Projection Distortion-based Object Tracking in Shader Lamp Scenarios

Niklas Gard, *Student Member, IEEE*, Anna Hilsmann and Peter Eisert, *Member, IEEE*

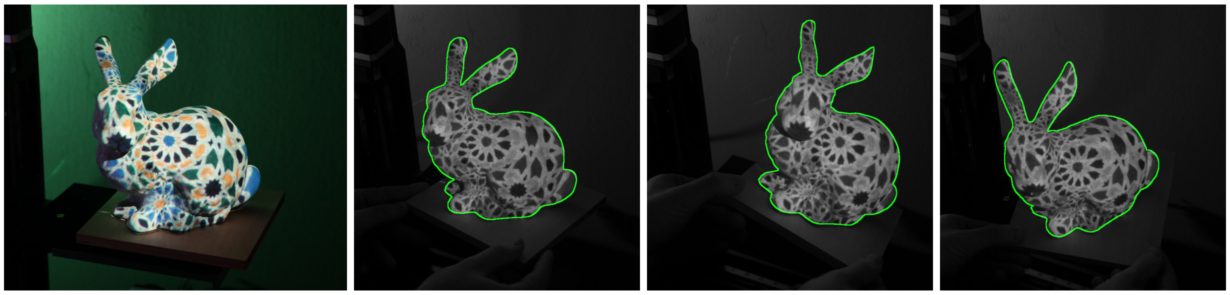

Fig. 1. The object's appearance is manipulated by projecting a new texture on its surface (left). The projector-camera system captures the moving object and estimates the object pose (highlighted by green contour) from projection distortions. The projected image can be corrected to generate a matching projection again.

Abstract— Shader lamp systems augment the real environment by projecting new textures on known target geometries. In dynamic scenes, object tracking maintains the illusion if the physical and virtual objects are well aligned. However, traditional trackers based on texture or contour information are often distracted by the projected content and tend to fail. In this paper, we present a model-based tracking strategy, which directly takes advantage from the projected content for pose estimation in a projector-camera system. An iterative pose estimation algorithm captures and exploits visible distortions caused by object movements. In a closed-loop, the corrected pose allows the update of the projection for the subsequent frame. Synthetic frames simulating the projection on the model are rendered and an optical flow-based method minimizes the difference between edges of the rendered and the camera image. Since the thresholds automatically adapt to the synthetic image, a complicated radiometric calibration can be avoided. The pixel-wise linear optimization is designed to be easily implemented on the GPU. Our approach can be combined with a regular contour-based tracker and is transferable to other problems, like the estimation of the extrinsic pose between projector and camera. We evaluate our procedure with real and synthetic images and obtain very precise registration results.

Index Terms—Projector-camera systems, projector-camera calibration, shader lamp systems, object tracking, object registration, spatial augmented reality, projection mapping.

1 Introduction

Projector-camera systems are a useful tool to perform projection mapping and to present information directly matched with the local environment. This is beneficial in different contexts for example to allow robots to project their intentions spatially correct into working environments [1], to enhance medical procedures by projecting aditional information [6, 7], or to improve interaction with objects with spatial user interfaces [16] e.g. in industrial visual inspection scenarios (Fig. 2). In contrast to glasses-based systems, multiple people can see the projection at once without the need for a wearable hardware. The spreading of mobile projector systems has been increasing over the last years and currently pico projectors are even integrated into mobile phones. Laser projectors or LCoS projectors with laser light source overcome the limitation of depth of field and generate a *focus free* image [10]. Also 3D printers and low cost 3D scanners gained popularity and allow digitalization of objects and reproduce mockups, which can be used for spatial AR applications in the creative industry by e.g. designers or architects [19]. For the perfect illusion that the projection sticks to the surface, a precise alignment between real and virtual objects is extremely important. Hence, in dynamic scenes, an accurate 6D pose tracking is needed and the generated image has to be updated accordingly. Usually if a naturally textured object is moved by a specific transformation, the same transformation is applied on all its texture points and is visible in the next camera frame. However, the projected texture underlies a certain delay, since it slides over the object surface, when the object is moved. This generates a perspective distortion, which contradicts with the assumptions made by feature or contour-based object trackers and prevents correct pose estimation.

Our tracking system directly integrates this observation in the pose estimation and does not need extra hardware besides a camera and a projector. We follow the principle of model-based analysis-by-synthesis and generate a digital image of the object from the point of view of the camera. The projector is simulated by projective texture mapping [8], allowing a simulation of the movement of the object *under* the projection. Our main contribution is the derivation of a motion model describing the relation between the movement of a projected point on the image sensor and the movement of the object in 3D space. We embed the model in a robust pose estimation system and present solutions to make the virtual and the real image comparable. Iteratively the object pose is optimized to minimize the difference between the real observation and rendered expectation. Distortions of the projection are thereby used for tracking and compensated at the same time. Fig. 1 shows that our method can augment a dynamically moving object with a shader lamp projection by using exclusively the image of the projection captured with a normal camera. In order to optimize the object pose a very precise projector-camera calibration is needed. As an extension of our method, a new formulation of the optical flow model

• *Niklas Gard, Anna Hilsmann and Peter Eisert are with Fraunhofer HHI. E-mail: {niklas.gard | anna.hilsmann | peter.eisert}@hhi.fraunhofer.de.*
• *Niklas Gard and Peter Eisert are with Humboldt University of Berlin.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

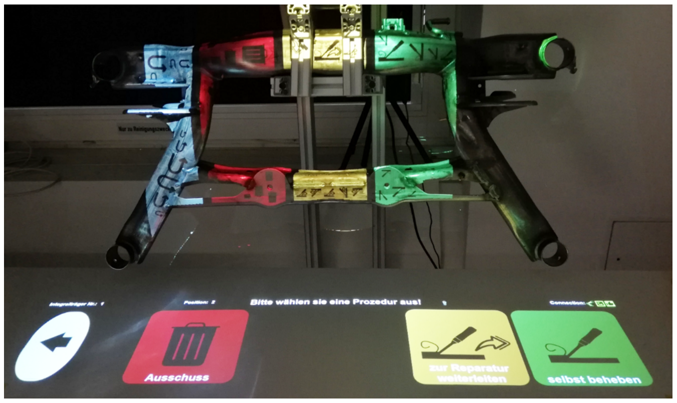
Bitte wählen sie eine Prozedur aus!
Ausschuss
zur Reparatur weiterleiten
selbst beheben

Fig. 2. Example application of projection mapping in an industrial scenario. The menu structure on the table is duplicated on the surface of the workpiece. By using the projection distortion-based tracking the projection can be used for object tracking.

