# High-precision Ground-truth Data for Evaluating Dense Stereo and Optical Flow Algorithms

András Bódis-Szomorú, Tamás Dabóczi

Dept. of Measurement and Information Systems, Budapest University of Technology and Economics, Magyar tudósok körútja 2., 1117 Budapest, Hungary
{bodis,daboczi}@mit.bme.hu, http://www.mit.bme.hu/~{bodis,daboczi}

**Abstract.** Although very promising results have been achieved by others in producing reference disparity maps for real-world indoor scenes with uncalibrated structured light and engineered scenes, few papers emphasise that 3D-synthesis remains indispensable in generating ground-truth data up to numerical precision, and also for long-range stereo, to discriminate among the best-performing stereo and optical flow algorithms. We introduce a fast and flexible method for generating synthetic data by using a combination of a free, easy-to-use rendering software and algorithms for post-processing the rendered output to produce dense depth maps, disparity maps, occlusion maps and 2D motion fields.

## 1 Introduction

Research interest in correspondence algorithms have been revived in the last decade due to some elementary breakthroughs in optimization strategies, as well as, due to some novel, increasingly popular stereo and optical flow datasets. Although, feature-based sparse methods received significant attention earlier, today there seems to be a focus on dense correspondence of rectified stereo pairs, which is unquestionably one of the most challenging problems of binocular stereopsis, mainly because of occlusions, textureless areas, noise, geometric distortions, radiometric distortions, specular reflections and repeated patterns. Sound overviews of existing stereo registration methods are the work of Brown, Burschka and Hager [1] and that of Scharstein and Szeliski [2].

Many researchers tend to publish new algorithms and leave it to the reader to qualitatively judge the resulted disparity, depth or occlusion maps, while comparably less work has been done in meaningful, quantitative evaluation of the methods [3]. Therefore, encouraged by similar achievements in the optical flow field by Barron, Fleet and Beauchemin [4], [2] introduces a well-established taxonomy and a systematic, quantitative evaluation framework for correspondence algorithms on rectified stereo pairs. The same group published a similar work for optical flow recently [5]. They provide evaluation results of numerous existing algorithms, both stereo and optical flow, posted by other researchers, at the Middlebury website[1]. Recently, this website has received great attention

[1] http://vision.middlebury.edu/flow/ and vision.middlebury.edu/stereo/

by the computer vision community. However, engineered indoor scenes may not necessarily serve as reference for long-range outdoor applications.

Driven by the need for quantitative evaluations, we performed a quick survey on datasets publicly available for our purposes and noticed that only few contain reference depth or disparity maps. Some promising efforts have already been made by others [6] in acquiring dense reference range information for engineered indoor scenes, but few papers emphasise that 3D-synthesis remains indispensable for generating ground-truth data up to numerical precision, and also for long-range stereo. Since, to the best of our knowledge, there is a lack of similar guidelines, we present a fast and flexible method for generating synthetic data by using a combination of a free, easy-to-use rendering software and simple additional algorithms for post-processing the rendered output to produce dense depth maps, disparity maps, binary occlusion maps and dense 2D motion fields.

Section 2 is a survey of existing, available datasets and methods used for generating dense reference data for stereo and optical flow algorithms. The proposed method is overviewed in Section 3 and details are given in Section 4, including our camera description, depth map, disparity map, binary occlusion map and dense motion field generation technique. A validation of the proposed techniques and two generated datasets for complex scenes are presented in Section 5. Finally, Section 6 concludes the paper.

## 2 Existing datasets and related work

For the subjected purpose, there exist some well-known, but rather old datasets, such as the CMU/VAST, the JISCT database, or the one provided by the Univ. of Karlsruhe[2]. Most of the contained datasets do not include reference data.

For stereo vision, the *Tsukuba* image pair [7] is probably the most well-known stereo dataset. The images are taken of a real indoor scene and only a pixel-accurate hand-labeled ground-truth disparity map is given.

Depth maps from stereopsis for *real scenes* can be validated using several range sensing principles [8]. Structured light techniques are prevalent for generating close-range reference maps [8],[9]. [6] achieves very promising results with an uncalibrated off-the-shelf projector and a movable camera. The resulted datasets (*Cones*, *Teddy*, *Art* etc.) are getting popular for algorithm evaluations in recent papers. However, numerical precision can not be guaranteed due to finite resolution of the projector, specularities, dark holes, inter-reflections in the scene, code ambiguities, interpolation errors etc. [6].

The same reseach group provides datasets of real indoor scenes for optical flow. Grainy fluorescent paint visible only in UV-lighting is applied to all surfaces in the scene in order to track surface points for reference [5].

It is also possible to measure the performance of a matching algorithm with respect to another one that incorporates knowledge about the scene, e.g. [10] represents the scene as a collection of piecewise planar objects (*Venus* and *Sawtooth* datasets). These methods only work in the special environments they model.

[2] `http://i21www.ira.uka.de/image_sequences/`

While radars and lidars are typically used up to a depth of 100 m with long-range vision sensors in Advanced Driver Assistant Systems (ADAS) to reduce the risk of false or lack of detections, these are reported to acquire less detail of the scene than vision sensors do [11].

If ground-truth data is not available but images from a third view are at service instead, *image-based rendering* offers an alternative solution for measuring the quality of stereo algorithms [12],[13]. It is a good solution when no reference data is available for applications like virtual reality, view interpolation, frame-rate conversion etc. [12]. However, we believe that qualification based on ground-truth depth maps, if available, still remain more representative for applications in robotics, e.g. bin picking, autonomous robots, collision avoidance in ADAS. Also, trinocular data may not always be available. In these cases, or as a first quantitative test of an algorithm, synthetic data is reasonable, even though some results can not always be generalised for real scenes [13]. As stated in [5], good-quality rendered images can be indistinguishable from real ones.

The old *Yosemite* and *Marble block* sequences are the most widely used synthetic ones, as they include reference flow and depth data. Some more sophisticated rendered datasets are made available recently [5],[13]. Unfortunately, only the rendering engine is named and no description is given on how these datasets are exactly generated.

Rendering a dataset has the advantage that it can be done without the need for a trinocular system or specific light projector, artifacts present at structured light techniques can be avoided and engineering far-range scenes does not induce any additional problem. Bonn University provides an old but specialized renderer (MRT) for stereo depth map generation, but only from a very limited scene description. Much more complex commercial, free or open-source renderering engines capable of rendering photorealistic images exist today. Consequently, it is not reasonable to develop a new rendering engine with built-in disparity map, occlusion map and optical flow rendering capabilities.

