# Real-Time Outdoor Illumination Estimation for Camera Tracking in Indoor Environments

Michael Krawez[1], Tim Caselitz[1], Jugesh Sundram[2], Mark Van Loock[2], and Wolfram Burgard[1]

***Abstract*—Dynamic illumination is a challenging problem for visual robot localization and tracking. In indoor environments, the main source of light during the day is outdoor illumination. We propose a method that estimates the appearance of an indoor scene in real-time based on a reflectance map and the current outdoor lighting. Our outdoor illumination model consists of three components, namely sun, sky and ground, where the sun position is obtained from the scene geolocation and the current time of day. The scene illumination is pre-computed using radiosity transport for each of those components. To deal with dynamic illumination resulting from changing weather conditions, we estimate the outdoor light brightness in each input frame and scale the pre-computed illumination accordingly. We evaluate our approach on real-world data covering diverse outdoor illumination settings and show that our adaptable model is beneficial for direct camera tracking.**

***Index Terms*—Visual Tracking, Localization, Mapping, RGB-D Perception**

## I. Introduction

VISUAL camera tracking has been an important research topic for the last decades in the robotics and computer vision communities. Indirect methods rely on the detection of prominent image features such as corners or lines. In contrast, direct approaches also utilize weak image gradients improving tracking in featureless areas [1]. However, in the case of direct frame-to-model camera tracking, this requires photometric consistency between the live color image and the model, which is not given under dynamic illumination conditions. For example, the position and shape of a shadow on a wall can change according to the scene illumination. If the model does not account for such changes, camera tracking may fail in frames where shadows induce the dominant image gradients.

In our previous work [2], we addressed this problem by adapting the map to the current scene illumination, where the map is represented as a 3D mesh with surface reflectance and radiosity information. We detect and segment artificial light sources, i.e., lamps, and use *radiosity transport* [3] to pre-compute the illumination components for each light source. In camera tracking mode, we detect the set of lamps currently switched on and combine the corresponding illumination components to adapt the scene appearance in real-time. We showed that tracking against the adapted map is more stable than without illumination adaptation.

Manuscript received: February, 24, 2021; Revised May, 13, 2021; Accepted May, 31, 2021.

This paper was recommended for publication by Editor Sven Behnke upon evaluation of the Associate Editor and Reviewers' comments.

[1]Michael Krawez, Tim Caselitz, and Wolfram Burgard are with Autonomous Intelligent Systems, University of Freiburg, 79110 Freiburg, Germany {krawez, caselitz, burgard}@informatik.uni-freiburg.de

[2]Jugesh Sundram and Mark Van Loock are with Toyota Motor Europe, R&D - Advanced Technology, B-1930 Zaventem, Belgium {Jugesh.Sundram, Mark.Van.Loock}@toyota-europe.com

Digital Object Identifier (DOI): see top of this page.

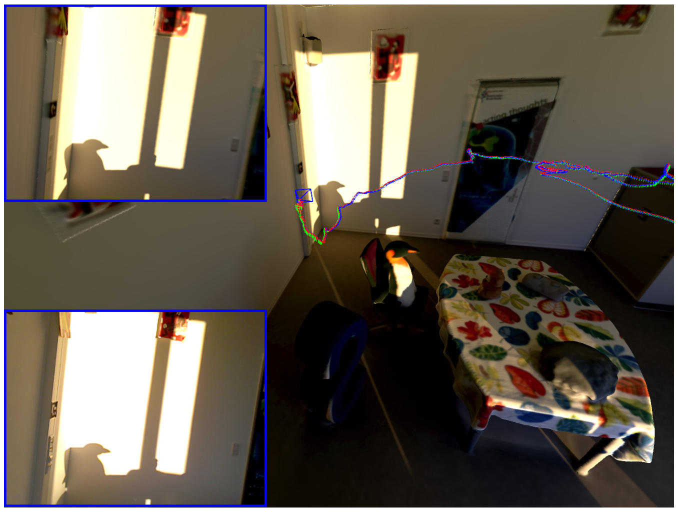

Fig. 1: 3D model of a scene used for camera tracking. We estimate the outdoor illumination using the current camera image shown in the left bottom corner. The image in the left top corner is rendered from the model with the same camera view.

During day time, however, an indoor environment is typically illuminated by outdoor light coming through windows. Outdoor illumination depends on several parameters that change dynamically through the course of a day. Among the most predominant ones are the position of the sun and the different weather conditions. Since direct sun light alters the scene appearance even more dramatically than artificial light sources, it is desirable to incorporate outdoor illumination into a lighting adaptable environment model. A novel approach to consider the effects of outdoor illumination is the major contribution of this paper.

We build our method upon a dense reflectance map [4] as before [2]. Our outdoor illumination model consists of three components, namely direct sun light, sky and ground, where we make the simplifying assumption that each component has a uniform brightness. The sun position is computed analytically from the current time and a given geographical location and orientation of the scene. Then, we employ radiosity transport to estimate the scene illumination for sun, sky and ground