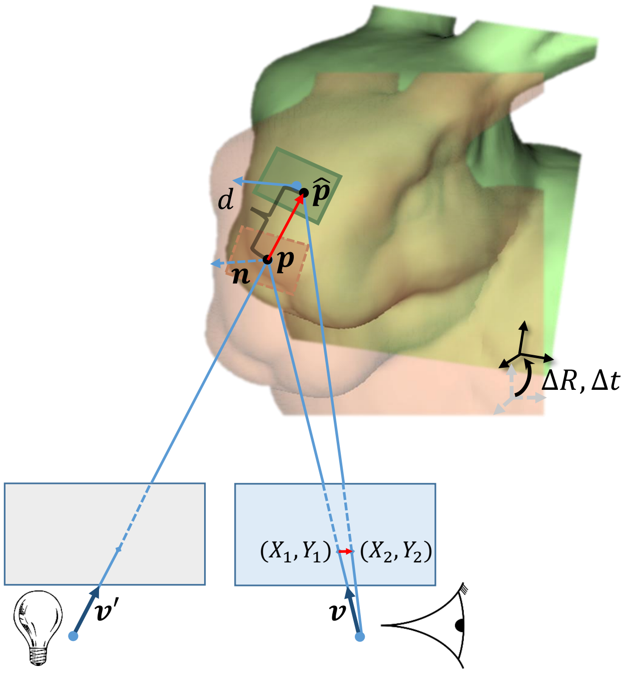

Fig. 3. Geometry of a projector-camera system. The expected object pose (orange) differs from the observed object pose (green). The transformation ($\Delta\mathbf{R}$, $\Delta\mathbf{t}$) is estimated from the displacement in image space, which is related to $\mathbf{v}'$ and d.

allows to optimize the pose of the projector relative to the camera, given a known object pose. Only small modifications of the previously described formulation are needed to do this, with the effect, that a calibration refinement of projector-camera systems can be substantially simplified. Furthermore, the accuracy of the projection on real-world objects is maximized.

The remainder of this paper is structured as follows. In Chapter 2 the principle is put into context of the current state-of-the-art. Chapter 3 deals with the mathematical background and describes the implementation of our approach. Chapter 4 describes the extended calibration as well as a possible combination with a traditional contour-based tracker. To evaluate the object tracking in Chapter 5, we use a simulation with synthetic images, but also test the pose estimation for real sequences. A motorized linear translation stage allows to create a valid ground truth movement.

2 Related Work

In this paper, we follow the shader lamps paradigm by Raskar et al. [22]. The appearance of a real object, with known 3D geometry, is manipulated by rendering a synthetic image of the object from the point of view of the projector. Using shader lamps in dynamic environments and the related sensing and tracking has been addressed in different ways in literature over the last years but still faces several challenges.

Raskar et al. [21] use fiducial markers to detect the 3D pose of objects with a projector-camera system and to dynamically adapt the projection with respect to the object geometry. Also, recent approaches [2, 18] make use of fiducials printed directly on the object surface. While the markers are optimized for high-speed tracking [18] and strategies are presented to diminish their visibility for the human eye with the projector [2], every object has to be carefully designed and prepared to be suitable for tracking.

Hashimoto et al. [11] make the projection invisible for the camera sensor by capturing with an infrared (IR) camera. In this way, it is possible for them to use a classical tracking approach, which is not irritated by the projection. Also, the dynamic face projection system of Bermano et al. [5] uses an IR image to track facial movements. Beside the need for a special hardware, in spatial augmented reality (AR) scenarios sensors working with short-wave illumination often establish gesture and person recognition. This leads to an interference in the image of the IR-camera, which can potentially disturb the tracking.

In contrast to this, other approaches [14, 15] use depth sensors, which are not distracted by a projection in the visible spectrum and estimate the object pose by aligning the object directly to the captured pointcloud with the iterative closest point (ICP) algorithm and a particle filter. Because of the limited resolution of those sensors, the accuracy of the alignment between the real object and the projection is reduced compared to an image-based approach [11].

Closed-loop approaches capture the projection with a camera and correct the projection in order to improve the matching between projected content and real world. Audet et al. [3] introduce to use the projection for image alignment by modelling the light emitted by the projector and reflected into the camera. Zheng et al. [32] elaborate on the closed-loop concept and extend it to different AR paradigms, but as well as Audet et al. they do the alignment only for 2D targets.

Resch et al. [23] present a closed-loop tracking approach for 3D objects. They use discrete feature point correspondences between the projector image and the captured camera image. The 2D-2D features are triangulated to reconstruct 3D points, which are then registered to the point cloud of the reference model. To allow the optimization algorithm to converge stably, a large number of feature point correspondences is needed.

In this work, we also focus on the task of estimating a 3D pose of an object in a closed-loop fashion, but combine it with the alignment methods as given by Audet et al. [3]. We follow an analysis-by-synthesis approach and use synthetically generated images of the projection on the reference surface [9]. We iteratively correct the object pose to simulate the appearance of the mismatched projection with the renderer, which at the same time leads to the observed pose offset. Compared to the approach of Resch et al. [23], our approach does not rely on point feature correspondence but instead minimizes the image difference between simulation and camera image. A radiometric calibration is omitted by the usage of edge images. Since we do not have to transform the model to a point cloud, but instead use the rendered images directly, the density of the mesh sampling does not limit the distance between target object and camera.