## 3 Overview of the proposed method

We have chosen POV-Ray[3], a freely available raytracer with easy-to-use Scene Description Language (SDL) to produce photorealistic images and ground-truth data. POV-Ray takes a text description of objects, lights and atmospheric effects, and generates an image of the scene from a camera also defined within the scene description. The ray-tracing process constists of tracing viewing rays backwards, from the camera center into the scene, via each pixel of the viewing window. If the back-traced ray hits a surface in any point, the color of a pixel is calculated by sending rays from the surface point to each of the light sources in order to compute their contribution to the final color of the pixel based on the surface normal, albedo and transparency. More rays are traced to determine the effect of the refracted rays on the pixel color if the surface is not opaque. The procedure

[3] `http://www.povray.org/documentation/`

can result in photo-real images, though, by default, it does not take into account inter-diffuse reflections (radiosity), i.e. light reflected by all other surfaces when computing the image of a surface. Even approximation of this effect increases drastically the computation time. An alternative way to simulate radiosity is ambient illumination that renders some color to pixels that would be totally black due to the lack of radiosity. The proposed method has two main stages:

1. Render the original scene and the same scene with different special effects and lighting in POV-Ray. As a result, several output image pairs are produced for stereo, or single-frame images for optical flow.
2. Post-process the acquired synthetic images to get dense, high-precision reference data for each frame in the sequence.

The images rendered in normal conditions (illumination, texture) may serve as input for the stereo correspondence or optical flow algorithms under test, while the post-processed reference maps can be used to measure the output quality of the algorithms. The following reference data types are computed for each view:

- *Labeled images* assigning the identifier of the object seen in each pixel to the pixel. Identifiers are integer labels predefined by the user in the scene description. Labeled images are required by the motion field estimation method presented and may also serve as reference for image segmentation algorithms.
- *Depth maps* encoding the distance between the optical center and the surface point observed in each pixel center. One depth map is computed per view.
- *Disparity maps* encode the displacement at each pixel center in a reference image. A horizontal and a vertical map is computed for each image.
- *Occlusion maps* are binary maps indicating which surface point projection to a pixel center can not be seen in the other view. Thus, an occlusion map is always determined by another view taken into consideration.
- *2D motion field* in multiple senses: it is the 2D projection of the 3D motion of surfaces or it is the ground-truth disparities between two successive frames in a sequence. We provide reference data for both.

The depth maps, disparity maps and motion fields are all high-precision and all the maps are dense, i.e. computed on a per-pixel basis. Sub-pixel density can also be achieved by increasing the resolution of the output maps. Note that the two disparity maps are not coding the same information, because of occlusions and also, because their domain is discrete while the values are double-precision.

Depth and disparity maps are univalued representations of 3D point-samples of the observed surface. A depth map does not contain any data of a surface point seen in other views but occluded in the current one. For algorithms that take such points into consideration (e.g. space carving or image-based rendering), a multivalued/volumetric representation might be more appropriate. These algorithms are out of the scope of this paper.

A 2D motion field should be distinguised from *optical flow*, the apparent motion of brightness fields. A motion field is independent of textureless areas, movement of specular highlights, changes in texture or in lighting, while optical

flow depends on these effects [12]. In practice, the goal of opical flow computation is application-dependent. We focus on applications that aim to estimate the 2D motion field, e.g. robot navigation. Some applications, like inter-frame interpolation, that estimate the apparent motion, are excluded by now.

POV-Ray originally supports several camera types but not stereoscopic cameras. Although, several add-ons exist for this purpose, they are not required, since the cameras can be easily and automatically swapped in the scene description between renderings. Camera and motion information used in the post-processing stage must exactly match the virtual camera and motion present in the scene rendered. As a consequence, camera matching is required.

We implemented the post-processing algorithms presented below in Matlab (commercial) but re-implementing them in SciLab, Octave or other free environments would also be simple, and it would make the complete procedure free.

## 4 Reference data for stereopsis and optical flow

### 4.1 Camera model and camera matching

In the followings, $\tilde{\mathbf{M}}$ denotes the Euclidean (inhomogeneous) representation of any homogeneous point $\mathbf{M}$ and $\sim$ denote similarity (equality up to scale).

Perspective projection of any homogeneous scene point $\mathbf{M} \sim (X, Y, Z, W)^T$, $W \in \{0, 1\}$, is described as $\mathbf{m} \sim \mathbf{PM}$, where $\mathbf{m} \sim (x, y, 1)^T$ is the image of $\mathbf{M}$. $\mathbf{P}$ is a $3 \times 4$ homogeneous camera matrix that can be decomposed as $\mathbf{P} \sim \mathbf{KR}\left[\mathbf{I} - \tilde{\mathbf{C}}\right]$, where $\mathbf{C}$ is the camera center in the world reference frame, the rows of the ortonormal matrix $\mathbf{R}$ represent the camera axes in the world, and $\mathbf{K}$ is the camera calibration matrix incorporating the axis skewness $\gamma$, the principal point $(x_0, y_0)^T$ and relative focal lengths $\alpha = -f/s_x$ and $\beta = -f/s_y$, where $f$ is the focal length and the pixels are parallelograms of side lengths $s_x \times s_y$.

$\mathbf{P}' = \mathbf{KR}\left[\mathbf{I} - \tilde{\mathbf{C}}'\right]$ describes a second camera, supposing the pair $\{\mathbf{P}, \mathbf{P}'\}$ is in standard configuration. An example on how to implement this pair in the SDL of POV-Ray is shown below. It can be treated as a pseudo-code.

```
#declare CamLoc2 = CamLoc1 - Baseline*xAxis;
camera {
   perspective
   #if (currentCamera=1) location CamLoc1 #else location CamLoc2 #end
   direction FocalLength*zAxis
   right -ViewSizeHoriz*xAxis // right handed system
   up ViewSizeHoriz/ImageAspect*yAxis  }
```

A stereo pair in general configuration can be described in a very similar way. It is supposed that the elements of the rotation matrix are explicitely specified in the vectors `xAxis`, `yAxis` and `zAxis`. An alternative solution is to first describe the camera $\mathbf{P} = \mathbf{K}\left[\mathbf{I}\,\mathbf{0}\right]$ and then rotate and translate it, specifying rotation with three rotation angles, which is easier to use with motion.

Figure 1 shows a simple test scene *MarbleBoxes* rendered in POV-Ray. We will use it to validate and illustrate our approach of ground-truth data generation throughout the paper. It is very similar to the *Marble* sequence referred

in Section 2. The top-left image indicate that the POV-Ray camera is correctly encapsulated in the camera matrix **P** used in the post-processing stage.

