image typically occupy a small fraction of total pixels present. However, in many cases this set of edge pixels may still retain much of the important structure and details of the original image. This ability to provide a compact yet descriptive representation of a scene makes edges a good candidate for use as features for real-time SLAM. In the case of RGB-D sensors, edges can be found in both the color and depth images. The pixels corresponding to these sets of edges can then be back-projected to form 3D point clouds of edge points.

[1]PhD Candidate , Department of Engineering, University Of Bristol lb7943@bristol.ac.uk
[2]Reader , Department of Engineering, University Of Bristol arthur.richards@bristol.ac.uk

Pair-wise registration between two sets of edge points can be achieved using the Iterative Closest Point (ICP) algorithm that is abundant throughout SLAM literature. This registration is generally far less computationally expensive than using ICP to register two entire RGB-D point clouds due to the vastly decreased number of points. Additionally when registering such edge point clouds, there are in general fewer incorrect local minimum which the ICP could potentially converge to. This has been shown to result in increased registration robustness and accuracy, given sufficient edges are present [1], [2]. We examine the application of such edge point cloud registration further in the edge-based SLAM system proposed in Section V.

II. RELATED WORK

Much work has been done in computer vision regarding the detection of different types of RGB image features and their utilization in SLAM. Many monocular camera based SLAM systems ([3], [4], [5]) utilize visual point features such as the well known SIFT [6] and SURF [7] feature descriptors. Other feature types such as edge-lets and lines have also been explored ([8], [9]) though to a lesser extent. Since the introduction of low cost RGB-D sensors such as the Microsoft Kinect, various approaches to RGB-D SLAM have been proposed. Many of these have made use of the ICP algorithm in some form [10], [11], [12]. One such approach is KinectFusion [13], which instead of attempting to detect and register specific feature types in the RGB-D images, used ICP to register the raw depth point clouds from the sensor against a dense 3D map of the environment. The demanding process of performing this registration in real-time however was only made possible with the use of a GPU. The operational volume of the original system was also limited due to the memory hungry representation of the environment. This problem was later addressed by the "Kintinuous" system created by Whelan et al. [14]. Another approach towards RGB-D SLAM was to maintain the use of visual point features detected in the RGB component, and use the additional depth information to back project them forming sparse point clouds [15], [16], [17]. These sparse point clouds could then be registered using ICP (enhanced with the feature information of each point). Other works investigated the use of certain features specific to the depth image. Planar surfaces are one such feature and many methods have been proposed to detect them efficiently [18], [19], [20], [21]. Such planar features are well suited for indoor targeted SLAM systems due to their abundance in man made environments. A number of SLAM systems using planar features alone or combined with other feature types have be demonstrated by [22], [23], [24]. Edges are another type of feature that can be found within depth images. These can take the form of edges due to sharp depth discontinuities caused by the occluding boundaries of physical objects or edges due to two differently orientated surfaces meeting such as two walls at right angles. However current work regarding the use of such features is more limited . Choi et al. [2] presented methods of edge detection in depth images. Occluding edges corresponding to local discontinuities in depth were found by examining each depth image pixel and its 8 local neighbours. Additionally high curvature edges in the depth image were detected by examining local surface normals, and the well known canny-edge-detector [25] was used to detect color edges in the RGB component. Each of these edge types was then investigated to see which provided the best trade off between tracking quality and performance in pair-wise registration. A SLAM system based on occluding edge features was then demonstrated, however due to high computational cost the reported implementation was not well suited for for real-time operation.

This paper presents a method of detecting occluding depth edges from an RGB-D video stream. The given approach uses information that was acquired when performing depth edge-detection on the previous depth image to determine in which areas of the current depth image edges are likely to occur. This prior information is used to avoid searching areas of a depth image in which edges are not likely to be present. It will be shown that on average this results in a large computational saving while still detecting the vast majority of edge pixels compared to performing an edge-detection search on the entire depth image. To the best of our knowledge, this work is the first to utilize information gathered from the previous RGB-D frame to accelerate depth edge-detection.

The proposed depth edge-detection approach is presented in Section III in two parts. A basic edge-detection method which operates upon an entire depth image in Section III-A and a modified version which only operates on specific areas of the depth image based on knowledge gathered from the previous depth image in Section III-B. Section IV discusses the use of RGB color edges. A SLAM system based on such edge features is presented in Section V. Finally experimental results and an evaluation of both the proposed edge-detection and SLAM system is presented in Section VI.

III. Occluding Depth Edges

The depth image component of an RGB-D video stream provides 3D geometric information of the scene in view. In such an image, physical objects occluding others further from the camera gives rise to discontinuities between the values of neighbouring pixels. Two types of edges arise from such discontinuities. Occluding edges whose pixels belong to geometric boundaries of foreground objects, and occluded edges whose pixels belong to those objects and surfaces being occluded. Note that occluded edges can be viewed as simply being the edges of "shadows" cast by occluding objects as illustrated in Figure 1.