3 Estimation of Object movement

The motion model describes how an image point projected onto a known 3D object moves in the image space of a camera if the object changes its 3D pose. This results in six degrees of freedom to be estimated, three for the translation and three for the rotational offset.

3.1 Motion Model

We assume that the projector is oriented towards an arbitrary object and that the intrinsic parameters of the projector and the camera and the extrinsic parameters between the devices are known. The pinhole camera model describes the camera as well as the projector, which is seen as an inverted camera. Each point $\mathbf{p} = [p_x, p_y, p_z]^T$ on the object surface and the corresponding surface normal $\mathbf{n}$ describe a plane. The following equations assume that this plane approximates the local environment of the point, meaning that the model is valid on locally smooth surface patches and special treatment is needed at steep edges.

For each pixel (X_1, Y_1) in the camera image, a viewing ray is defined by the direction

$$\mathbf{v} = [-\frac{X_1 - u_x}{f_x}, \frac{Y_1 - u_y}{f_y}, -1]^T \tag{1}$$

with f_x, f_y being the scaled horizontal and vertical focal length and u_x, u_y the pixel position of the principal point. The extrinsic parameters of the projector-camera system transform the point to the projector coordinate system, where it is back projected to the image plane of the projector leading to the viewing direction $\mathbf{v}'$ of the projector.
Assuming that the movement of the object between two successive video frames is small, the change of orientation can be approximated by a linearized rotation matrix

$$\Delta\mathbf{R} = \begin{pmatrix} 1 & -\Delta r_z & \Delta r_y \\ \Delta r_z & 1 & -\Delta r_x \\ -\Delta r_y & \Delta r_x & 1 \end{pmatrix} \tag{2}$$

with three degrees of freedom. The offset $\Delta\mathbf{t}$ of the object center $\mathbf{t}$, describes the change of object translation. This results in a movement of the projection on $\mathbf{p}$ by the distance d in direction of the projector's light beam. The ray will intersect the plane through the moved point at

$$\hat{\mathbf{p}} = \mathbf{p} + \mathbf{v}'d \tag{3}$$

with $\mathbf{v}'$ being the view vector of the projector in the camera coordinate system. In general, a plane is defined by all 3D points $\mathbf{x}$ fulfilling the equation $(\mathbf{x} - \mathbf{p_0}) \cdot \mathbf{n} = 0$, where $\mathbf{p_0}$ is an arbitrary point on the plane. If a 3D transformation with $\Delta\mathbf{t}$, $\Delta\mathbf{R}$ is applied on the object, we get

$$(\mathbf{x} - (\Delta\mathbf{R}(\mathbf{p} - \mathbf{t}) + \mathbf{t} + \Delta\mathbf{t}))^T \Delta\mathbf{R}\mathbf{n} = 0 \tag{4}$$

To find the intersection with the transformed plane and the viewing ray, we set $\mathbf{x} = \hat{\mathbf{p}}$ in Eq. (4). In the next step, we solve the equation for d.

$$d = \frac{(\Delta\mathbf{R}(\mathbf{p} - \mathbf{t}) + \Delta\mathbf{t} - (\mathbf{p} - \mathbf{t}))^T \Delta\mathbf{R}\mathbf{n}}{\mathbf{v}'^T \Delta\mathbf{R}\mathbf{n}} \tag{5}$$

Given $\hat{\mathbf{p}}$ from Eq. (3), a new pixel coordinate (X_2, Y_2) of the point in the camera image is calculated by

$$X_2 = -f_x \frac{\hat{p}_x}{\hat{p}_z} + u_x \quad Y_2 = f_y \frac{\hat{p}_y}{\hat{p}_z} + u_y \tag{6}$$

if the position of the object is translated with $\Delta\mathbf{t}$ and rotated with $\Delta\mathbf{R}$. Fig. 3 visualizes the model. The object movement has the effect, that a projector ray intended to hit the object at $\mathbf{p}$ meets it approximately at $\hat{\mathbf{p}}$, which is directly related to the movement of the point in image space.

3.2 Flow-based Pose Estimation

In this section, the motion of the object, whose 3D shape and normals are known by a mesh or CAD-model, is estimated by evaluating spatial and temporal image gradients. This leads to a set of linear equations [26] to calculate the pose offset values $\Delta\mathbf{t}$, $\Delta\mathbf{r}$ from a known initial pose. The extension of optical flow for a projector has been described as illumination flow before [29], but has not been explicitly solved for the pose parameters.

The image I_1 of the actual projection is captured by the camera. Image $\hat{I}_1$ of the projection on a known initial object pose is either the previous image in a tracking sequence or synthetically generated. A motion compensation minimizes the displacement error $(u_m, v_m) = (X_1 - X_2, Y_1 - Y_2)$ between $\hat{I}_1$ and I_1. By using the first order Taylor expansion, the projection of a point to the sensor can be expressed as

$$\begin{aligned} \frac{\hat{p}_x}{\hat{p}_z} &\approx \frac{p_x}{p_z} + \frac{b}{p_z(\mathbf{n}^T\mathbf{v}')}(\Delta\mathbf{r}^T\mathbf{h} + \Delta\mathbf{t}^T\mathbf{n}) \\ \frac{\hat{p}_y}{\hat{p}_z} &\approx \frac{p_y}{p_z} + \frac{c}{p_z(\mathbf{n}^T\mathbf{v}')}(\Delta\mathbf{r}^T\mathbf{h} + \Delta\mathbf{t}^T\mathbf{n}) \end{aligned} \tag{7}$$

with

$$b = v'_x - v'_z\frac{p_x}{p_z}, \quad c = v'_y - v'_z\frac{p_y}{p_z}, \quad \mathbf{h} = \mathbf{p} \times \mathbf{n}$$

The general optical flow constraint equation [12] involving the partial derivatives of $\hat{I}_1$ in horizontal and vertical direction and the temporal difference is

$$\frac{\partial\hat{I}_1}{\partial\hat{X}_1}u_m + \frac{\partial\hat{I}_1}{\partial\hat{Y}_1}v_m \approx \hat{I}_1 - I_1 \quad . \tag{8}$$