Time-dependence of an inrinsic or extrinsic camera parameter can be described by expressing the parameter with the `clock` variable that encodes time.

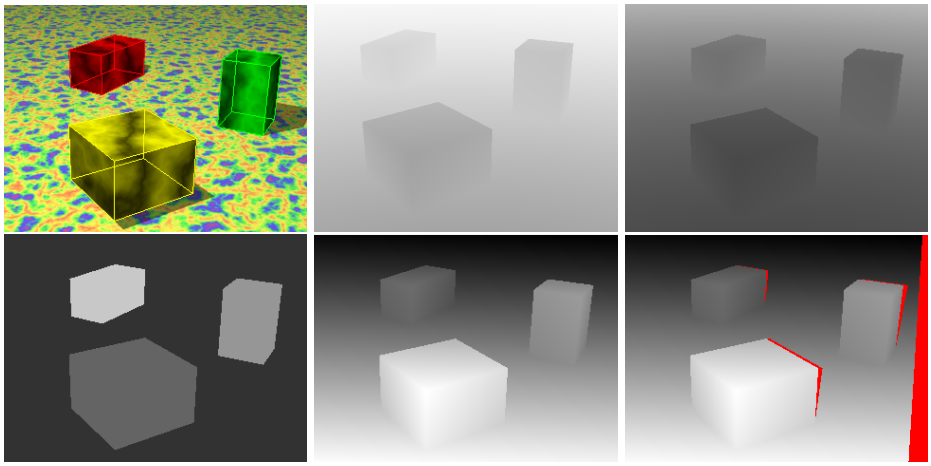

**Fig. 1.** From left to right: (a) Rendered view of the *MarbleBoxes* scene and overlayed box edges. The rasterized box edges coincide with edges projected with the camera matrix **P**: POV-Ray's camera model is correctly captured by **P**. (b-c) Untextured, foggy scene with $D = 5$ (b) and $D = 15$ (c): intensity is an exponential function of depth. (d) Labeled reference image rendered with ambient light only. Intensity-stretched dispariy map (e) without occlusion detection and (f) with occlusions.

### 4.2 Depth map generation and surface reconstruction

Currently, there is no single-call solution for directly extracting a per-pixel depth map of a scene from a viewpoint in POV-Ray (v3.6). However, as also observed in [14], depth information is encoded in the fog media. Fog renders regions whiter where depth is higher while it has less effect on the pixel colors in closer regions. More precisely (see `http://www.povray.org/documentation/`),

$$I(\lambda) = \exp\left\{-\frac{\lambda}{D}\right\} \cdot I_{trace} + \left(1 - \exp\left\{-\frac{\lambda}{D}\right\}\right) \cdot I_{fog}, \tag{1}$$

where $\lambda$ is the depth along the viewing direction, $I$ is the final intensity (or color) of a pixel, $I_{trace}$ is the intensity (or color) of the pixel computed by the raytracer without fog effect, $I_{fog}$ is the fog intensity (color) and $D$ the fog density parameter. To simplify this function, we set $I_{fog}$ as absolute white ($I_{max}$) and $I_{trace}$ as absolute black (0) by eliminating all lights and transparencies and using

an absolute diffuse black material on all scene objects. As a result

$$I(\lambda) = \left(1 - \exp\left\{-\frac{\lambda}{D}\right\}\right) \cdot I_{max} \quad \text{and} \quad \lambda = -D \ln\left(1 - \frac{I}{I_{max}}\right). \tag{2}$$

Thus, the depth $\lambda$ can be extracted by logarithmic correction. Although, [14] builds on the same idea, i.e. using a foggy scene for depth extraction in POV-Ray, a major difference is that [14] approximates a linear function of the depth with multiple fog instances as $\sum_i I_i(\lambda) \approx c \cdot \lambda$, where $c$ is a scalar, while we perform distortion correction in the post processing step to avoid approximation errors. The resulted foggy scene at different fog densities is depicted in Figure 1 and the resulted characteristics are plotted in Figure 2. Exponential compression maps

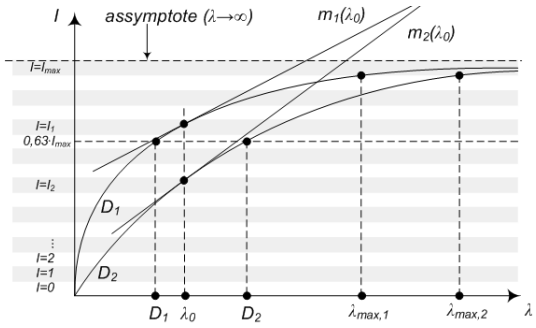


**Fig. 2.** The $I(\lambda)$ intensity vs. depth characteristics at $I_{trace} = 0$, $I_{fog} = 1$ and $D_1 < D_2$. The horizontal white or gray stripes represent a 1-LSB quantum in intensity. $\lambda_{max,i}$ is the maximum distance that can be represented due to quantization for $D_i$, $i = 1, 2$.

the depth interval $[0, \infty]$ to the valid intensity range $[0, I_{max}]$ which is useful to store the depth map in any standard file format or to display it, another reason why it is better to save the compressed exponential map first and perform corrections later. However, this introduces a quantization error that increases with distance. $I_{max}$ correspond to a depth range of $[\lambda_{max}, \infty]$. We introduce the resolution $r = I_{max}/\Delta I$ where $\Delta I$ is the quantum size. Supposing $\Delta\lambda$ is the size of the depth range corresponding to $\Delta I$, than the depth resolution is

$$\Delta\lambda(\lambda, D, r) = -D \ln\left\{1 - \frac{1}{r} \cdot \exp\left(\frac{\lambda}{D}\right)\right\}. \tag{3}$$

The resolution bounds the highest depth to represent. More precisely, $\lambda < D \ln r$. Thus, $\lambda < 5.54D$, for an 8-bit fog map and $\lambda < 11.09D$, for a 16-bit one. It is recommended to use a 16-bit grayscale map to encode depths in POV-Ray, as it supports the 48-bit (16 bits per channel) PNG format.

Note that $D$ is specified by the user. Increasing $D$ increases both $\lambda_{max}$ but decreases the depth resolution. We search for the optimal $D_{opt}$ fog density that

maximizes resolution within the valid depth range:

$$\sup\{\Delta\lambda(\lambda) \mid \lambda \in [0, \lambda_{max}]\} = \Delta\lambda(\lambda_{max}) \to \min_{D}. \tag{4}$$

Differentiating (3) and using the approximation $\ln(x) \approx x - 1$ around $x \approx 1$, one may find that $D_{opt} \approx \lambda_{max}$, provided that $D_{opt} \gg \lambda_{max}/\ln r$, that is, if $D_{opt} \gg \lambda_{max}/5.54$ for an 8-bit map and, if $D_{opt} \gg \lambda_{max}/11.09$ for a 16-bit one.