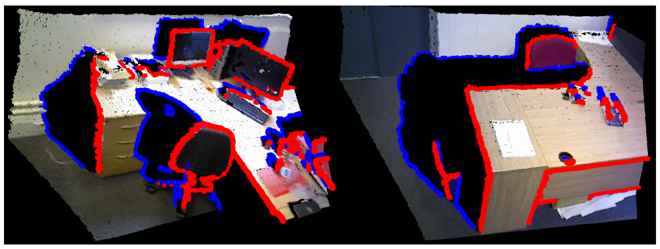
Fig. 1: Examples of occluding depth edges (red) and the related occluded edges (blue).

Algorithm 1 $P\_Scan(P, T)$
Detect large differences between the values of neighbouring pixels indicating the presence of occluding edges

```
INPUT:    $P = \{(V_n, L_n) : n \in \{0...N\}\}$
          // P is a list of depth pixels along a line
          // $V_n$ value of $n^{th}$ pixel
          // $L_n$ image location of $n^{th}$ pixel
          $T$   // edge detection threshold ratio value

OUTPUT:  $E = \{\}$   // edge pixel locations

$last\_valid = 0$   // index of the last observed valid pixel
for $n = 0$ to $N$ do
    if $V_n \neq 0$ then   // if current pixel value is valid
        $threshold = Min(V_n, V_{last\_valid}) \times T$
        // check for large differences between pixel values
        if $(V_{last\_valid} - V_n) > threshold$ then
            $E = E \cup L_n$
        else
            if $(V_n - V_{last\_valid}) > threshold$ then
                $E = E \cup L_{last\_valid}$
            end if
        end if
        $last\_valid = n$
    end if
end for
return $E$
```

As such they do not actually physically represent objects in the scene and are not useful for SLAM or camera tracking purposes. The following edge-detection process inherently detects both types of edges, however occluded edges are simply ignored.

A. Edge Detection by Row and Column Iteration

We employ a two step search process to find pixels belonging to occluding edges within a depth image. This involves examining the pixels of the depth image twice, first by rows then by columns. In either case an identical process is used to locate local depth discontinuities within a given line of pixels (be it a row or column). This pixel search process, outlined in Algorithm 1, iterates over a given list of pixels, skipping over invalid pixels (of zero depth value) while keeping track of the last found valid pixel. At each iteration the value of the current pixel (if valid) is compared to that of the last

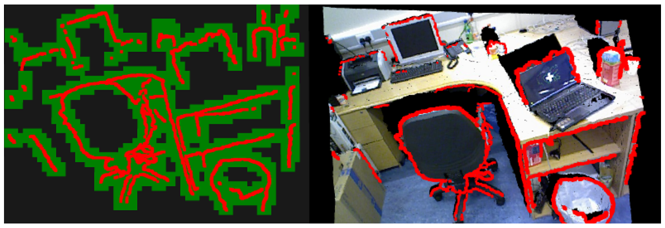
Fig. 2: Occluding depth edges (red) and the associated image patches which were flagged for edge detection (green).

found valid pixel. If the difference between these two values is above a certain threshold then this discontinuity in depth between nearby pixels is deemed large enough to indicate the presence of an occluding edge. If this is the case the pixel with the smaller of the two depth values (that being closer to the sensor) is identified as an occluding edge pixel. The other pixel which corresponds to an occluded edge is ignored. The prior mentioned threshold is given by $d \times T$ where d is the smaller of the two pixel values and T is a sensitivity constant. Having such a proportional threshold is necessary due to the nature of structured light RGB-D sensors where, both noise and spacing between readable values is proportional to depth [26], [27]. The two steps of this search process constitute scanning across the depth image in two orthogonal directions (by rows going left to right and by columns going top to bottom). In practice the organised image structure is exploited to minimize the computational time of this process.

B. Sub Image Edge Detection

When working with a standard RGB-D video stream (30hz 640×480), each depth image is likely highly similar to the previous one due to small camera movements between video frames. Occluding edge pixels are thus also likely to occur in similar locations from one depth image to the next due to this image similarity. This prior knowledge of where edge pixels are likely to occur can be exploited to significantly speed up the occluding edge detection process. This is done by only searching for edge pixels in specific areas of the depth image (around which edge pixels were previously detected) instead of simply searching the entire image in a brute force manner.

Our implementation of this concept considers each depth image as consisting of an $N \times M$ grid of smaller equally sized images (referred to as "image patches") rather than a single image. For example a grid size of 4×3 would consider a 640×480 depth image as consisting of 12 image patches of 160×160 pixels each. An $N \times M$ array of boolean flags F is also stored and used to determine which image patches should be searched for occluding edge pixels. With each new depth image from the RGB-D video stream, all image patches whose associated flag is set to true are searched using the same occluding edge-detection process described in Section III-A. Searched image patches in which no edge pixels are detected then have their associated flag reset to false. Conversely those image patches in which edge pixels were detected have both their own flag and those of their neighbouring patches set to true. By setting the values of the boolean array in this manner, the next image will only be searched for edges in those image patches located around where edge pixels were detected in the current image. While sufficient for detecting edge pixels belonging to edges that were also found in the previous image, this flagging scheme may fail to detect pixels belonging to new edges unique to the latest depth image (or edges that were simply not detected previously). To address this, with each new depth image a certain number edge-detection flags are randomly selected and set to true to facilitate the detection of new edges.