Combining Eq. (8) and (7) results in a linear equation with $\mathbf{a} = (a_0, a_2, \ldots a_5)$ and six unknown motion parameters $\Delta\mathbf{t}$ and $\Delta\mathbf{r}$

$$\mathbf{a}\begin{pmatrix}\Delta\mathbf{r}\\ \Delta\mathbf{t}\end{pmatrix} = \hat{I}_1 - I_1 \quad . \tag{9}$$

One equation is formulated for each pixel corresponding to the projection in $\hat{I}_1$. The set of equations forms an overdetermined system which can be solved in a least squares sense or with a robust estimator. The six coefficients are defined as

$$\mathbf{a} = \left(\frac{1}{p_z(\mathbf{n}^T\mathbf{v}')}\left(-f_x b\frac{\partial\hat{I}_1}{\partial\hat{X}_1} + f_y c\frac{\partial\hat{I}_1}{\partial\hat{Y}_1}\right)\right)\begin{pmatrix}\mathbf{h}\\ \mathbf{n}\end{pmatrix}^T \quad . \tag{10}$$

When the system of equations is solved, the linearized rotation $\Delta\mathbf{R}$ is concatenated with the previous rotation. The translation offset $\Delta\mathbf{t}$ can simply be added to the previous value. Additionally, a smoothness term including a damping factor can be added to the system to temporally smooth the rotation and reduce jitter between frames.

Limitations Related to the aperture problem of optical flow [4], ambiguities in the pose estimation arise for certain object shapes. If the illuminated object is a perfect sphere, a rotation around its center will not change the appearance of the projection in the camera image and therefore no pixel can give information about rotation of the object. The distortion of an image projected on a flat surface only provides information about three degrees of freedom, since the in-plane translation und rotation can't be recognized from it [9]. A cylinder can be moved along its axis, without changing the local appearance of the projection. A more complex object shape helps to prevent ambiguities, since differently orientated regions provide different hints about the movement and allow the unique determination of all motion parameters. A combination with projection-independent terms (see Section 4.2) also prevents ambiguous situations.

3.3 Implementation Details

In our analysis-by-synthesis approach, the image $\hat{I}_1$ synthetically simulates the projection on the object at time $t-1$, while I_1 is captured by the camera at time t. If a motion between $t-1$ and t has occurred, the projection does not match perfectly with the expected appearance simulated in $\hat{I}_1$. The goal of the optimization is to alter the object pose while keeping the last projector image to synthesize the appearance of the camera image, by using the calculations given in Section 3.2. When projecting on a real-world surface, environmental conditions as illumination, scene, camera and projector characteristics, influence the captured image. In order to keep the simulation simple and allow a fast computation of the 6-DoF pose, we transform both image into binary edge images and minimize the image difference between those.

To account for different image characteristics we apply adaptive thresholds. First, the simulated image of the projection without any background information serves as a reference. The magnitude of the

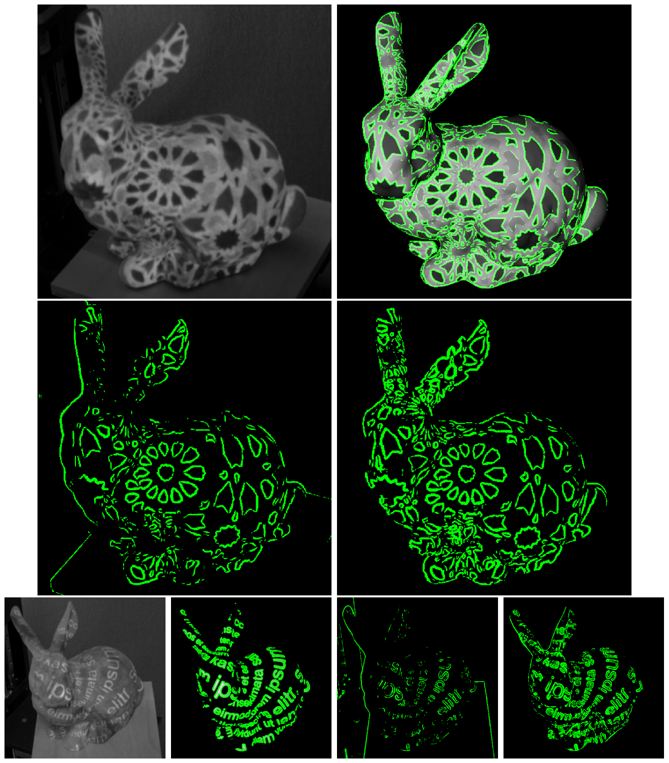
Fig. 4. Visualization of the effect of the adaptive thresholding. The figure shows the camera image (top left), the simulated projection and the detected edges using a global threshold (top right), the edges from the camera image using the same global threshold (bottom left) and the edges from the camera image using adaptive thresholds (bottom right). In the bottom row a further example is shown.

Sobel filtered image is binarized with a fixed threshold. While doing this, an *NxN* grid is laid over the image and in each tile, the number of edge pixels is counted. During edge detection in the camera image the same grid is laid over the image and an individual binarization threshold is found for each tile, leading to approximately the same number of edge pixels. Fig. 4 shows the effect of this operation. The adaptive thresholds ensure that the same amount of information is present in both images, even if there are contrast and illumination variations. With the binarization we achieve a comparable value range for the optical flow.

Both images are blurred with a 7*x*7 box filter to spatially extend the edges. The gradients between the two images is then computed in *x*, *y* direction and temporally in *t* direction. The equation system is built up pixel-wise, whereby three filters are applied to decide whether an equation is added to the system or not.

1. Equations can only be created in regions where a depth value is available in $\hat{I}_1$ because this information is necessary for the calculation.

2. Pixels, which are close to the geometric borders of the object are ignored, since the assumption of locally planar regions is not valid at the object border. To do so, we render a separate image containing the geometrical object contour and edges, spatially extend them and ignore all values in overlapping regions.

3. Pixels at positions with steep angles between viewing ray of the projector and the surface normal are ignored, since the intensity of the projection vanishes there.