Consequently, in order to set $D_{opt}$ for optimal depth resolution, $\lambda_{max}$ must be known, however, to determine $\lambda_{max}$, the scene must be rendered with $D$ set. The suggested solution is a two pass method: render the fog map with some $D_0$, perform logarithmic correction (2) to get the depth $\lambda(x, y)$ for each pixel center $(x, y)^T$, find $\lambda_{max} = \max_{x,y} \lambda(x, y)$ and repeat rendering and correction with $D = \lambda_{max}$. Finally, the resolution error $\Delta\lambda(x, y)$ can be computed based on (3). For correct results, rendering should be done *without* antialiasing.

From known $\lambda(x, y)$ depth map, the surface point $\mathbf{M}(x, y) = (X, Y, Z, W)^T$ seen in any pixel center $\mathbf{m} = (x, y, 1)^T$ can be determined easily. The normalized viewing direction is $\mathbf{v}(x, y) = \alpha\mathbf{K}^{-1}\mathbf{m}$ with $\alpha = ||\mathbf{K}^{-1}\mathbf{m}||^{-1}$. If $\lambda(x, y) = \infty$, then the homogeneous point observed $\mathbf{M}_c(x, y) = (\mathbf{v}^T, 0)^T$ lies on the plane at infinity and $\mathbf{M}(x, y) = (\mathbf{v}^T\mathbf{R}, 0)^T$. Otherwise, the homogeneous surface point in the camera reference frame is $\mathbf{M}_c(x, y) = (\lambda\mathbf{v}^T, 1)^T$ and, in the world reference frame, it is $\mathbf{M}(x, y) = (\lambda\mathbf{v}^T\mathbf{R} + \tilde{\mathbf{C}}^T, 1)^T$.

### 4.3 Computing the disparity maps

We describe two methods we use to compute disparity maps, one for a general and one for a standard geometry. Both build on the computed depth map $\lambda(x, y)$ and on the decomposed camera matrices $\mathbf{P}$ and $\mathbf{P}'$.

Starting from a pixel center $\mathbf{m} = (x, y, 1)^T$, the 3D point $\mathbf{M}_c$ in the camera reference frame of the first view can be determined knowing $\lambda(x, y)$ and $\mathbf{P} = \mathbf{KR}\left[\mathbf{I} - \tilde{\mathbf{C}}\right]$ (see Section 4.2). Then $\mathbf{M}_c$ is reprojected to the second view as $\mathbf{m}' = (x', y', 1)^T \sim \mathbf{P}'\mathbf{M}_c$. $\mathbf{m}'$ does not necessarily fall to a pixel center. Then $d_x = x' - x$ and $d_y = y' - y$ are the horizontal and vertical disparities, respectively.

If the cameras are in a standard configuration, then all the epipolar lines and disparities are horizontal ($d_y = 0$). In such a case, a surface point at depth $Z_c$ measured along the optical axis is observed with a disparity $d_x = \alpha b/Z_c$, where $b$ is the base distance and $\alpha$ is the relative focal length in horizontal pixel sizes. This equation is valid even if there is an axis skew ($\gamma \neq 0$). With known disparity, the observed 3D point can be reconstructed at a depth $Z_c = \alpha b/d_x$. To achieve a depth resolution $\Delta Z_c$, the required resolution in disparity is [1]

$$\Delta d = \frac{\alpha \cdot b \cdot \Delta Z_c}{Z_c(Z_c - \Delta Z_c)}. \tag{5}$$

In order to determine the precision of the disparity map computed, the depth $\lambda(x, y)$ measured along the viewing ray must be converted to depths $Z_c(x, y)$, measured along the optical axis. The same applies for the resolutions $\Delta\lambda(x, y)$ and $\Delta Z_c(x, y)$. If $\mathbf{v}(x, y) = (v_x, v_y, v_z)^T \sim \mathbf{K}^{-1}(x, y, 1)^T$ is the *normalized* viewing direction, then $Z_c(x, y) = v_z(x, y)\lambda(x, y)$ and $\Delta Z_c(x, y) = v_z(x, y)\Delta\lambda(x, y)$.

### 4.4 Finding occluded areas

It is unfair to expect from a correspondence algorithm to correctly match scene points only seen by one camera but it is reasonable to require the detection of such regions. At least, we need to know which pixels are occluded in the other image, and which are not, to correctly measure the performance of an algorithm. The basic ideas of three different approches are as follows.

1. Perform a dense left-right consistency checking on the ideal per-pixel disparity maps $\mathbf{d}(x, y) = (d_x, d_y)^T$ and $\mathbf{d}'(x', y') = (d'_x, d'_y)^T$, as suggested in [15] for stereo matching algorithms based on block-matching.
2. Use rendered, labeled maps to distinguish between object projections.
3. Replace one of the cameras with a light source and re-render.

The first method starts from a pixel center $(x, y)$ in the first view. Then it maps (forward-warps) it to the second view with its corresponding precise disparity $(d_x, d_y)$. The point $(x + d_x, y + d_y)$ does not necessarily fall into a pixel center in the second view, so based on the map $\mathbf{d}'(x', y')$ in the second view, we find an approximate disparity $\hat{\mathbf{d}}'(x + d_x, y + d_y) = (\hat{d}'_x, \hat{d}'_y)^T$ for it. If $\hat{\mathbf{d}}'$ points back to the initial point $(x, y)$ in the first view, i.e. the squared Euclidean distance $c(x, y) = ||\mathbf{d} + \hat{\mathbf{d}}'||_2^2 = (d_x + \hat{d}_x)^2 + (d_y + \hat{d}_y)^2$ is small, then the disparity maps are consistent at location $(x, y)$. If the pixel $(x, y)$ is occluded than inconsistency should occur and $c(x, y)$ is expected to be high (more than 1 pixel$^2$), see the left side of Figure 3. As $\hat{\mathbf{d}}'$ is an approximation, full $c(x, y) = 0$ consistency is not expected. A user-specified threshold is needed, which is a disatvantage.