The number of flags randomly selected in this manner (R) is determined by Equation 1 below.

$$R = Max(1, Round(N \times M \times rand\_search)) \quad (1)$$

Here $N \times M$ is the number of image patches and $rand\_search \in [0, 1]$ is a specific value relating to what percent of each depth image we wish to be randomly selected and searched for edges. Thus at least one image patch is always randomly searched, increasing up to all image patches (and thus the entire image) being searched as $rand\_search$ approaches 1. In this scheme the detection of new edge pixels may be delayed a number of frames until the flag associated with the image patch they reside in is randomly selected and set to true. This potential detection delay is one of the trade-offs for the reduced computation this approach provides. Using higher values of $rand\_search \in [0, 1]$ will decrease the average number of frames of this detection delay but increase the average computational cost of the whole occluding edge detection process. The value of $rand\_search = 0.05$ was found to provide sufficiently small average detection delay and is used throughout the rest of this paper.

Algorithm 2 outlines this entire edge-detection process taking a set of image patches P, edge-detection flags F and number of patches to randomly search R as inputs and returning a list of edge pixel locations E and updated boolean flags F. This algorithm can also take an additional parameter K determining which rows and columns of each image patch should be skipped over instead of being searched for edge pixels. For greater clarity we will refer to this parameter as $rowcol\_skip$ from now on. The default value $rowcol\_skip = 1$ results in no rows/columns being skipped, with $rowcol\_skip = 2$ only every other row/column is searched, $rowcol\_skip = 3$ only every 3^{rd} row/column is searched and so on. This parameter can be viewed as allowing downscaled edge detection search to be conducted, resulting in fewer edge pixels being returned (and thus sparser edge point clouds being created from these pixels). Examples of this approach illustrating the image patches grid and corresponding edge-detection flag values are show in Figure 2. Detailed results of running this proposed occluding edge detection on a number of RGB-D sequences are given in Section III-A.

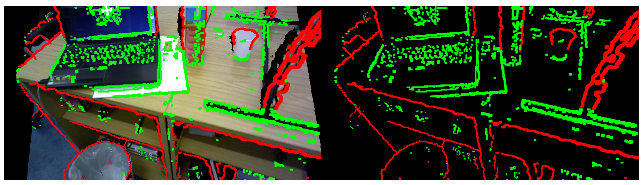
Fig. 3: RGB edges (green) and occluding edge (red).

Algorithm 2 *Occluding_Edge_Detection*(P, F, T, R, K) Detect occluding depth edge pixels within flagged image patches and update edge detection flags

```
// image patches $P$ and their associated boolean flags $F$
INPUT:   $P = \{P_{xy} : x \in \{0...N\}, y \in \{0...M\}\}$
         $F = \{F_{xy} : x \in \{0...N\}, y \in \{0...M\}\}$
         $T$   // edge detection threshold value
         $R$   // number of patches to randomly search
         $K$   // row and column skip value

OUTPUT:  $E = \{\}$   // edge pixel locations

Set $R$ randomly selected flags from $F$ to True
for all $P_{ij} \in P$ do
    if $F_{ij}$ then   // if patch flagged for edge detection
        $row\_edges = \{\}$
        $col\_edges = \{\}$
        for all $K^{th}$ rows $R$ of image patch $P_{ij}$ do
            $row\_edges = row\_edges \cup P\_Scan(R, T)$
        end for
        for all $K^{th}$ columns $C$ of image patch $P_{ij}$ do
            $col\_edges = col\_edges \cup P\_Scan(C, T)$
        end for
        if $row\_edges \neq \{\}$ or $col\_edges \neq \{\}$ then
            $E = E \cup row\_edges \cup col\_edges$
            $F_{ij} = True$
            Set flags of neighbouring image patches to True
        else
            $F_{ij} = False$   // reset patches flag
        end if
    end if
end for
return $E, F$
```

IV. RGB Edges

Each frame of RGB-D video provides not only a depth image but a corresponding RGB color image. Edges present in this RGB image can be back-projected and used for pair-wise registration and SLAM in the same way as the occluding depth edges of Section III. In practice the detection of such RGB edges presents a more challenging problem than that of occluding depth edge-detection. There are a number of RGB image specific issues such as image blur and variable lighting levels. Additionally RGB images will generally have a higher detail density compared their corresponding depth image making the identification of edges more complex.

Our employed method of RGB edge detection functions in a very similar way to the occluding depth edge-detection described in Section III-A, searching rows and columns of pixels from the RGB image for abrupt local changes in color values. Rows and columns can also similarly be skipped over as described by the use of the $rowcol\_skip$ parameter in the previous section. However it does not make use of any information from the previous RGB image, and simply searches all rows and columns of the image. We seek to improve or use an alternative RGB edge-detection method in the future work, however it is not the focus of this paper.