An iteratively reweighted least squares (IRLS) scheme [31] makes the calculation robust against outliers. Outliers occur e.g. if a part of the object is occluded, if the object has a texture, which remains visible under the projection or if there are strong reflections or overexposure not visible in the synthetic image. As proposed by Sun et al. [27], we use the robust Charbonnier penalty $p(x) = \sqrt{x^2 + \varepsilon^2}$ with a small ε (e.g. $\varepsilon = 0.001$) to weight the equations individually. Iterating two or three times using the residual error of the previous iteration to calculate the weights stabilizes the pose estimation.

To overcome bigger differences between two frames, we use an image pyramid and render a new synthetic image for each lower pyramid level, where the last result serves as starting pose. The optimal depth for the pyramid depends on the amount of detail in the projection in relation to the size of the object in the image. If the texture is very detailed and the image is scaled down too much, the pose estimation can become unstable in the smallest pyramid level, propagating the error to higher pyramid levels. Instead of minimizing the image too much, we recommend to improve processing speed and synchronization, as proposed in Section 3.6.

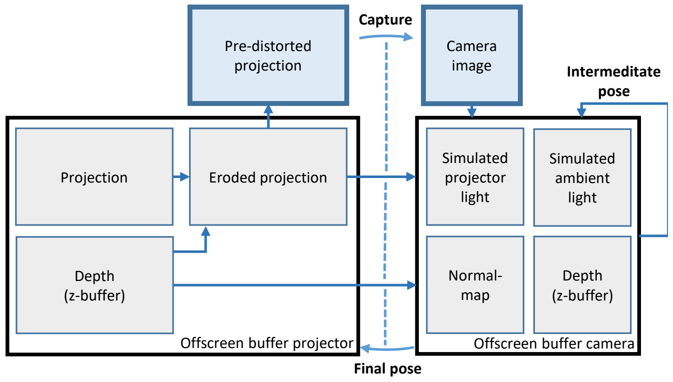

Fig. 5. The visualization of the graphics pipeline shows the closed-loop principle. The final object pose serves as input for the projection in the following frame. The camera image is captured after the projection is visible.

3.4 Graphics Pipeline

All synthetic images are rendered with OpenGL. The projection is displayed in fullscreen window on the projector. The pose estimation uses the simulation of the projection in the camera frame. It is rendered in an offscreen buffer, which is utilized and updated every iteration of the computation.

If the pose of the object is known in the camera coordinate system, the corresponding projection can be created following the shader lamps principle [22]. The object is visualized from the point of view of the projector. In the first step, the projection is also rendered in an offscreen buffer and a depth buffer-based mask removes artifacts at the object contour by applying a morphological erosion on the object border [13].

The created image as well as the depth buffer information is stored as a texture and is used during the simulation of the projection from the point of view of the camera. Before projecting the image, a radial lens distortion of the projector can be compensated by pre-distorting the image.

We apply projective texture mapping [8] to simulate the projection on the object surface and incorporate shadow maps to reproduce self-occlusions, which make the projection invisible at parts of the object. Furthermore, z-buffer and surface normals are needed for the pose estimation. The simulated image is decomposed into one image showing only the object surface with ambient illumination and one image showing only the reflected projector light on the object surface. In sum, both images simulate the appearance of the projection on the object. Fig. 5 shows the whole rendering pipeline and stresses the closed-loop principle.

3.5 Initialization

Initially, the tracker requires a start pose close to the actual pose to achieve convergence of the estimation. Since the rendered image is directly compared to the camera image, an overlap at the blurred edges is needed. The required accuracy of the initial pose depends on the number of pyramid levels and the size of the box filter used to blur the image.

Our pose initialization requires that there is no shader lamp projection covering the object and is based on a contour-based tracker [28]. Along a scanline perpendicular to the object contour, local maxima of the gradient of the corresponding grayscale profile are searched. By this means, we find 2D-2D correspondences and minimize the distance along the normal at the contour point. Combined with a refinement based on the model-based rigid body motion estimation from Steinbach et al. [26] and the IRLS scheme the estimation becomes stable and accurate. The length of the scanline (e.g. 50 pixel in all pyramid levels) is bigger than the size of the box filter. While still having a local tracker, it is possible to deal with larger pose offsets than with the projector-based approach alone.

If the initial pose is completely unknown, machine learning-based pose estimation methods as Pose-CNN [30] achieve state-of-the-art results, but also cannot deal with an interfering projection.

Additionally we estimate the scene illumination and the brightness of the projection to make the simulated images appear similar to the camera image. Since we compare the edge images with adaptive thresholds, a simple approximation model is sufficient and only gray scale images are used. We use Lambert illumination model to describe the brightness for points on the object.

$$I_{ProjOff} = (I_D \cdot max\{\mathbf{n}^\mathbf{T}\mathbf{l}, 0\} + I_A) \cdot (c_{Off} + c_{Obj}) \tag{11}$$

Because the object is already registered, the normal direction $\mathbf{n}$ and the brightness in the image $I_{ProjOff}$ are known. In addition, the approximate object color c_{Obj} or its texture brightness is predetermined. The ambient intensity I_A, the diffuse intensity I_D, the direction of light $\mathbf{l}$ and an additive brightness offset c_{Off} are estimated by solving a linear system with one equation per pixel of the object surface.

To estimate the brightness of the projector, we project a white texture on the object. The intensity in the camera image I_{ProjOn} is influenced by the cosine between the surface normal and the direction of the projector light $\mathbf{v}'$ and linearly attenuated by the distance between the point and the projector $\|\mathbf{p}'\|$.

$$I_{ProjOn} = I_{ProjOff} + \frac{I_P \cdot max\{\mathbf{n}^\mathbf{T}\mathbf{v}', 0\}}{\|\mathbf{p}'\|} \tag{12}$$

The brightness of the projector I_P is calculated with a linear equation system and one equation per pixel. Example results of the illumination estimation are shown in Fig. 6.