It seems straightforward to interpolate the unknown disparity $\hat{\mathbf{d}}'$ at location $(x + d_x, y + d_y)$ from known disparities $\mathbf{d}'(x', y')$ of the four neighbouring pixel centers, but this is likely to fail at object boundaries where some of the four pixels belong to a closer object while others to a further one. See center of Figure 3: if one could detect which of the neighbouring pixel centers (black dots) fall on the same side of the boundary line as the point $(x + d_x, y + d_y)$ (depicted by $\times$ in the figure), we could avoid this problem by using the center-disparities for those pixels at interpolation. From the available data, there is no way to tell this, although an approximation of the boundary may be found with active contours. Very similarily, by using a higher resolution of the disparity maps boundary localization may be better, which decreases the likelihood of false detections.

The second method determines whether the same object is seen at $(x, y)$ in the first view and at $(x + d_x, y + d_y)$ in the second using labeled maps for both views. We render the label maps in POV-Ray for the same scene by automatically eliminating all lights, removing textures and assigning purely ambient maps with a distinct gray value for each object (see Figure 1). These predefined values act like object identifiers. As depicted on the right of Figure 3, finite resolution of the labeled images is expected to cause, again, problems at object boundaries. The partial solutions for the first method apply here, as well. The second method has the advantage that it does not require a user-defined threshold for binarization, it directly produces binary maps. A price payed is that labeled maps are required.

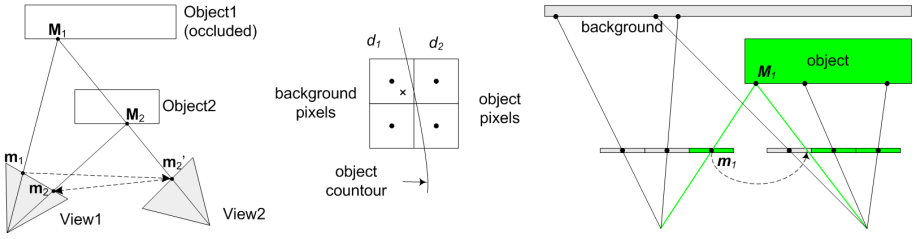


**Fig. 3.** Occlusion detection with left-right consistency checking: the point $\mathbf{M}_2$ on *Object2* occludes $\mathbf{M}_1$ on *Object1*, so based on the ideal disparity maps, $\mathbf{m}_1 \mapsto \mathbf{m}_1' \equiv \mathbf{m}_2'$ in *View2* but $\mathbf{m}_2' \mapsto \mathbf{m}_2 \not\equiv \mathbf{m}_1$ in *View1* (left). Problem with finite resolution at consistency checking (center). Similar problem with the label-based method (right).

The third method consists of re-rendering the scene from one view by replacing the other camera by a white intense light source, removing all other lights and setting all material to diffuse white. Since the surfaces do not illuminate each other by default, light renders ideal white each surface point seen by the second view and ideal black shadows represent occluded areas. Because the renderer starts by tracing a ray through a pixel center and it then checks whether the surface point hit by the ray is illuminated from the light source, we do not expect any rounding artifact around object borders or at depth discontinuities. However, the light source does not model the finite rectanglar viewing window of the camera, i.e. it radiates in all directions. Therefore, we need to correct the occlusion map to include out-of-view points (red region along the right edge in Figure 1f). This is done in a post-processing step by checking, for each pixel center $(x, y)$, whether $(x + d_x, y + d_y)$ falls within the second view.

In POV-Ray (v3.6), shadows are never computed for the points at infinity, neither when the `sky_sphere` feature is on. Thus, projections of points at infinity, if there are any, are never detected as occluded. A solution is to use a normal, shadow-capable sphere with finite but huge radius $\rho$ encapsulating all scene objects. In Figure 4, the finite point $\mathbf{M}$ occludes $\mathbf{M}_\infty$. $\mathbf{m}_\infty$, that is the image of $\mathbf{M}_\infty$, should be detected as occluded. $\mathbf{M}_\infty$ can not receive shadows in POV-Ray, so $\mathbf{M}_s$ is shadowed on the inner surface of the introduced sphere, instead. This causes a shift from the correct point $\mathbf{m}_\infty$ to the approximation $\mathbf{m}_s$. The approximation error can be expressed with the euclidean distance $d_E(\mathbf{m}_s, \mathbf{m}_\infty)$, as follows. Suppose $\mathbf{m}_s$ is the center of a rendered pixel and $\mathbf{P} = \mathbf{KR}\left[\mathbf{I}\ \mathbf{0}\right]$ is the camera matrix. Then the viewing direction of $\mathbf{m}_s$ is $\mathbf{d}_s = \mathbf{R}^T\mathbf{K}^{-1}\mathbf{m}_s$. $\mathbf{M}_s$ lies both on the viewing ray $\tilde{\mathbf{C}} + \alpha\mathbf{d}_s$, where $\alpha \geq 0$ is a scalar, and on the sphere, satisfying $||\tilde{\mathbf{M}}_s(\alpha) - \tilde{\mathbf{L}}||^2 = \rho^2$. Supposing that $\mathbf{C}$ falls inside the sphere, this equation has only one positive solution $\alpha = \alpha_s$. This determines the point $\mathbf{M}_s$. This, just like $\mathbf{M}_\infty$ behind it, falls in the direction $\mathbf{d}_\infty = \tilde{\mathbf{M}}_s - \tilde{\mathbf{L}}$ from the light source located at $\mathbf{L}$. $\mathbf{M}_\infty$ is seen in the same direction $\mathbf{d}_\infty$ from the camera center, so it can be expressed as $\mathbf{M}_\infty \sim (\mathbf{d}_\infty^T, 0)^T$. Its image is $\mathbf{m}_\infty \sim \mathbf{PM}_\infty = \mathbf{KRd}_\infty = \mathbf{KR}(\tilde{\mathbf{C}} - \tilde{\mathbf{L}}) + \alpha_s\mathbf{m}_s$. Finally, the appoximation error is $d_E(\mathbf{m}_s, \mathbf{m}_\infty)$ for any pixel center $\mathbf{m}_s = (x, y, 1)^T$. The error is small

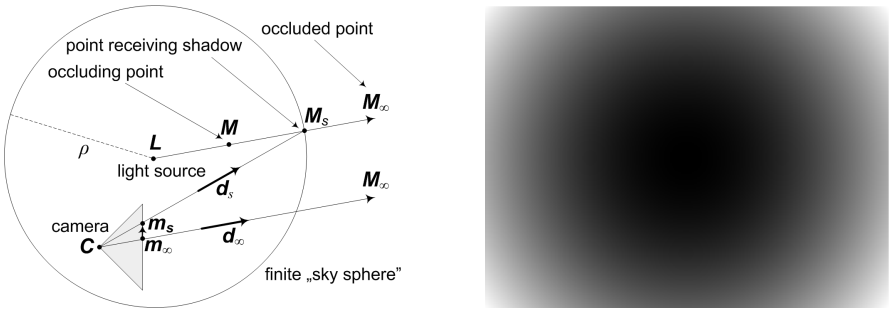


**Fig. 4.** Computing the error of occlusion estimation with the third method when a finite sky sphere is used for enforcing shadows on "far enough" points (left). Error map for *MarbleBoxes* (right). Brighter means larger errors.

in directions nearly perpendicular to the baseline and increases radially (see Figure 4). Errors decrease as the sphere radius $\rho$ increases.