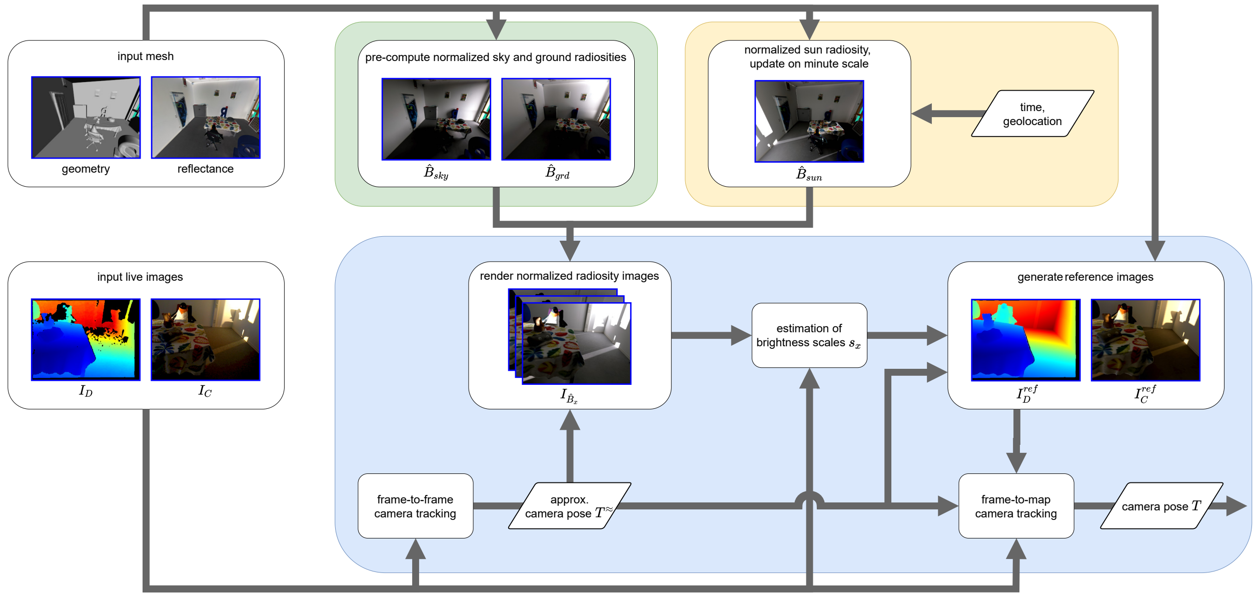
input mesh
geometry
reflectance
pre-compute normalized sky and ground radiosities
$\hat{B}_{sky}$
$\hat{B}_{grd}$
normalized sun radiosity, update on minute scale
$\hat{B}_{sun}$
time, geolocation
input live images
$I_D$
$I_C$
render normalized radiosity images
$I_{\hat{B}_x}$
estimation of brightness scales $s_x$
generate reference images
$I_D^{ref}$
$I_C^{ref}$
frame-to-frame camera tracking
approx. camera pose $T^{\approx}$
frame-to-map camera tracking
camera pose $T$

Fig. 2: Overview of our system. As input it takes the scene geometry, reflectance and geolocation, as well as an RGB-D image stream. We operate on three timescales: the normalized radiosities for sky and ground are pre-computed once for the scene (green box), the normalized sun radiosity is updated on minute scale (yellow box), brightness estimation and camera tracking are performed for each frame (blue box).

up-to-scale, i.e., normalized to the brightness of the respective component. Even though models exist which predict the sun and sky brightness analytically, they are only meaningful for a clear sky and do not account for dynamic weather changes. We therefore estimate the outdoor light brightness in real-time for each frame, where we fit the pre-computed normalized illumination to the input color image.

Our main contribution is a real-time system that continuously and at 30 frames per second adapts the appearance of the map to the present outdoor illumination, accounting for both, long- and short-time light dynamics. The real-time capability is achieved through splitting the process into pre-computation of the spatial light distribution and per-frame scaling of the illumination brightness, where parallelization on GPU is employed. A further contribution is that our approach to adapting the model to the outdoor light improves direct frame-to-model camera tracking. We perform extensive testing on a wide range of real-world data covering illumination changes due to season, time of the day and weather. Besides camera tracking, our system can be used in augmented and mixed reality applications.

## II. Related Work

Feature-based approaches to visual camera tracking rely on finding descriptive parts of the image, such as corners or lines, and on matching these across multiple images. One of the most prominent feature-based methods is ORB-SLAM [5] and its extensions [6], [7]. Such methods work well in texture-rich environments but can fail if not sufficient features of choice are detected [1]. Direct methods align a query image to a reference frame or model by minimizing an error function based on raw pixel values, where the error can be, among others, photometric [8], [9], geometric [10], or based on mutual information [11]. Thus, direct methods can exploit even weak image gradients in areas that lack pronounced features. However, also a combination of feature-based and direct approaches is possible [12]. Our approach aims to improve direct, model-based tracking methods. Its most distinct characteristic is how changes in the scene illumination are handled. Where existing methods try to operate on the illumination-invariant elements of the map, we propose to explicitly model aspects of dynamic illumination, e.g., shadows, thus providing additional information to the tracking algorithm.

Our adaptable map approach is closely related to the problem of artificial scene re-lighting from the augmented reality (AR) community. There, however, the goal is to render a virtual object into a real scene, whereas our objective is to keep the model appearance itself consistent with the current scene illumination. We further review works which exploit outdoor illumination for localization or camera tracking.

Daylight modeling plays an important role in architectural design, as it is desirable to know how the interior of a building is illuminated throughout a day and seasons before it is built. Daylight Coefficients [13] is an illumination model very similar to the one we employ in our work. It also considers a sun, a sky, and a ground light component, where the sky and ground are further subdivided into segments. The radiosity contribution of each segment to an interior scene point is pre-computed and can be scaled efficiently by the light intensities of the segments later on. Over the years numerous modifications and extensions of this basic model were proposed [14], [15]. In contrast to our work, the above neither estimate light brightness from live color images, nor is the model used to support camera tracking.

Kolivand *et al.* [16] blend a virtual building model into a real outdoor scene. Analogously to our approach, they derive the sun position from geolocation and time and use it to render realistically cast shadows. Also, they employ a sky dome to model ambient illumination. However, the sun and sky color and intensity are acquired from an atmosphere model and not estimated from image data as in our work. Madsen *et al.* [17] employ an illumination model similar to ours in order to re-light a virtual object in a photograph of an outdoor scene, where also the light intensity is estimated from the image. However, they assume a static camera pose and re-light the object and not the scene itself.