It is important to note that we reject all RGB edge pixels located in close proximity to occluding depth edges in the corresponding depth image. The back projected points from such pixels can often be highly unreliable due to poor alignment and synchronisation between the depth and RGB data provided by the RGB-D sensor. Additionally it is highly likely that such RGB edge pixels are the result of an occluding foreground object, in which case an occluding depth edge will already be present in the same location. The occluding depth edge detection flags described in Section III-B are used to enable fast rejection of such unwanted RGB edge pixels.

V. RGB-D Edge SLAM

The pixels belonging to the different types of edges in an RGB-D image can be back-projected to form 3D point clouds of edge points. Each of these clouds can be viewed as a down-sampling of the full RGB-D point cloud to only points expressing a specific type of information. Such edge point clouds are far smaller in size than the full cloud, but may still provide excellent information for localization. We created a SLAM system utilizing edge point clouds resulting from both the occluding depth edges discussed in Sections III and RGB edges discussed in Section IV. It was shown by [2] that these two edge types provided superior sensor odometry on Frieburg RGB-D dataset sequences ([28]) compared to a number of alternative features making them good candidates for use in SLAM. Additionally using two different types of edge features decreases the likelihood of localization being lost due to insufficient features being present (there will exist scenes in which occluding edges are not present but RGB edges are plentiful, and vice versa). This increases both the SLAM system's robustness and the range of environments in which it can successfully operate.

The standard pose graph set-up was used, in which the constructed map consists of a set of key-frames $K = \{k_0, k_1, ..., k_N\}$ with estimated poses $p_0, p_1, ..., p_N$. Each of these key-frames contains various features and other information. The current pose of the sensor is then tracked relative to the map by registering the latest observed features against those of the key-frames. For our system specifically each key-frame $k_i = (O_i, C_i, T_i^O, T_i^C)$ consists of a point cloud of back-projected occluding edge pixels O_i and second cloud of color edge pixels C_i. Additionally each key-frame stores two KD-trees T_i^O, T_i^C constructed from these point clouds used for point cloud registration.

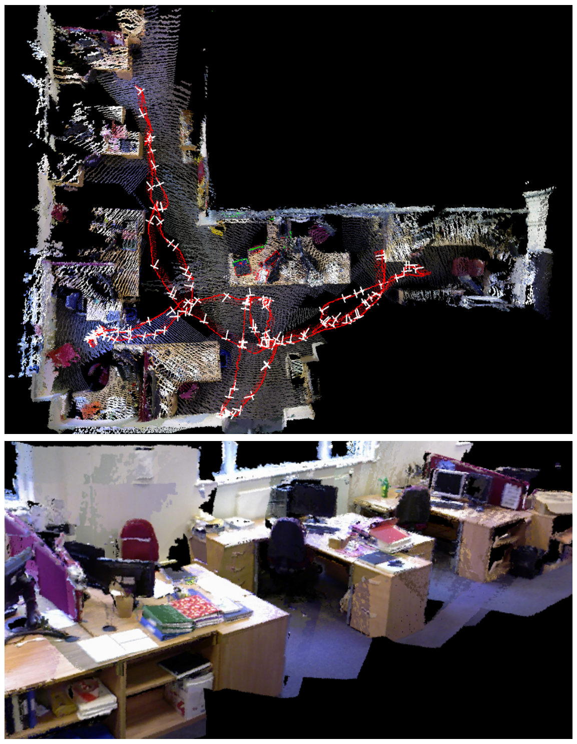
Fig. 4: An example map created by the proposed SLAM system. Key-frames poses are drawn in white and estimated camera trajectory (57m) in red.

A. Registration And Sensor Tracking

An ICP registration process denoted $reg(C^t, C^s, T^s, p^i)$ is used to estimate the transformation to best align a set of N target point clouds $C^t = \{c^t_0, ...c^t_N\}$ with a set of N source point clouds $C^s = \{c^s_0, ...c^s_N\}$. The input set of KD-trees $T^s = \{t^s_0, ...t^s_N\}$ created from the C_s clouds is used to greatly speed up nearest neighbour searches required by this process. The input transformation p^i provides an initial guess for the alignment transformation.

To track the pose of the sensor relative to the map (p^{sensor}), occluding and color edges are extracted from each new RGB-D image. The pixels of these edges are then back projected to form occluding and color edge point clouds O_{new} and C_{new}. The pose of the sensor relative to a certain key-frame $k_i = (O_i, C_i, T^O_i, T^C_i)$ can then be estimated by registering the point clouds O_{new}, C_{new} with O_i, C_i using reg as shown in Equation 2. The initial guess of best registration pose is taken to be p^{rel}, the relative pose between the sensors current estimated pose p^{sensor} and the key-frames current pose p_i.

$$reg(\{O_{new}, C_{new}\}, \{O_i, C_i\}, \{T^O_i, T^C_i\}, p^{rel}) \qquad (2)$$

This ICP process uses both occluding and color edges to estimate the registration transformation. Naturally at each ICP iteration, when determining pairs of associated points between the two sets of clouds, occluding edge points are only paired with other occluding edge points (and similarly for color edge points).