### 4.5 Determining the motion field

We aim to compute an ideal motion field, supposed a depth map and a labeled map are given for each frame of an image sequence and object motions are rigid and known. See Sections 4.2 and 4.4 on how to get depth and labeled maps.

The motion of $n$ (potentially) moving objects in the scene is described by the instantaneous pose of the objects, that is, by the parameter set $\{\mathbf{R}^i(t), \mathbf{t}^i(t)\}_{i=1}^n$, $j$ being the object index, $\mathbf{t}^i(t)$ the location of the object in the virtual world at time instance $t$ and the rows of $\mathbf{R}^i(t)$ defining normalized axis directions of the local (object) reference frame. Similarily, the instantaneous camera pose is defined by $\{\mathbf{R}(t), \mathbf{t}(t)\}$, $\mathbf{t}$ being the camera center denoted by $\tilde{\mathbf{C}}$ earlier. All these parameters and their time-derivatives are supposed to be analytically known, which is natural, since we design the virtual world, including the motion.

Consider any pixel center $(x, y)$ and suppose that the $i$-th object is seen at finite depth $\lambda$ through it. The object identifier $i$ and the depth $\lambda$ can be read out from the labeled and depth maps, respectively. By using the camera parameters, the observed surface point $\mathbf{c}^i(t) \in \mathcal{R}^3$ can be determined in the camera reference frame, as detailed in Section 4.2. If the same point in the object's local reference frame is $\mathbf{p}^i(t)$, then it can be expressed in the camera reference frame in the form $\mathbf{c}^i(t) = \mathbf{Q}^i(t)^T\mathbf{p}^i(t)+\mathbf{q}^i(t)$, where $\mathbf{Q}^i(t)$ and $\mathbf{q}^i(t)$ and their derivatives are analytically known from the above-mentioned camera and object motion parameters. Exploiting the rigidity constraint $\mathbf{p}^i(t) = const = \mathbf{p}^i$, the instantaneous spatial velocity of the point in the camera reference frame is

$$\mathbf{D}(t) = \frac{d\mathbf{c}^i(t)}{dt} = \left(\frac{d}{dt}\mathbf{Q}^i(t)\right)^T \mathbf{Q}^i(t)[\mathbf{c}^i(t) - \mathbf{q}^i(t)] + \frac{d\mathbf{q}^i(t)}{dt}. \tag{6}$$

If $\Delta t$ is the time between frames and $k$ is the frame index, then the point in continuous motion is sampled as $\mathbf{c}_k^i = \mathbf{c}^i(k\Delta t)$, while its image is $\mathbf{m}_k^i =$

$(x_k^i, y_k^i, 1)^T \sim \mathbf{K}\mathbf{c}_k^i$. Define the 2D projection of the sampled instantaneous velocity $\mathbf{D}_k = \mathbf{D}(k\Delta t)$ as $\mathbf{v}_k = (v_{xk}, v_{yk})^T$, the 2D Forward Difference Motion (FDM) as $\mathbf{\Delta}_{Fk} = \tilde{\mathbf{m}}_{k+1} - \tilde{\mathbf{m}}_k$ and the 2D Backward Difference Motion (BDM) as $\mathbf{\Delta}_{Bk} = \tilde{\mathbf{m}}_k - \tilde{\mathbf{m}}_{k-1}$. It depends on the purpose of the optical flow algorithm whether the ground-truth instantaneous velocity $\mathbf{v}_k(x, y)$, the FDM $\mathbf{\Delta}_{Fk}(x, y)$ or the BDM $\mathbf{\Delta}_{Bk}(x, y)$ should be used as a reference for evaluation. The proposed framework can generate all three, though, computation of the FDM and BDM are easier to implement, since they do not require parameter derivatives.

## 5 Results

For the *MarbleBoxes* scene, it is not difficult to implement a double-precision depth computation *without* using POV-Ray. Such a $\bar{\lambda}(x, y)$ depth map can be used to validate our $\lambda(x, y)$ depth map resulted from the procedure in Section 4.2. Hence, we implemented a basic (non-recursive) raytracing algorithm in Matlab that determines depths for a planar scene, as well as, for the rectangular faces of any given box. For each pixel, we chose the closest intersection of each ray with the scene, similarily to the Z-buffer algorithm. Figure 5 shows a comparison of the difference $I(\bar{\lambda}) - I(\lambda)$, where $I(\xi)$ denotes the exponential function (2) of $\xi$. The experiment has been performed with a 16-bit fog map, with the fog density parameter $D = 10$. This is not optimal, since it differs from the maximum scene depth of 19.501 units (see Section 4.2). The error histogram clearly indicates a uniform distribution, explained by the quantization error in the POV-Ray fog map $I(\lambda)$. Interestingly, comparing the truncated exponential function $\lfloor I(\bar{\lambda}) \rfloor$ of the reference depth map, with $I(\lambda)$, we still find 22+33 out of $320 \times 240$ pixels with non-zero depth errors (errors are of $\pm 1$ LSB exactly). These are randomly spread over the image and may be due to numerical errors in intersection computations.

For direct comparison of the depths with no exponential distortion, the theoretical depth resolution $\max_{x,y} \Delta\lambda(D)$ has been computed based on (3). Then several pairs of fog maps have been generated in POV-Ray with different fog densities $D_i$ and the maximum depth error $\max_{x,y}\{\lambda(x, y) - \bar{\lambda}(x, y)\}$ is computed for each pair. This experimental error is near to but always less than the resolution predicted from theory (see plot in Figure 5). This is confirmed for all the pixels of the image, except for exactly the same 22 misbehaving pixels that had a -1 LSB error in the fog map. The depth error at these locations exceeded the depth resolution with a neglectable amount, only up to the fourth significant digit (in the error of the error). Thus, the worst-case error prediction is proven to be good at least up to the third significant digit. The smallest worst-case error is experimented around $D = D_{opt}$ being approximately equal to the maximum scene depth (of 19.501 units here), just as predicted in Section 4.2.