Ma *et al.* [18] use the sun direction to improve global outdoor localization in the context of autonomous driving. They estimate the sun position relative to the car using a convolutional neural network and obtain the car orientation by comparing it to the true sun direction computed from local time. Similarly, Clement *et al.* [19] deploy the sun location to reduce angular drift in their visual odometry method.

Outdoor illumination prediction was also used by Mashita *et al.* [20] to improve global feature-based localization. The authors detect point features on a 3D model of a building and then predict the feature appearance by rendering the model for different outdoor lighting conditions. They propose to store the appearance variation of each feature point as a distribution over the descriptor vectors and describe how those can be used for image matching. Similar to our method, the authors use rendering to predict scene appearance in order to robustify localization. However, their method is feature-based whereas we are targeting direct camera tracking. Thus, our method can be applied in scenes with little or no distinct features by exploiting weak gradients such as soft shadow borders.

Liu and Granier [21] estimate the dynamic sun and sky brightness in a video of an outdoor scene captured by a moving camera. They derive the sun position from geo-location and time as we do in our approach. The authors cluster sparse point features based on reflectivity and surface normals, and track these features across frames to detect illumination changes. Similarly to [16] and [17], this method aims at AR applications where the estimated illumination parameters are used to render an artificial object into the video, but not to improve camera tracking. In contrast, we adapt the actual scene model to the illumination estimate, which in turn improves camera tracking.

## III. Proposed Method

Our goal is to improve direct frame-to-model (f2m) camera tracking by adapting the model appearance to the current outdoor illumination conditions. As input we use a 3D mesh with diffuse reflectance values, which can be constructed as in our previous work [4]. Our outdoor illumination model (subsection III-A) consists of a sun, a sky, and a ground component, for which we pre-compute the *normalized* radiosities $\hat{B}_{sun}$, $\hat{B}_{sky}$ and $\hat{B}_{grd}$ (subsection III-B). Given the brightness scale $s_x$ of a component $x \in \{sun, sky, grd\}$ the radiosity $B_x$ can then be computed as $s_x \hat{B}_x$ in real-time. $\hat{B}_{sky}$ and $\hat{B}_{grd}$ must be calculated only once for a mesh, whereas $\hat{B}_{sun}$ changes with the sun position and is constantly updated in a separate thread (subsection III-E).

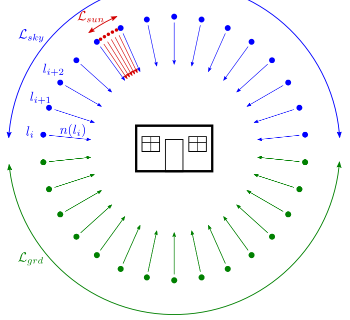

Fig. 3: Schema of our outdoor illumination model. A set of directional light sources $l$ with directions $n(l)$, represented as dots and arrows, are placed uniformly around the scene, forming a sphere. These are split by the horizon plane into the subsets $\mathcal{L}_{sky}$ (blue) and $\mathcal{L}_{grd}$ (green). For a given sun position, the sun area is modeled as $\mathcal{L}_{sun}$ (red) where the samples $l$ are placed more densely.

In tracking mode, for every new frame we first employ frame-to-frame (f2f) tracking to obtain a coarse camera pose $T^{\approx}$, which is used to render a normalized radiosity image $I_{\hat{B}_x}$ for each illumination component. Next, the brightness scales $s_x$ are estimated based on the input color image (subsection III-C). Finally, we generate an RGB image of the adapted map and use it as reference in f2m tracking (subsection III-D). Figure 2 summarizes our pipeline.

### A. Outdoor Illumination Model

At day time an indoor scene is typically illuminated by light coming through windows. Ignoring artificial light sources like street lights, outdoor illumination is ultimately generated by the sun. That can be further divided into three main components, namely direct sun light, sky and ground illumination. We model the outside world as being projected onto a sphere with infinite radius as shown in Figure 3. Each point $l$ on the sphere is considered a directional light source defined by its direction $n(l)$ and brightness scale $s_l$. The sun, sky and ground components are modeled as sets of such points, namely $\mathcal{L}_{sun}$, $\mathcal{L}_{sky}$ and $\mathcal{L}_{grd}$, where for brevity we write $\mathcal{L}_x$ with $x \in \{sun, sky, grd\}$. We further make the simplifying assumption that all $l$ in one $\mathcal{L}_x$ have the same brightness scale $s_x$.

$\mathcal{L}_{sun}$ depends on the sun position which can be computed analytically from the current date and time of the day and the geographic location of the scene, which we assume as given. In several approaches dealing with outdoor illumination [16], [20], [21], the sun is modeled as a light source with a single direction, i.e., one point on the sky sphere, where in fact the sun occupies an non-neglectable area on the sky. Thus, the sun can shine "around" geometry edges producing narrower shadow borders when compared to the point sun model, as demonstrated in Figure 4. For many use cases the former simplification might be sufficient. However, in our camera tracking application we want to exploit sun light gradients such as cast shadows, thus a precise shadow geometry is required.

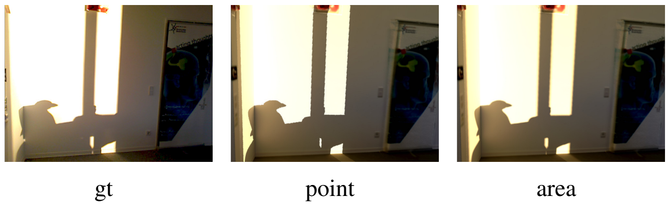

Fig. 4: Differences in cast shadows depending on sun model. The point sun model generates thicker cast shadows than in the ground truth (gt) image, best seen on the vertical window frame shadow in image center. The area sun model produces smoother shadow borders with the shadow area better matching the gt.