In practise only one key-frame is used for sensor tracking at any point in time. Whenever the registration between the sensor and the tracking key-frame is not strong enough, other key-frames near the sensors current estimated pose are tested to see if they provide a stronger registration. If another key-frame is found to provide sufficiently strong registration then this takes over as the tracking key-frame. Thus the map key-frame used for sensor tracking is automatically changed as the sensor moves about the world. In the case where no key-frame is found to provide sufficient registration, a new key-frame is added to the map located at the sensors current estimated pose.

B. Loop Closure Detection And Pose Graph Optimization

Map optimization and loop closure is performed simultaneously on a separate CPU core to that performing sensor tracking. This uses the standard pose graph concept wherein relative pose constraints are added between pairs of key-frames poses. Each time a new key-frame is added to the map, ICP registrations are attempted between this new key-frame and each of the previous key-frames. The initial transformation used for such a registration between a new key-frame k_{new} and a previously existing key-frame k_i, is taken to be p^{rel} the relative pose between the current key-frame poses p^{new} and p^i. Each of these registrations is evaluated and assigned a score based upon the strength of the registration (i.e. what proportion of points were paired between the edge point clouds of the two key-frames by the ICP) and the extent to which the final registration transformation differs from the the current relative pose between two key-frames p^{rel}. This is then to determine if a registration should be used to create a new loop closure constraint. Thus strong registrations or registrations whose reported transformations is similar to p^{rel} are used to add new loop closure constraints to the pose graph. Conversely weak registrations whose reported transformation differs greatly from p^{rel} (which are likely caused by the ICP converging to an incorrect local minima or no overlap existing between the point clouds) are rejected as constraints.

Simultaneously pose graph optimization is performed to try and determine the set of poses most compliant with the given loop closure constraints. We implemented a mesh relaxation based method very similar to that presented by Andrew Howard et al. [29] but extended to 6DOF poses to conduct this optimization.

VI. Results

This section presents a sample of results from various experiments conducted to evaluate the performance of both the edge-detection proposed in Section III and edge based SLAM system described in Section V. We use the publicly available Freiburg RGB-D datasets [28] to conduct this evaluation. These consist of a number of 640×480 RGB-D video sequences of various environments along with sensor ground truth trajectories obtained from motion capture. The results were obtained from an Intel i5 CPU, 4GB RAM laptop running Ubuntu 14.10.

TABLE I: Occluding depth edge detection results from Freiburg sequences using a variety of image patch grid sizes

	FR1 desk			FR1 plant			FR1 room		
Grid Size	**Occluding edges**	**Pixel %**	**avg % img searched**	**Occluding edges**	**Pixel %**	**avg % img searched**	**Occluding edges**	**Pixel %**	**avg % img searched**
Whole Image	8.9±2.25 ms	100	100	9.83±2.50 ms	100	100	10.91±1.97 ms	100	100
4x3	8.31±2.57 ms	99.7	96.65	8.9±2.07 ms	99.8	98.18	8.75±2.03 ms	99.8	98.98
8x6	7.65±2.25 ms	99.1	92.13	8.69±2.31 ms	99.5	95.80	8.15±2.11 ms	99.2	93.74
16x12	6.74±2.26 ms	97.8	74.85	7.92±2.02 ms	98.9	86.92	7.98±2.22 ms	98.4	78.11
32x24	4.91±1.95 ms	96.0	52.29	6.41±1.92 ms	97.6	66.67	5.11±1.68 ms	96.6	55.34
64x48	3.13±1.32 ms	91.8	34.28	5.35±1.79 ms	96.3	51.49	4.08±1.7 ms	90.7	36.43
128x96	2.41±1.04 ms	80.7	22.38	4.09±1.44 ms	84.5	30.62	2.56±1.03 ms	77.2	23.92

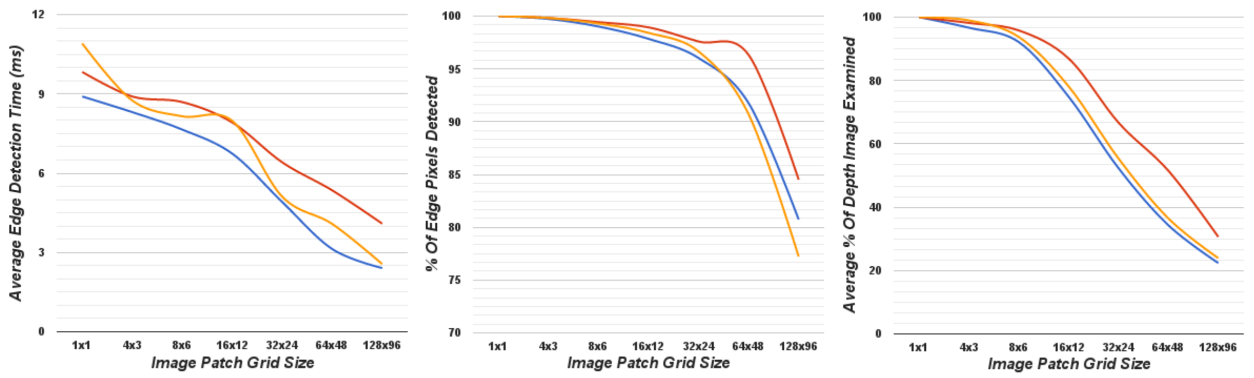

Fig. 5: Plots showing how various performances metrics of the occluding edge detection vary with image patch grid size and RGB-D sequence, FR1 desk (Blue), FR1 plant (Red) and FR1 room (Yellow).