A comparison of the two disparity map computation methods yields similar results. For the standard arrangement of *MarbleBoxes*, we computed the disparity maps $\bar{d}_x^g(x, y)$ and $\bar{d}_y^g(x, y)$ with the general method from the double-precision Matlab depth map $\bar{\lambda}(x, y)$, on the one hand, and the disparity $d_x^s(x, y)$ with its

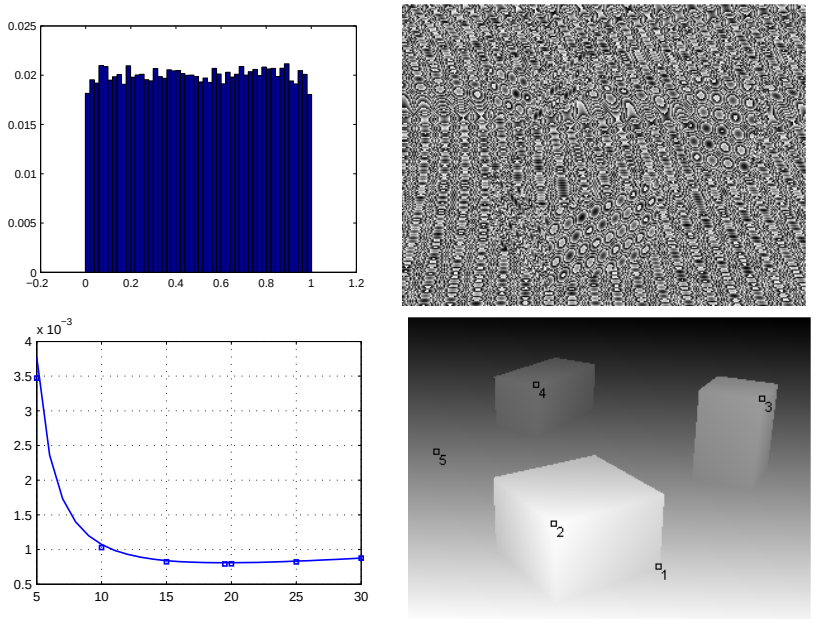


**Fig. 5.** From left to right:(a) 50-bin error histogram and (b) grayscale error map of the computed depth map for *MarbleBoxes*. White corresponds to an error of 1 LSB, black to no error. (c) Theoretical $\max_{x,y} \Delta\lambda(D)$ resolution error function is plotted vs. the fog density parameter $D$ (continuous curve). The maximum errors evaluated at several $D$ values experimentally are plotted as squares. Note that $D_{opt} \approx 19.5$, as predicted. (d) Stretched horizontal disparity map and some points with results detailed below.

resolution $\Delta d_x^s(x, y)$ from the POV-Ray data, assuming standard geometry, on the other hand (the upper indices stand for *g*eneral and *s*tandard method). We experimented that $\bar{d}_y^g(x, y)$ is zero up to numerical precision (13 digits) and the error $e(x, y) = d_x^s(x, y) - \bar{d}_y^g(x, y)$ is always less than the resolution $\Delta d_x^s(x, y)$, except for the 22 misbehaving pixels with inferior numerical errors. In order to illustrate this validation, we selected some pixels in the first image (see Figure 5) and summarised the results for them in Table 1. Due to the quantization error of the 16-bit POV-Ray fog map, the disparity values computed by the proposed method are not double-precision but have a precision of about $10^{-3}$ pixels. Comparing this to the highest accuracy of about 0.1 pixels of known matching algorithms, we can conclude that the proposed method produces disparity maps that are accurate enough to validate the best stereo algorithms known today.

We tested the three proposed occlusion detection methods with *MarbleBoxes* and experienced false detections along box edges, at depth discontinuities in the first two cases, as expected from theory. The third method is free from such artifacts (see Figure 1f for a result).

We validated our optical flow computation methods, as well, both for stationary and moving camera. Due to space limitations and for clairity, we only present

**Table 1.** A comparison of the horizontal disparities (in pixels) with the error-free values for some selected pixel centers depicted in Figure 5. See the text for the explanation of the symbols. Errors are all below the corresponding resolution error.

| # | $\bar{d}_x^g$ | $d_x^s$ | $d_x^s - \bar{d}_x^g$ | $\Delta d_x^s$ |
|---|---|---|---|---|
| 1 | 18.29436 | 18.29556 | 0.00120 | 0.00125 |
| 2 | 21.17369 | 21.17419 | 0.00051 | 0.00161 |
| 3 | 15.15390 | 15.15447 | 0.00058 | 0.00090 |
| 4 | 12.35276 | 12.35331 | 0.00055 | 0.00066 |
| 5 | 12.87468 | 12.87477 | 0.00009 | 0.00069 |

the stationary camera and backward-difference motion results here. Three consecutive images and the computed BDM is depicted in Figure 6. We recall that the depth and segmentation maps are also needed for each image. For a rough validation, we computed the double-precision motion vectors for the box corners and took the min. and max. vector length for each box. Then we compared the minimum and maximum BDM computed for the pixel centers within the image of each box (see Table 2). Subpixel differences in the lengths come from the fact that box corners do not fall to pixel centers, while the large difference in the minimum vector lengths for the green box is due to a whirpool with a center of zero motion *within* the image of the box, far from the box corners.

**Table 2.** A rough validation of the BDM computed for the 2nd and 3d images of the sequence in Figure 6. For each box, the min/max vector lengths of the double-precision motion vectors of the box corners and those of the per-pixel BDM are shown.

| Object | Corners | | Pixels | |
|---|---|---|---|---|
| | Min | Max | Min | Max |
| yellow box | 26.90 | 36.42 | 26.86 | 36.42 |
| green box | 12.41 | 69.19 | 0.13 | 69.13 |
| red box | 0.00 | 94.47 | 0.60 | 94.44 |

Reference data for some more complex scenes is shown in Figure 7. In the *Saint* pair, the sky is mapped on a hollow sphere of radius $\rho = 10^6$. The corresponding displacement errors at occlusion map generation (see Section 4.4) is radial, similarily to that shown in Figure 4. The error is in the $[0.0012, 0.0013]$ pixels range which is comparable to the accuracy of the disparity maps.

The *Workdesk* pair competes with the widely used real-world *Tsukuba* pair in complexity. While the *Tsukuba* dataset contains hand-labeled reference disparities only up to pixel precision, with the automatically rendered *Workdesk* dataset, one can discriminate among the best algorithms of sub-pixel accuracy.