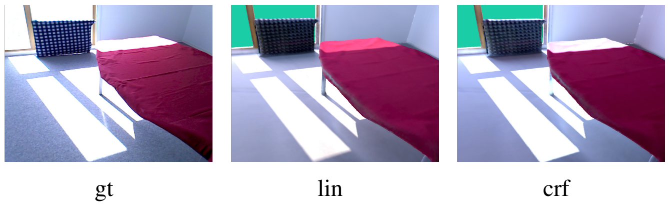

Fig. 5: Comparison of illumination scales estimated with the linear-least-squares (lin) and CRF-based error function. The lin approach tends to underestimate the brightness in over-exposed image regions, as seen in the sun-lit part of the table. The CRF method, in contrast, is significantly closer to the ground truth (gt).

Therefore, we construct $\mathcal{L}_{sun}$ from multiple points $l$ sampled uniformly in the area around the computed sun position. The resulting shadows, shown on the right of Figure 4, are closer to the real-world data than the point model.

The remainder of outdoor illumination originates either from sunlight being scattered in the atmosphere or being reflected by the ground. We sample points $l$ by creating an icosphere and assign $l$ above the horizon to $\mathcal{L}_{sky}$ and to $\mathcal{L}_{grd}$ otherwise. Here, we favor a computationally efficient sky and ground model in order to support real-time capability, but it would be straightforward to extend it if needed. Thus, $\mathcal{L}_{sky}$ and $\mathcal{L}_{grd}$ could be partitioned into a number of subsets in order to account for spatial brightness differences of the ground and sky. Further, $\mathcal{L}_{grd}$ could be computed using a 3D mesh, if actual outdoor geometry is available.

## B. Normalized Radiosity

Given the three illumination components $\mathcal{L}_x$ we aim to adapt the radiosity in the model to the scene's current lighting condition in real-time. We observe that $\mathcal{L}_{sky}$ and $\mathcal{L}_{grd}$ are static, while $\mathcal{L}_{sun}$ may change quickly, and that only the brightness scales $s_x$ can be highly dynamic depending on the weather conditions. Our idea is thus to pre-compute the scene illumination for each $\mathcal{L}_x$ with all $s_x$ set to 1 to obtain what we call normalized radiosites $\hat{B}_x$. Once the $s_x$ are estimated as shown in subsection III-C we can perform fast re-scaling to obtain the radiosities $B_x = s_x \hat{B}_x$.

To simulate light propagation within the scene we employ radiosity transport where we refer the reader to the original paper [3] for detailed background and derivation, or to our previous papers [2], [4]. In this paper, we present a variant of the algorithm suited to the considered problem. We operate on a mesh with the set of vertices $V$, where each $v \in V$ has a reflectance value $\rho(v)$. The radiosity $B(v)$ here is the amount of light reflected by the surface patch at $v$, and $B_x(v)$ the radiosity originating from light component $\mathcal{L}_x$. In order to estimate $B_x(v)$, we first compute the radiosity $B_l(v)$ induced by a single directional light source $l \in \mathcal{L}_x$ and then sum up these parts.

We use an iterative version of the basic radiosity algorithm, with

$$B_l^0(v_i) = \rho(v_i) s_l F(l, v_i) G(l, v_i) \tag{1}$$

being the light reaching the scene directly from $l$, and

$$B_l^{k+1}(v_i) = \rho(v_i) \sum_{j \neq i} B_l^k(v_j) F(v_i, v_j) G(v_i, v_j) \tag{2}$$

being an iterative estimate for light inter-reflections within the scene. The form factor $F(v_i, v_j)$ is computed as in our previous work [4], for a directional light source $F(l, v_i)$ simplifies to $-cos(n(l) \cdot n(v_i))$. The visibility term $G(v_i, v_j)$ is 1 if the line of sight between $v_i$ and $v_j$ is free, and 0 otherwise. Similarly, $G(l, v_i) = 1$ only when the ray with direction $n(l)$, starting at $v_i$, does not intersect any scene geometry. We set $B_l(v_i) = B_l^K(v_i)$, with $K = 10$ in our experiments, and $B_x(v_i) = \sum_{l \in \mathcal{L}_x} B_l(v_i)$. Setting $s_l = 1$ in Equation 1 gives us $\hat{B}_x$, and once the $s_x$ are estimated, the scaled scene appearance in radiosity space is $B = \sum_{x \in X} s_x \hat{B}_x$ with $X = \{sun, sky, grd\}$.

The above radiosities and scales are computed separately for each of the three color channels. $\hat{B}_{sun}$ is constantly adapted to the current sun position in a thread running in parallel to camera tracking, more details are given in subsection III-E.

## C. Brightness Scale Estimation

The brightness of the sun, sky and ground can change dynamically, demanding a constant update of the model illumination. In each frame we thus compute the scales $s_x$ taking the current input RGB image $I_C$ as reference. Using the corresponding camera pose $T^{\approx}$ and $\hat{B}_x$ we render a normalized radiosity image $I_{\hat{B}_x}$ for each illumination component. To reduce computational costs we operate on a pixel subset $\Omega$, which is sampled from $I_C$ in a uniform grid of 15 pixels, meaning that at most $1/15^2$ of all pixels are used. We further filter $\Omega$ by removing pixels with missing model geometry or with low model reflectance, since the radiosity estimate in such areas tends to be unreliable. Here, we choose a reflectance threshold of 0.03. For a pixel $u$ we define

$$I_B(u) = \sum_{x \in \{sun, sky, grd\}} s_x I_{\hat{B}_x}(u) \tag{3}$$

and with that the error function

$$E_{crf} = \sum_{u \in \Omega} \Big( I_C(u) - f\big(\Delta t \cdot I_B(u)\big) \Big)^2, \tag{4}$$