TABLE II: Comparison of Occluding edge detection methods

Sequence	**Choi et al[2]**	**Proposed** (32×24)	**Proposed** (16×12)
FR1 desk	24.06 ± 1.22 ms	4.91 ± 1.95 ms	6.74 ± 2.26 ms
FR1 desk2	24.71 ± 0.79 ms	4.88 ± 1.91 ms	6.64 ± 1.93 ms
FR1 room	23.86 ± 1.47 ms	5.11 ± 1.68 ms	7.98 ± 2.22 ms
FR1 plant	24.61 ± 1.71 ms	6.41 ± 1.92 ms	7.92 ± 2.02 ms
FR1 rpy	23.89 ± 0.99 ms	5.35 ± 1.83 ms	6.18 ± 1.94 ms
FR1 xyz	24.45 ± 1.36 ms	4.16 ± 1.30 ms	5.19 ± 1.03 ms

A. Depth Edge Detection

We evaluated the occluding depth edge-detection described in Section III-B on a number of datasets and with various image patch grid sizes. In each case the total number of edge pixels detected across the entire sequence was recorded. This total was then compared to the total number of edge pixels obtained when using brute force occluding edge detection (searching the entirety of each depth image). From this comparison the percentage of total edge pixels detected (compared to brute force) was calculated. What percentage of each depth image examined for edge pixels was also recorded in each case.

A detailed sample of these results is given in Table I showing the average computation time of the edge detection process, percentage of total edge pixels detected, and what percentage of each depth image was examined on average. These results are illustrated in the plots of Figure 5. As expected the average computation time and percentage of depth image examined generally decreases as the image patch grid size increases. This is due to the smaller image patches flagged for edge-detection more tightly fitting about the edges detected in the previous image. This tighter fitting then results in edge-detection being performed upon a smaller percentage of the entire depth image. The down side of larger grid sizes however is an increased likelihood of edges failing to be re-detected from one image to the next due to changes in their locations between images (moving them into image patches not flagged for edge detection). This issue occurs most often when the sensor is undergoing a rapid change in orientation.

It can be seen however that using an image patch grid size such as 32x24 can provide up to a 50% saving in computation time while still detecting over 95% of the edge pixel in the sequence. The average computation time when using such a grid size is also well within the 33ms required to process a standard 640×480 30hz RGB-D video stream in real-time and leaves plenty frame time remaining in which other processes can take place.

Table II shows a comparison between the computation times of our proposed occluding edge detection and that introduced in [2] across a number of Freiburg sequences. The use of prior knowledge to only selectively search certain areas of the depth image gives the proposed method a far lower computation time compared method conducted by [2] which searches the entire image. All results in this section were obtained using the parameter value $rowcol\_skip = 1$.

TABLE III: Evaluation results of the SLAM system proposed in Section V on various RGB-D video sequences along with comparisons to other SLAM systems. Reported results were obtained with the edge detection parameter $rowcol\_skip = 5$.

	SIFT based RGB-D SLAM			Occluding edge based SLAM			Proposed RGB-D edge SLAM		
Sequence (length)	**Trans RMSE**	**Rot RMSE**	**Total Runtime**	**Trans RMSE**	**Rot RMSE**	**Total Runtime**	**Trans RMSE**	**Rot RMSE**	**Total Runtime**
FR1 desk (23 s)	**0.049 m**	**2.42 deg**	199 s	0.153 m	7.47 deg	65 s	0.075 m	3.43 deg	**14 s**
FR1 desk2 (25 s)	0.102 m	3.81 deg	176 s	0.115 m	5.87 deg	92 s	**0.098 m**	**3.75 deg**	**16 s**
FR1 plant (42 s)	0.142 m	6.34 deg	424 s	0.078 m	5.01 deg	187 s	**0.076 m**	**4.09 deg**	**29 s**
FR1 room (49 s)	0.219 m	9.04 deg	423 s	**0.198 m**	6.55 deg	172 s	0.210 m	**5.66 deg**	**30 s**
FR1 rpy (28 s)	**0.042 m**	**2.50 deg**	243 s	0.059 m	8.79 deg	95 s	0.055 m	4.20 deg	**16 s**
FR1 xyz (30 s)	**0.021 m**	**0.90 deg**	365 s	**0.021 m**	1.62 deg	111 s	0.038 m	1.92 deg	**17 s**

Fig. 6: Plots comparing results obtained by different SLAM systems on a number of Freiburg datasets. RGB-D SLAM [30] is shown in red, occluding edge RGB-D SLAM [2] in green, and our proposed edge based SLAM in Blue.