3.6 Synchronization and Timing

We intentionally limit ourselves to use off-the-shelf hardware and omit expensive and inconvenient hardware-based synchronization. Petković et al. [20] propose a method to achieve 30 fps tracking with simple software synchronization, which is also applicable for the described tracker if the following attributes are fulfilled by hard- and software. Here t_D is the sum of input lag of the projector and the delay of the camera trigger, and t_R is the frame readout time measured in advance.

1. Frames are displayed in sync with the vertical blank interrupt.
2. If a time-multiplexing projector is used, the exposure time of the camera t_E is set to the exact rotation time of the color wheel.
3. It is possible to trigger the camera with an accurate software trigger with a delay of t_D after the last frame has been displayed.
4. The time $t_E + t_R + t_D$ is smaller than 33.33 ms.
5. The preparation time of one frame, including pose estimation and the generation of a new projector image is smaller than 33.33 ms.

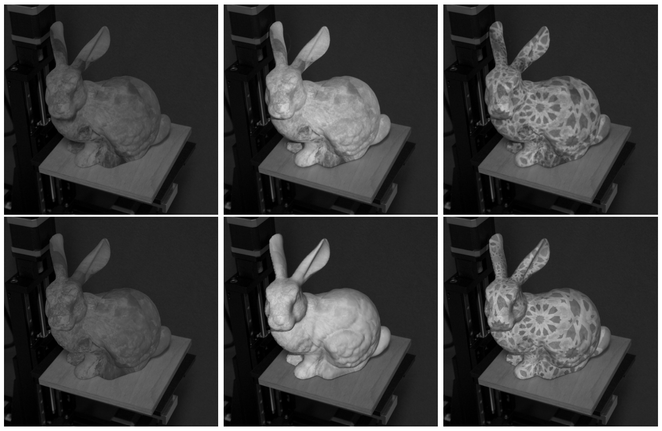
Fig. 6. Comparison of camera images (top row) and images with a rendered overlay (bottom row) during initial illumination estimation using an object with known texture. The estimated ambient illumination (left) and the estimated projector brightness (middle) allow the simulation of a texture projection (right).

OpenGL natively supports Requirement 1. Requirement 2 as well as 3 are default features of most machine vision cameras. To fulfill requirement 4 the projector used in our experiments is too slow, since it runs at 60 Hz, forcing t_E to be 16.66 ms and its input lag is longer than the allowed time interval as well. A projector running at 120 Hz with an input lag below the given thresholds is needed for 30 fps tracking [20]. However, this is true for a lot of consumer devices. Theoretically, the object moves under the projection during the exposure time. This can cause motion blur in the image of the projection. It could either be modelled mathematically to support the tracking [25] or minimized by the usage of a high-speed camera capturing each position of the color wheel separately. So far, we have not experienced this type of blur to be a problem, so both is not covered by this paper.

Our algorithm is designed to use only small image filters and local pixel-wise operations. This makes it portable to a graphics processing unit (GPU) to fulfill requirement 5. Building up the equation system can be realized with compute shaders using standard OpenGL. Our current CPU implementation takes about 100 ms to process one frame with a resolution of $1224x1024$ pixels, including the intermediate downloading of images from the graphics card to the CPU, using two pyramid levels and three iterations of the IRLS. We also created a GPU implementation which fulfills requirement 5 with the same settings on an *Nvidia GeForce GTX TITAN* graphics card. Including the time to upload the texture to the GPU initially, we reduce the processing time by a factor of four.

4 Extensions

The projection-based tracking offers possibilities to be transferred to different problems and extended with additional constraints as described below.

4.1 Calibration and Refinement of Extrinsic Parameters

We calibrated the projector-camera system with the method of Moreno et al. [17]. For our test projector, it requires to project a sequence of 44 gray-code patterns on a static checkerboard pattern. This sequence is caputed for different checkerboard orientations, resulting in several hundred pictures for one calibration. If the setup changes, the process has to be repeated. A model-based approach has been presented by Resch et al. [24] before, using also gray-code patterns and similarity ICP. However, with slight modifications of the equations given in 3.2 our method can be adapted to refine the extrinsic parameters of the projector-camera system, given an accurately known pose of a 3D object. In contrast to other methods, the shader lamp projection is directly integrated in the estimation without the need for additional patterns and allows a fast recalibration.

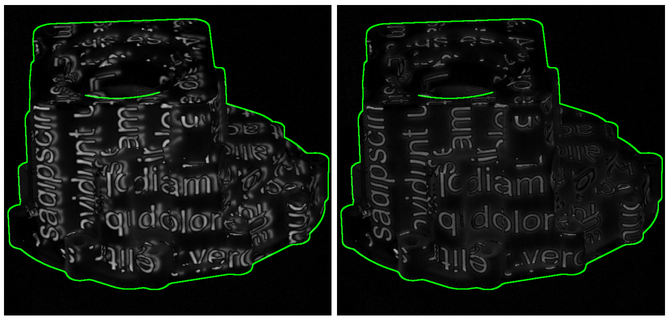

Fig. 7. Refining the extrinsic pose between projector and camera (right) improves the alignment between captured projection and simulation, compared to the result after the initial calibration (left).

In the new model, the translation offset of the projector relative to the camera is $\Delta\mathbf{t}_P$ and the rotation offset is $\Delta\mathbf{R}_P$. We reformulate the motion model given in 3.1, whereby this time the viewing ray $\mathbf{v}'$ of the projector is rotated and the translation offset is directly added to $\mathbf{p}$.

$$\hat{\mathbf{p}} = \mathbf{p} + \Delta\mathbf{t}_P + d\Delta\mathbf{R}_P\mathbf{v}' \tag{13}$$

Corresponding to Eq. (5), the distance d is

$$d = \frac{(\mathbf{p} - \mathbf{t}_p - \Delta\mathbf{t}_P)^T\mathbf{n}}{(\Delta\mathbf{R}_p\mathbf{v}')^T\mathbf{n}}. \tag{14}$$

We construct a linear equation system and using the values $\mathbf{a} = (a_0, a_2, \dots a_5)$ to derive the unknown motion parameters.