## 6 Conclusions

A review of existing datasets for evaluating stereo vision and optical flow algorithms has been presented. We conclude that quantitative approches have started to gather ground but few research groups provide accurate ground-truth

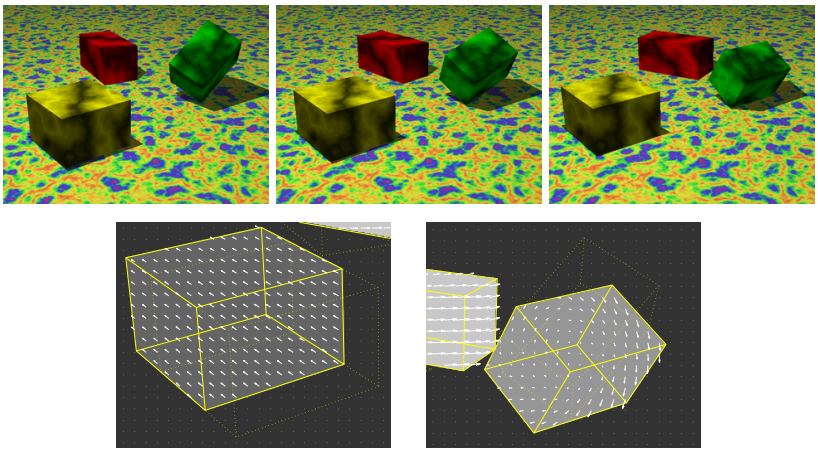

**Fig. 6.** Three consecutive images of a motion sequence of the *MarbleBoxes* scene and two details of the backward-difference motion field computed for the 2nd and 3d images. The camera is stationary while the boxes are translating and rotating.

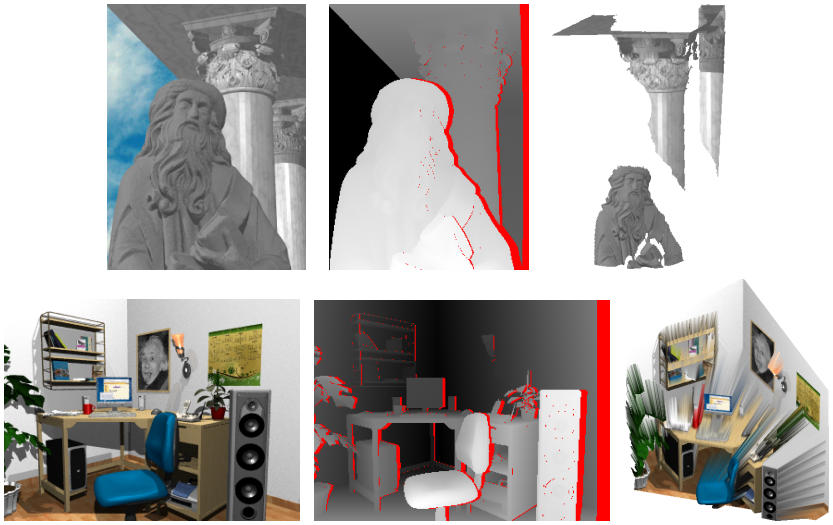

**Fig. 7.** From left to right: (a) Right image of the *Saint* pair, (b) the corresponding stretched reference horizontal disparity map with occluded pixels (in red). Disparities range from 0 to 20.5 pixels. (c) *Saint* scene reconstructed from the depth map. (d) Right image of the synthetic *Workdesk* pair competing the real-world *Tsukuba* set in complexity, (e) its stretched ground-truth disparity map with occlusions and (f) a reconstruction without hidden face removal. Disparities range from 8.2 to 21.3 pixels.

evaluation data besides the input images. Motivated by this fact, we presented a two-step method for generating this kind of data for virtual scenes of any complexity (in fact, up to the designer's patience). In the first stage we make use of the POV-Ray renderer to produce the original images (input for the al-

gorithms to test) and we also render the scene with special modifications to textures and illuminations. We recall that all these modifications can be made automatically. In the second stage, we apply special post-processing algorithms to produce high-precision ground truth depth maps, disparity maps, occlusion maps and 2D motion fields, including the 2D projections of 3D motion, e.g. useful in robotics, and the forward/backward difference motion, e.g. for video processing algorithms. After a validation of the method, we presented some photo-real close-range reference images (the *Saint* and the *Workdesk* set) to work with.

Synthetic data seems to be inferior compared to real images, but the most important reason for this is the lack of modeled radiosity, supposed that the scene is engineered carefully for photo-reality. However, we must emphasize that POV-Ray can locally model radiosity for the price of rendering time.

In the future, we plan to generate sophisticated, photo-real outdoor, long-range reference data with the proposed method and evaluate some of the best known stereo and optical flow algorithms based on the generated data.

## References


1. M. Z. Brown, D. Burschka, and G. D. Hager, "Advances in computational stereo," *PAMI*, 25(8):993–1008, Aug 2003.
2. D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *IJCV*, 47(1):7–42, 2002.
3. R. Szeliski and D. Scharstein, "An experimental comparison of stereo algorithms," in *International Workshop on Vision Algorithms*, pp. 1–19, 1999.
4. J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Performance of optical flow techniques," *IJCV*, 12(1):43–77, 1994.
5. S. Baker et al., "A database and evaluation methodology for optical flow (MSR-TR-2009-179)," Tech. Rep., Microsoft Research, Redmond, December 2009.
6. D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *CVPR*, pp. 195–202, 2003.
7. Y. Nakamura et al., "Occlusion detectable stereo - occlusion patterns in camera matrix," in *CVPR*, pp. 371–378, 1996.
8. P. J. Besl, "Active, optical range imaging sensors," *Machine Vision and Applications*, 1:127–152, 1988.
9. J. Batlle, E. Mouaddib, and J. Salvi, "Recent progress in coded structured light as a technique to solve the correspondence problem: a survey," *Pattern Recognition*, 31(7):963–982, 1998.
10. S. Baker, R. Szeliski, and P. Anandan, "A layered approach to stereo reconstruction," in *CVPR*, pp. 434–441, 1998.
11. M. Bertozzi et al., "Obstacle detection and classification fusing radar and vision," in *Intelligent Vehicles Symposium*, pp. 608–613, 2008.
12. R. Szeliski, "Prediction error as a quality metric for motion and stereo," in *ICCV*, pp. 781–788, 1999.
13. S. Morales, T. Vaudrey, and R. Klette, "A third eye for performance evaluation in stereo sequence analysis," in *CAIP'09*, pp. 1078–1086, 2009.
14. D. Mackay, "Generating synthetic stereo pairs and a depth map with POV-Ray (TM 2006-197)," Tech. Rep., Defence Research and Development Canada, 2006.
15. P. Fua, "A parallel stereo algorithm that produces dense depth maps and preserves image features," *Machine Vision and Applications*, 6:35–49, 1993.