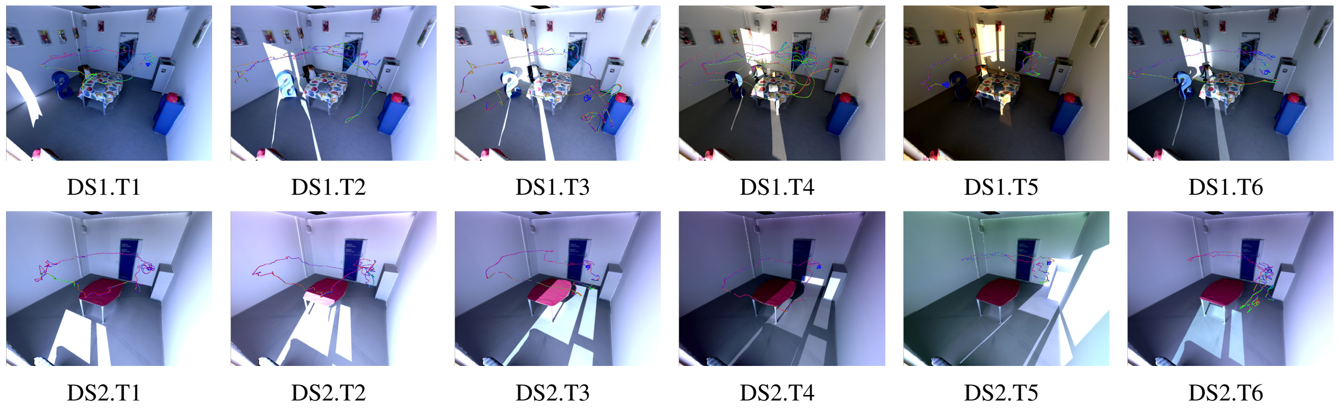

Fig. 6: Illumination conditions and GT camera trajectories for scene DS1 (top) and DS2 (bottom).

where $f()$ is the camera response function (CRF) and $\Delta t$ the exposure time of $I_C$. The scales $s_x$ are then obtained by minimizing $E_{crf}$:

$$s_x^* = \arg\min_{s_x} E_{crf} \tag{5}$$

The simulation of the image formation process by the CRF in the error function is crucial to correctly handling over-exposed pixels in the reference image. Such pixels essentially tell us a lower bound of the model radiosity and not its exact value. Formulating the error function as a linear least squares problem can lead to under-estimation of the brightness in over-exposed areas which are ubiquitous in our daylight scenario. We demonstrate this in Figure 5 by minimizing an alternative error function

$$E_{lin} = \sum_{u\in\Omega} \left( I_B^{live}(u) - I_B(u) \right)^2, \tag{6}$$

with $I_B^{live} = f^{-1}(I_C)/\Delta t$ being the live color image transformed into radiosity space before the minimization.

Changes of outdoor lighting usually occur on a time scale of seconds [21], so that we can expect the brightness to vary only marginally between consecutive frames. That allows us to average the per-frame brightness scale estimates in a short time window, here 30 frames, leading to more robust scales $s_x$ while remaining flexible enough to cope with dynamic lighting.

### D. *Camera Tracking*

For each frame at time $t$ we aim to estimate the camera pose $T_t \in SE(3)$ relative to the scene. To that end, we follow our previously used tracking method [2] which solves

$$T^* = \arg\min_T E_C(T) + w_G E_G(T), \tag{7}$$

where $E_C$ is a photometric error term [8], $E_G$ a geometric error [10] used for additional robustness, and $w_G$ the weight between $E_C$ and $E_G$, set to 10 in the experiments. The live depth image is utilized in $E_G$ and is also used for projective data association.

$T_t$ is obtained in a two-step approach. First, we perform frame-to-frame tracking by solving Equation 7 with the previous frame $t-1$ as reference. This gives us the initial pose estimate $T_t^{\approx}$. With that, we adapt the brightness scales of the model as described in subsection III-C where $T_t^{\approx}$ is used for rendering and the input color image $I_C$ as reference for scale estimation. Given the scales, we render the color image $I_C^{ref}$ as well as the depth image $I_D^{ref}$ which serve as reference in Equation 7 for the frame-to-model tracking. The minimization procedure is initialized with $T_t^{\approx}$ and returns the final pose estimate $T_t$.

### E. *Implementation Details*

We aim for a real-time tracking system which operates at a frame rate of at least 30fps. To achieve this we make heavy use of parallel processing on modern GPU hardware. Here we outline the details of our implementation we consider crucial for real-time performance.

The system is implemented in C++, where the Optix library is used for ray tracing. Most of the image processing and radiosity estimation is implemented in CUDA. Equation 4 for brightness estimation is minimized with the Levenberg-Marquardt algorithm using the ALGLIB[1] library.

In camera tracking mode, brightness estimation and tracking run in the same CPU thread, while in a parallel thread the normalized radiosity $\hat{B}_{sun}$ for the next sun position is computed. The linearity in Equation 1 and Equation 2 allows us to divide the $\hat{B}_{sun}$ estimation into chunks, with only one chunk executed per frame. First, by choosing a proper chunk size, we can control how much GPU processing resources are needed for the sun update per frame, thus guaranteeing a stable frame rate in the tracking thread. Second, we know how many frames the sun update will take and therefore choose the next sun position accordingly.