$$\mathbf{a} = \frac{f_x}{p_z}\frac{\partial \hat{I}_1}{\partial \hat{X}_1}\left(\mathbf{u_1} + \frac{b}{\mathbf{n}^T\mathbf{v}'}\begin{pmatrix}\mathbf{h}\\ \mathbf{n}\end{pmatrix}\right) - \frac{f_y}{p_z}\frac{\partial \hat{I}_1}{\partial \hat{Y}_1}\left(\mathbf{u_2} + \frac{c}{\mathbf{n}^T\mathbf{v}'}\begin{pmatrix}\mathbf{h}\\ \mathbf{n}\end{pmatrix}\right) \tag{15}$$

with

$$\mathbf{u_1} = \begin{pmatrix} v'_y\frac{p_x}{p_z} & -v'_x\frac{p_x}{p_z} - v'_z & v'_y & -1 & 0 & \frac{p_x}{p_z}\end{pmatrix}^T$$
$$\mathbf{u_2} = \begin{pmatrix} v'_y\frac{p_y}{p_z} + v'_z & -v'_x\frac{p_y}{p_z} & -v'_x & 0 & -1 & \frac{p_y}{p_z}\end{pmatrix}^T$$

Furthermore, b and c are calculated as in Eq. (7) and $\mathbf{h} = -(\mathbf{v} \times \mathbf{n})$. Besides interchanging the equations, the pipeline from the previous chapter remains unchanged. Potentially, all objects suited for tracking can become calibration targets, provided that the object pose can be estimated very accurately with the method from Section 3.5 and that the intrinsic parameters as well as the approximate extrinsic parameters are known. For *focus-free* projectors (laser scanning or laser + LCoS), the focus is fixed and intrinsic parameters are not influenced if the projector-camera setup changes. We recommend to use a texture with sharp edges when doing the recalibration. Using an object which fills a large part of the image and which extends in z-direction helps the estimation to converge. Also the full resolution without subsampling and reduced ambient light help to obtain stable results.

Fig. 7 shows a difference image between the rendered image and camera image after a calibration with the method of Moreno et al. [17] and after our refinement. The improved alignment between expectation and observation also stabilizes the tracking, since the simulation can better predict the actual image.

4.2 Combination with a Contour-based Tracker

Besides the computational simplicity of the direct pose offset computation with a linear system, another advantage is that the system can be extended with more equations describing other motion models. If the motion parameters are the unknowns in the equations, additional constrains can be added to the optimization. We exploited this and allow a combination of the projection-based approach with the model-based rigid body tracking equations [26]. The synthetic image containing the geometrical object contour used only for filtering in the previous chapter can be fed into the algorithm from Section 3.3. Now the optical flow is additionally computed between the contour image and a binarized Sobel image with thresholds adapting to this contour image. Therefore, the equation system includes a further condition to fit the object contour to contours in the image. This prevents ambiguous configurations where a projection on the object appears similar in the camera frame under different object poses.

The usefulness of the combined model depends on the visibility of the actual object contour in the image. It is influenced by ambient light and the character of the projected content. Especially, if the projector illuminates only a part of the object the contour can provide useful information.

5 Results

In dynamic closed-loop shader lamp scenes, the projected image is affected by the previously estimated pose. The following camera frame in turn contains this projection. Hereby, recording a test sequence of a moving object and evaluating this sequence with varying parameters is hindered. To prove the functionality of our approach, synthetic sequences allow the creation of repeatable tests with easily adjustable parameters. Since the synthetic tests run in an idealized environment and not all physical parameters of light and material can be model, tests with real data are essential too. We decided to use a motorized linear stage to produce reproducible object movements. This allows comparing various parameter settings with ground truth data. During the evaluation, we use the CPU version of our algorithm.

5.1 Synthetic Data

We create a synthetic test environment to simulate the tracking. A target object is placed in a 3D scene and a texture projection on the object is simulated. The objects moves on a specific path and the initial pose and the corresponding projection are known. For following frames, the pose to create the projection is estimated from a synthesized camera image showing the previous projection on the moved object. Thus, effects like drift over time can be simulated. The virtual projector-camera system uses calibration and illumination parameters from the real data test. The system is placed inside a textured, cube shaped box, serving as background. If the object moves, a part of the projected texture misses the target object and hits the background object, simulating the delay in the closed-loop approach. The rest of the projection is perspectively distorted by the object surface.

For synthetic tests, we utilize the well known Stanford Bunny model. We have chosen two different textures to texturize the model during the tests, reproducing different spatial AR scenarios. Both textures cover the object surface evenly with a detailed pattern. Generally, the accuracy of the estimation depends both on the level of detail in the texture as well in object shape and these two factors can compensate for each other to a certain degree (see Section 3.2).

Texture 1 contains white dummy text on a black background. Text projection is crucial for AR assistance scenarios, where additional information has to be placed on the object surface.

Texture 2 is a colored geometric pattern, as it could be found in design mockups or art installations. Compared to the first texture there are no black areas and the projector light illuminates the whole object.

Additionally the model is textured, as shown on the left side of Fig. 6. This introduces another difficulty, since in the adaptive edge image contours not related to the projection can appear.

To test the projector-based tracking two experiments are conducted with synthetic data. For both experiments, in one iteration 100 different sequences of random movements are simulated. The initial distance between object and camera is 0.7 meter. The initial orientation is chosen randomly for each sequence. The target pose differs randomly from the initial pose, whereby the absolute sum of the translation offsets along the axes is fixed to be m cm or n degrees for the rotation respectively.