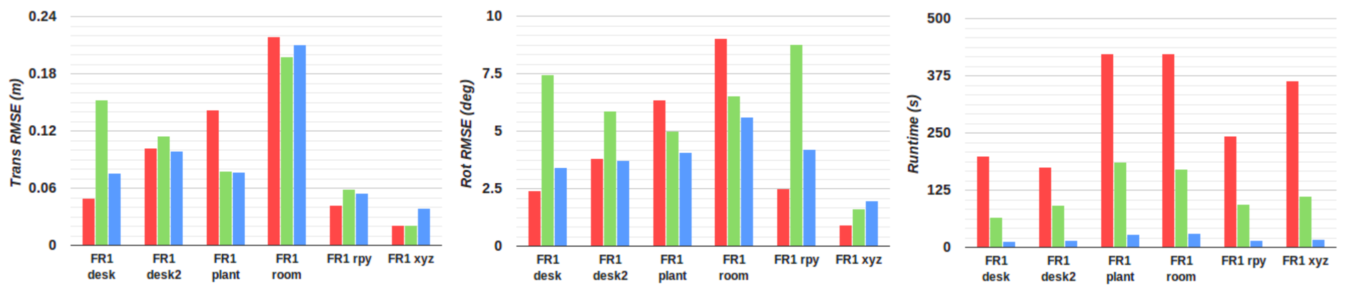

TABLE IV: Results detailing the effects of changing the $rowcol\_skip$ parameter on SLAM system performance (translational RMSE, rotational RMSE and total runtime).

Row Col Skip Value	**FR1 desk 23 s**	**FR1 desk2 25 s**	**FR1 plant 49s**	**FR1 rpy 28 s**
1	0.081 m 3.65 deg 79 s	0.101 m 3.69 deg 93 s	0.078 m 4.22 deg 149 s	0.061 m 4.39 deg 101 s
5	0.075 m 3.43 deg 14 s	0.098 3.75 deg 16 s	0.076 m 4.09 deg 29 s	0.055 m 4.20 deg 16 s
10	0.085 m 3.76 deg 7 s	0.111 m 3.88 deg 8s	0.098 m 5.19 deg 15 s	0.056 m 4.25 deg 8 s
20	0.103 m 4.13 deg 4 s	0.122 m 4.21 deg 5 s	0.098 m 4.10 deg 10 s	0.056 m 4.24 deg 4 s

B. RGB-D Edge SLAM

The same Freiburg datasets were again used to evaluate the performance and accuracy of the SLAM system laid out in Section V. The estimated sensor trajectory produced by the SLAM was compared to the provided ground-truth trajectory using the evaluation tools provided by [28]. All results were obtained using an image patch grid size of 32×24 as discussed in Section III-B.

Table IV gives an evaluation of the proposed system on a number of datasets and also details the effects of altering the row/column skipping parameter $rowcol\_skip$ given to the detection processes. Increasing the value of $rowcol\_skip$ results in fewer edge pixels being returned by the edge detection processes, and thus smaller down sampled edge point clouds being used by the SLAM system. We observed that the system's accuracy displayed a surprising level of robustness to this down sampling. With $rowcol\_skip = 10$, only every 10^{th} row and column of each image patch is searched for edge pixels. On average this results in 10 times fewer edge pixels being detected (and thus 10 times smaller edge point clouds) compared to when $rowcol\_skip = 1$. Despite using such smaller point clouds the resulting sensor trajectories are still comparable in accuracy to those obtained when using no down sampling. Using larger values of $rowcol\_skip$ also greatly decreased the total runtime on each data set due to the smaller edge point clouds resulting in much faster ICP registration. The results obtained using $rowcol\_skip = 1$ were in fact slightly less accurate to those of $rowcol\_skip = 5$, due to cases where the ICP registration did not fully converge within the the allowed 15 iterations. Because of this robustness to down sampling and the desire of real time performance, $rowcol\_skip = 5$ was chosen to be default for SLAM system use.

Table III shows a comparison between results from the proposed system and other RGB-D SLAM systems on the same Freiburg datasets. These systems are the well known SIFT feature based RGB-D SLAM [30] (running on a "quad-core CPU with 8 GB of memory"), and the occluding edge based SLAM system presented by [2] (running on a Intel Core i7 CPU, 8GB memory). The edge based SLAM system of [2] uses occluding depth edges in a similar manner to our own system but does not make use of RGB edges and utilizes a different method of occluding edge detection with higher run times (as illustrated Table II). We see that in general our system provides a comparable levels of accuracy while having far shorter total runtimes, being able to process the 30hz sequences in real-time.

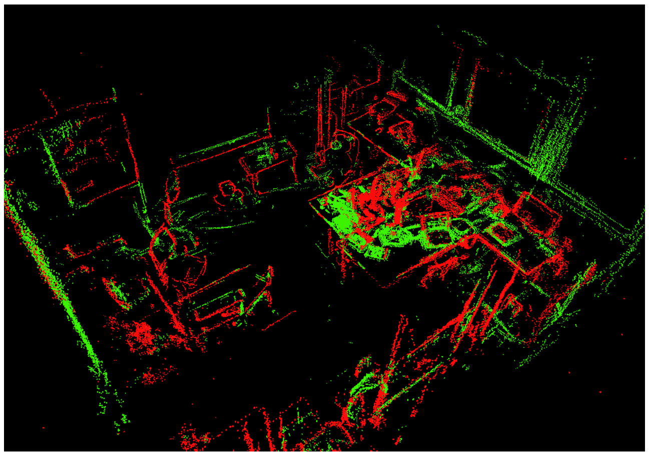

Fig. 7: Map created by the proposed SLAM system on the FR1 room dataset. Occulding edge point clouds are drawn in red and RGB edge points in green.