The direct sun light component $\hat{B}_{sun}^0$ requires $|V| \cdot |\mathcal{L}_{sun}|$ ray tracing operations, where we compute one $\hat{B}_l^0$ per frame for all $l \in \mathcal{L}_{sun}$. In contrast, one iteration of $\hat{B}_{sun}^{k+1}$ requires $|V|^2$ ray computations. With $|V|$ typically being on a scale of $10^6$ this becomes intractable in practice, thus we partition the model into a set of uniformly sized patches $P$, where each $p \in P$ groups connected vertices in planar regions. Equation 2 is modified to compute radiosity propagation from patch to

[1]ALGLIB (www.alglib.net), Sergey Bochkanov

vertex instead of vertex to vertex, as described in [4]. This reduces the above costs to $|V| \cdot |P|$ with $|P| \ll |V|$.

Further, the visibility term $G$ in Equation 2 depends only on the scene geometry which is assumed to be static. Therefore we can pre-compute $G$ for all patch-vertex pairs and use the results to avoid ray tracing in the sun update. As CPU to GPU memory copies are expensive, the complete visibility data is uploaded as a bit array to GPU memory only once during program start. The memory requirement for that data is $|V| \cdot |P|$ bits and can be fit to available GPU memory by modifying the number of patches.

## IV. Experimental Evaluation

In this section we present the experiments we performed to validate our approach. First, we describe the experimental setup, the recorded datasets, and details on model construction. Then we present our experiments on camera tracking and discuss the results. Finally, we perform a run-time analysis of our method.

### A. Setup

The experiments were carried out in a 4.5x4.8x2.9m sized room with one window, where we set up two scenes, DS1 and DS2, as shown in Figure 6. DS1 is richly decorated such that a high number of geometric and texture features is provided. In contrast, DS2 contains just a minimal amount of texture and geometry.

For tracking evaluation we recorded twelve RGB-D sequences using an Asus Xtion Pro camera. The sequences DS1.T1-T6 and DS2.T1-T6 each capture the respective scene with different daylight illumination, where distinct times of the day, seasons and weather conditions are covered. Accordingly, data acquisition for DS1 took place over the course of three days in winter and for DS2 on a single day in summer. T1-T5 depict both scenes under clear sky conditions, whereas DS1.T6 and DS2.T6 display dynamics in sun brightness due to cloud movement. We disabled the camera's auto-exposure and in each sequence set the exposure and gain to a fixed value, appropriate to the corresponding illumination. The camera trajectories and illumination conditions are shown in Figure 6.

Ground truth (GT) poses were acquired by means of the HTC Vive tracking system with one base station being placed in each corner of the room and a tracker being mounted on top of the camera. Vive's global coordinate frame is not static [22] and may change if the tracker is switched off, however, we require all GT poses in one scene to be in the same reference frame. To this end, we mounted a second tracker at the ceiling in sight of all four base stations and computed the GT poses relative to this static tracker. Reflective surfaces and bright sunlight can negatively affect the tracking stability of the Vive system, resulting in occasional jumps in reported poses. Thus, we manually filtered out such poses and excluded them from the evaluation.

### B. Model Construction

We first constructed an initial geometric model with the hashed voxel method [23] with a resolution of 0.5cm for DS1 and 0.7cm for DS2. Since direct sunlight causes depth data loss in Asus Xtion depth images, both scenes, DS1 and DS2, were scanned after sunset with ceiling lamps switched on.

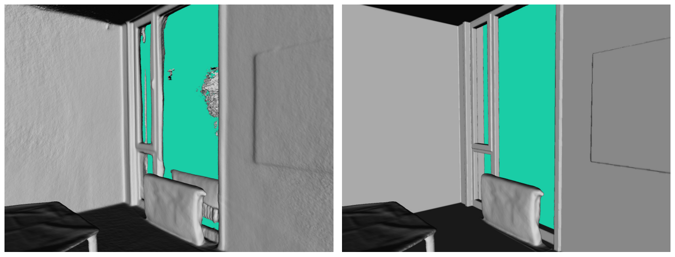
Fig. 7: Window geometry recovery. On the left, the reconstruction from raw TSDF exhibits geometric noise as a result of glass reflections. On the right the geometry is augmented by a CAD model.

Window glass poses a fundamental problem for an active depth sensor, since it partially reflects the light of the infrared projector. This causes depth image noise on glass areas, resulting in geometry clutter blocking the window opening. Further, parts of the window frame geometry behind the glass are missing in the reconstruction. We addressed this problem by replacing the window frame in the TSDF volume with a manually created CAD model. Further, we augmented the scene geometry by fitting large flat surfaces, such as walls, ceiling, and floor, to planes. The difference in geometry before and after augmentation is demonstrated in Figure 7. After the mesh had been extracted from the TSDF volume, we proceeded as in [4] to recover the scene reflectance.

### C. Camera Tracking

We carry out an ablation study to analyze the contribution of the illumination adaptation to tracking performance. First, the *refl-f2m* baseline differs to the proposed method in that the reference image in f2m tracking is rendered from the reflectance map and scaled to fit the mean brightness of the live color image. The idea here is to provide texture gradients which are not affected by illumination. Second, *depth-only* employs only depth information for both the f2f and f2m steps, i.e., $E_C$ is not considered in Equation 7. Finally, *no-color-f2m* performs f2f tracking with both, $E_C$ and $E_G$, but uses only $E_G$ in model tracking. In other words, it utilizes all information as the proposed approach except the model color.

We evaluate the baselines and the proposed method on the twelve test sequences by computing two metrics, the root mean square error (RMSE) [24] and the number of tracking losses per sequence. For the RMSE, we only count poses deviating from GT less than 10cm in translation and $10°$ in rotation. Otherwise, we consider the tracking to be lost and reset the tracker to the corresponding GT pose.

Table I summarizes the results of these experiments. The bottom row of the table shows the average values over all test sequences. There, the proposed method has 3.6cm RMSE and $2\%$ tracking loss, which outperforms all considered baselines. All baselines are slightly worse on the RMSE score but have

| Sequence | refl-f2m | depth-only | no-color-f2m | proposed |
|---|---|---|---|---|
| DS1.T1 | 0.0418 | 0.0433 | 0.0433 | **0.0360** |
| | 9.67% | 1.67% | 1.63% | **0.33%** |
| DS1.T2 | 0.0527 | 0.0491 | 0.0491 | **0.0457** |
| | 17.28% | 12.15% | 12.06% | **1.20%** |
| DS1.T3 | 0.0448 | 0.0397 | 0.0397 | **0.0240** |
| | 22.24% | 7.92% | 7.90% | **2.70%** |
| DS1.T4 | 0.0374 | 0.0487 | 0.0488 | **0.0279** |
| | 7.41% | 10.94% | 10.86% | **2.24%** |
| DS1.T5 | **0.0429** | 0.0508 | 0.0508 | 0.0453 |
| | 23.80% | 10.16% | 10.16% | **4.38%** |
| DS1.T6 | 0.0356 | 0.0428 | 0.0427 | **0.0353** |
| | 1.89% | 2.38% | 2.34% | **0.00%** |
| DS2.T1 | 0.0655 | 0.0417 | 0.0418 | **0.0359** |
| | 59.55% | 8.48% | 9.10% | **0.47%** |
| DS2.T2 | 0.0713 | 0.0406 | 0.0408 | **0.0252** |
| | 80.43% | 11.23% | 11.28% | **0.26%** |
| DS2.T3 | 0.0671 | 0.0526 | 0.0524 | **0.0279** |
| | 51.53% | 28.22% | 27.93% | **0.22%** |
| DS2.T4 | 0.0626 | 0.0438 | 0.0437 | **0.0270** |
| | 40.00% | 24.84% | 24.73% | **0.16%** |
| DS2.T5 | 0.0654 | 0.0508 | 0.0508 | **0.0443** |
| | 63.61% | 37.84% | 37.94% | **3.47%** |
| DS2.T6 | 0.0649 | 0.0434 | **0.0434** | 0.0463 |
| | 44.28% | 10.98% | 11.05% | **9.65%** |
| Average | 0.0499 | 0.0453 | 0.0453 | **0.0361** |
| | 31.98% | 11.89% | 11.90% | **2.16%** |

TABLE I: Tracking experiment results by method. The top number of each cell is the RMSE in meters of the tracking method on the corresponding sequence. The bottom number is the percentage of tracking loss w.r.t. the number of frames in that sequence. The lowest values on one sequence, for both metrics, are highlighted.

much higher tracking loss numbers of over 11% for depth-only and no-color-f2m, the clearly worst performing baseline is refl-f2m with over 30% tracking loss.

The results on individual sequences mirror this performance hierarchy with the proposed method being at the top, with two exceptions: On DS1.T5 the proposed method is marginally outperformed on RMSE by refl-f2m with the latter, however, having a significantly higher tracking loss. On DS2.T6, no-color-f2m scores slightly lower on RMSE and slightly higher on tracking loss, making the results comparable to the proposed approach.

We interpret the above results as follows. The attempt to match the live image with a high amount of illumination-induced gradients against a reflectance-based reference image, naturally lacking these gradients, causes tracking to fail in many cases. This explains the poor performance of the refl-f2m baseline. In contrast, depth-only and no-color-f2m operate only on the geometric model and therefore perform more stably, however, they also lack the additional color constraints given by the illumination-adapted model. Thus we conclude that a model with color information is beneficial for direct camera tracking, but only if it can be adapted to the current illumination.

We further examined how the brightness scale estimation parameters influence tracking. Decreasing the size of the time window for scale averaging below 30 frames did not have a significant effect on the tracking performance. However, this resulted in noticeable color balance jumps between frames which might be problematic for, e.g., AR applications. Choosing the linear error function defined in Equation 6 instead of the CRF-based as in Equation 4 significantly increased the tracking loss from 2% to 10% averaged over all trajectories.

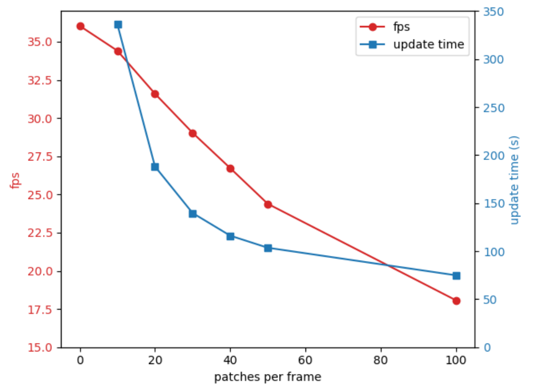

Fig. 8: Average frames-per-second depending on number of patches processed per frame (red line) and corresponding time required for one sun radiosity update (blue line).

Another aspect worth discussing is depth value loss in direct sunlight. Pixels with missing depth values are not considered in the geometric error term, thus, tracking becomes less robust for images where large scene portions are lit by direct sun. Also, since the live depth image is used for data association, these pixels are neither utilized in the photometric error. However, the latter is only a minor issue as missing depth values coincide with overexposed pixels that produce virtually no gradients, i.e, carry no additional information.

## D. *Run-Time Analysis*

The time profiling is carried out on a system with an AMD Ryzen 7 3700x CPU, an RTX 2080 Ti GPU and 32GB of RAM. We perform the run-time analysis on the DS2.T2 sequence with 4169 frames and 2.1 million vertices in the scene mesh. We choose a patch size of 10x10cm with 11.000 patches in total and $K = 10$ iterations for inter-reflection estimation.