VII. Conclusions

This paper presented a fast, easy to implement method of occluding depth edge-detection for RGB-D video. The method makes uses of knowledge obtained from the previous depth image, to determine what areas of the current image should be searched. In doing so it is able to ignore large sections of the image where depth edges are unlikely to occur. Evaluating on public data sets demonstrated that such a method can have substantial computational savings over searching the entire image. An edge based RGB-D SLAM system using the proposed edge detection methods was also demonstrated, with similar accuracy and significantly lower computation times compared to other state of the art systems.

References

[1] Wei Li, Lei Zhang, David Zhang, and Jingqi Yan. Principal line based ICP alignment for palmprint verification. *Proceedings - International Conference on Image Processing, ICIP*, pages 1961–1964, 2009.

[2] Changhyun Choi, Alexander J B Trevor, and Henrik I. Christensen. RGB-D edge detection and edge-based registration. *IEEE International Conference on Intelligent Robots and Systems*, pages 1568–1575, 2013.

[3] Georg Klein and David Murray. Parallel Tracking and Mapping for Small AR Workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10, 2007.

[4] Andrew J. Davison. Real-time simultaneous localisation and mapping with a single camera. *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 2:1403–1410, 2003.

[5] Hauke Strasdat, J M M Montiel, and Andrew J Davison. Scale Drift-Aware Large Scale Monocular SLAM. *Robotics: Science and Systems*, 2:5, 2010.

[6] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[7] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3951 LNCS:404–417, 2006.

[8] a.P. Gee and W. Mayol-Cuevas. Real-time model-based slam using line segments. *Lecture Notes in Computer Science, vol. 4292*, 4292:354–363, 2006.

[9] E. D. Eade and T. W. Drummond. Edge Landmarks in Monocular SLAM. *Procedings of the British Machine Vision Conference 2006*, pages 2.1–2.10, 2006.

[10] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[11] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the ICP algorithm. *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on. IEEE*, pages 145–152, 2001.

[12] Aleksandr V Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-ICP. *Proc. of Robotics: Science and Systems*, 2:4, 2009.

[13] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. 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*, pages 559–568. ACM, 2011.

[14] Thomas Whelan, Michael Kaess, and Maurice Fallon. Kintinuous: Spatially extended kinectfusion. *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.

[15] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D Mapping : Using Depth Cameras for Dense 3D Modeling of Indoor Environments. *RGBD Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS*, 1(c):9–10, 2010.

[16] José Martınez-Carranza and Walterio Mayol-Cuevas. Real-time continuous 6d relocalisation for depth cameras. 2013.

[17] Benjamin Huhle, Timo Schairer, Philipp Jenke, and Wolfgang Straßer. Fusion of range and color images for denoising and resolution enhancement with a non-local filter. *Computer vision and image understanding*, 114(12):1336–1345, 2010.

[18] Kristiyan Georgiev, Ross T Creed, and Rolf Lakaemper. Fast plane extraction in 3d range data based on line segments. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3808–3815. IEEE, 2011.

[19] Junhao Xiao, Jianhua Zhang, Jianwei Zhang, Houxiang Zhang, and Hans Petter Hildre. Fast plane detection for SLAM from noisy range images in both structured and unstructured environments. *2011 IEEE International Conference on Mechatronics and Automation, ICMA 2011*, pages 1768–1773, 2011.

[20] Jann Poppinga, Narunas Vaskevicius, Andreas Birk, and Kaustubh Pathak. Fast plane detection and polygonalization in noisy 3D range images. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pages 3378–3383, 2008.

[21] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-time plane segmentation using rgb-d cameras. In *RoboCup 2011: Robot Soccer World Cup XV*, pages 306–317. Springer, 2012.

[22] Yuichi Taguchi, Yong Dian Jian, Srikumar Ramalingam, and Chen Feng. Point-plane SLAM for hand-held 3D sensors. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5182–5189, 2013.

[23] Jan Weingarten and Roland Siegwart. 3d slam using planar segments. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3062–3067. IEEE, 2006.

[24] Renato F Salas-Moreno, Ben Glocken, Paul HJ Kelly, and Andrew J Davison. Dense planar slam. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, pages 157–164. IEEE, 2014.

[25] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[26] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[27] Krystof Litomisky. Consumer rgb-d cameras and their applications. *Rapport technique, University of California*, page 20, 2012.

[28] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 573–580. IEEE, 2012.

[29] Andrew Howard, Maja J Matarić, and Gaurav Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 2, pages 1055–1060. IEEE, 2001.

[30] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the rgb-d slam system. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1691–1696. IEEE, 2012.