Figure 8 shows the system performance in frames-per-second (fps) depending on the number of processed patches per frame (ppf), where 0 ppf means that we tracked on a pre-computed sun radiosity without a background update. The average fps is 35 for 0 ppf and drops linearly to 17 fps for 100 ppf. Two settings allow real-time camera tracking with over 30 fps, namely those with 10 and 20 patches processed per frame. In the latter case, a full sun radiosity update is completed in less than 4 minutes. For higher ppf numbers the workload per frame is increasingly dominated by the sun update, decreasing the frame rate and asymptotically pushing the update time to 75 seconds.

In Table II we also analyze the average per-frame run-time of the main system components, namely f2f and f2m tracking, rendering of the radiosity images, and brightness estimation.

| patches per frame | components | | | |
|---|---|---|---|---|
| | f2f | f2m | rendering | estim. |
| 0 | 6.8 | 5.8 | 0.6 | 15 |
| 20 | 9.2 | 5.8 | 2 | 15 |
| 100 | 9.6 | 5.8 | 24.8 | 15 |

TABLE II: Average per-frame run-time (ms) of individual components.

Without the sun update, processing time is evenly split between tracking and brightness estimation with approx. 15ms both and rendering being neglectable with 0.6ms. However, during a sun update with 100 ppf, the rendering time rises to 25ms, while the other components remain relatively constant. We explain this by the fact that rendering is the component relying most on the GPU, which is heavily used for sun updating.

## V. Conclusion

In this paper, we presented an approach to adapting the appearance of a dense 3D map to the current outdoor illumination. We use an outdoor illumination model consisting of sun, sky and ground components. The scene illumination is pre-computed for each component and is then scaled in real-time to match the present illumination brightness, which is estimated for each input image. In our experiments we show that direct camera tracking is more robust and accurate using the adapted map compared to a map without light adaptation.

One possible direction for future work is the automatic removal of geometry clutter on windows. An interesting approach in this context is to derive window geometry from shadows cast by the window frame, captured at different time points [25]. Further, it might be interesting to investigate how outdoor illumination can be exploited in the context of global localization.

## References

[1] N. Yang, R. Wang, X. Gao, and D. Cremers, "Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 2878–2885, 2018.

[2] T. Caselitz, M. Krawez, J. Sundram, M. Van Loock, and W. Burgard, "Camera tracking in lighting adaptable maps of indoor environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[3] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile, "Modeling the interaction of light between diffuse surfaces," *ACM SIGGRAPH Computer Graphics*, vol. 18, no. 3, pp. 213–222, 1984.

[4] M. Krawez, T. Caselitz, D. Büscher, M. Van Loock, and W. Burgard, "Building dense reflectance maps of indoor environments using an RGB-D camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[5] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics (T-RO)*, vol. 31, no. 5, pp. 1147–1163, 2015.

[6] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics (T-RO)*, vol. 33, no. 5, pp. 1255–1262, 2017.

[7] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM," *IEEE Transactions on Robotics (T-RO)*, 2021.

[8] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense RGB-D images," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011.

[9] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, 2014.

[10] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[11] G. Caron, A. Dame, and E. Marchand, "Direct model based visual tracking and pose estimation using mutual information," *Image and Vision Computing*, vol. 32, no. 1, pp. 54–63, 2014.

[12] P. F. Georgel, S. Benhimane, and N. Navab, "A unified approach combining photometric and geometric information for pose estimation," in *British Machine Vision Conference (BMVC)*, 2008.

[13] P. R. Tregenza and I. Waters, "Daylight coefficients," *Lighting Research & Technology*, vol. 15, no. 2, pp. 65–71, 1983.

[14] D. Bourgeois, C. F. Reinhart, and G. Ward, "Standard daylight coefficient model for dynamic daylighting simulations," *Building Research & Information*, vol. 36, no. 1, pp. 68–82, 2008.

[15] G. Besuievsky, E. Fernández, J. P. Aguerre, and B. Beckers, "A radiosity-based methodology considering urban environments for assessing daylighting," in *Journal of Physics: Conference Series*, 2019.

[16] H. Kolivand, A. El Rhalibi, M. S. Sunar, and T. Saba, "Revitage: Realistic virtual heritage taking shadows and sky illumination into account," *Journal of Cultural Heritage*, vol. 32, pp. 166–175, 2018.

[17] C. B. Madsen, T. Jensen, and M. S. Andersen, "Real-time image-based lighting for outdoor augmented reality under dynamically changing illumination conditions," in *International Conference on Graphics Theory and Applications*, 2006.

[18] W.-C. Ma, S. Wang, M. A. Brubaker, S. Fidler, and R. Urtasun, "Find your way by observing the sun and other semantic cues," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[19] L. Clement, V. Peretroukhin, and J. Kelly, "Improving the accuracy of stereo visual odometry using visual illumination estimation," in *International Symposium on Experimental Robotics*, 2016.

[20] T. Mashita, A. Plopski, A. Kudo, T. Höllerer, K. Kiyokawa, and H. Takemura, "Simulation based camera localization under a variable lighting environment," in *26th International Conference on Artificial Reality and Telexistence and the 21st Eurographics Symposium on Virtual Environments*, 2016.

[21] Y. Liu and X. Granier, "Online tracking of outdoor lighting variations for augmented reality with moving cameras," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 4, pp. 573–580, 2012.

[22] D. C. Niehorster, L. Li, and M. Lappe, "The accuracy and precision of position and orientation tracking in the HTC Vive virtual reality system for scientific research," *i-Perception*, vol. 8, no. 3, p. 2041669517708205, 2017.

[23] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, pp. 1–11, 2013.

[24] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[25] S. Savarese, M. Andreetto, H. Rushmeier, F. Bernardini, and P. Perona, "3D reconstruction by shadow carving: Theory and practical evaluation," *International Journal of Computer Vision*, vol. 71, no. 3, pp. 305–336, 2007.