Experiment 1 tests the tracking quality for small pose offsets between frames and a continuous motion. To move the object to the target pose we linearly interpolate the movement over f frames. We calculate the root mean square error (RMSE) of the motion parameters for each

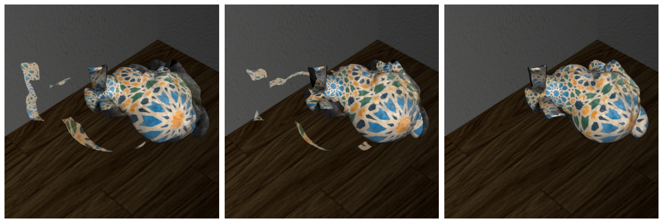

Fig. 8. Example result of Experiment 2. Initially (left) a big part of the projection misses the object. In the fifth frame (middle) the matching has already improved. After 12 frames (right) the projection perfectly matches the object shape.

sequence independently to judge the tracking quality. The translation error t_{err} and the rotation error r_{err} are the mean of the absolute errors for r_x, r_y, r_z (degree, euler angles) or t_x, t_y, t_z (mm) respectively. Furthermore, a sequence is considered as *valid* if its RMSE of r_{err} and t_{err} is smaller than 5 degrees or 5 mm. The other sequences where higher error values indicate a drifting pose over time are considered as outliers in the following. We want to measure the tracking quality for all sequences in which our algorithm is able to track the object to the end of the sequence and state the average error values over all *valid* sequences in Table 1. For each texture, the last column shows how many out of 100 sequences were considered as *valid*.

Test			**Texture 1**			**Texture 2**		
m	n	f	t_{err}	r_{err}	*valid*	t_{err}	r_{err}	*valid*
6	60	60	0.08	0.20	100	0.27	0.54	97
6	60	30	0.10	0.25	100	0.31	0.66	94
6	60	15	0.14	0.38	98	0.55	1.06	75

Table 1. Results of Experiment 1 with two textures. The values under r_{err}, t_{err} are the mean of the RMSE calculated over all valid sequences.

In the first iteration of the test, a movement of 60 degrees and 6 cm is distributed over 60 frames. In iteration two and three, the speed is doubled by using 30 or 15 frames respectively for the movement. It is observable that the estimated orientation is very precise for the slowest movement for both textures. For Texture 2, which is more detailed than Texture 1, the RMSE is higher but still below one mm and one degree. Even if the speed is quadrupled the r_{err} exceed this mark only minimally. The outliers arise because larger objects movements can lead to a sticking to a local minimum. Since the object is moving continuously, an error can increase over time, especially if the texture is very detailed in relation to the camera resolution. This can be avoided by giving the algorithm more time to converge as we show in the next experiment.

Experiment 2 tests the ability to cope with larger pose offsets. Like in the first experiment, a random target pose is calculated, whereas this time the objects jumps to the target abruptly. In the following, f frames are allowed for the tracker to converge to the visible pose. Here for each frame an updated projector image is generated and virtually projected. Fig. 8 shows input images in course of one successfully converging sequence. In this test, only the error of the last frame of each sequence is relevant, so r_{err} and t_{err} are averaged over those final errors instead of calculating the RMSE of the sequence.

Test			**Texture 1**			**Texture 2**		
m	n	f	t_{err}	r_{err}	*valid*	t_{err}	r_{err}	*valid*
1	10	20	0.04	0.13	100	0.19	0.34	96
2	20	20	0.06	0.17	94	0.07	0.20	81

Table 2. Results of Experiment 2 with two textures. The values under r_{err}, t_{err} are the mean of the final errors calculated for each valid sequence individually.

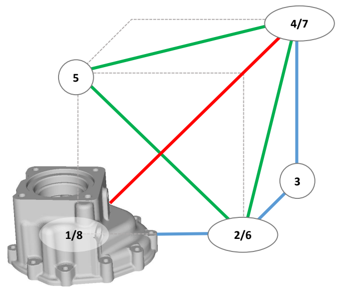

Fig. 9. Trajectory of the object during test with motorized linear translation stage. The object moves following increasing numbers. The side length of the cube is 5 cm.

As shown in Table 2, an abrupt movement bigger than the largest movement between two frames in experiment 1 can be compensated if the algorithm iterates longer and has more time to converge. Due to the iterative pose improvement, the quality of the outcome is not effected by the initial offset in case of convergence. The second row shows that the number of invalid tests increases again after a large distance is exceeded. Offsets in that magnitude can be avoided in practice by the previously described initialization process.

5.2 Real Data

In this section, the tracking is evaluated on real data. We test three approaches in presence of a shader lamp projection. The first uses the projection-based algorithm, the second uses a contour-based tracker, ignoring the projection (described in Section 3.5), the third uses the combined approach from Section 4.2.

Our tests system consists of a *Basler ace acA2440-75uc* camera with a *Kowa LM12SC* objective and a *Cremotech Laser Beam Pro C200*. The devices area rigidly mounted on a tray with a baseline of approximately 15 cm. It is calibrated, by starting with the classical method [17] with a stereo reprojection error of 0.489 pixel, followed by an iterative refinement of the extrinsic parameters. The distance between projector and target object is 0.7 meter.

The linear stage allows movements along the three coordinate axes. Even though only a translation is applied on the objects, all six degrees of freedom are estimated throughout the tests. Fig. 9 shows the trajectory along which the objects are moved. A top-down view on the test setup, as well as the two target objects with the applied projection are shown in in Fig. 10. In the experiments, 3D-printed, uniformly white spare parts of industrial workpieces serve as test objects.

Initially the object is placed on the platform and the method from Section 3.5 provides an estimation of the illumination and an accurate start pose. Then the shifting unit moves 5 cm along its x-axis and also at this position a pose estimation is performed carefully. The object is placed on the platform with its y-axis pointing upwards. The x-axis direction and the up-vector allow us to determine the orientation of the global coordinate system of the translation stage in relation to object orientation.

Besides interchanging the two textures from the synthetic test, we test the tracking in presence and in absence of ambient illumination. The ambient illumination increases the visibility of the object contour, demanding more robustness during pose estimation. Apart from that, it also allows to use the combined approach. The contour-based pose estimation is tested with and without shader lamp illumination. The moving speed of each axis is 5 mm per second. If two or three axes move at once (green and red lines in Fig. 9), the speed of the object increases accordingly. In the test, the translation error is the distance between the currently estimated position and the trajectory. The rotational error is the absolute difference between the estimated rotation and the initial rotation.

The obtained error values for Object 1 are summarized in Table 3. Using Texture 1, it can be tracked throughout the whole trajectory with (**Proj. + Amb.**) and without (**Proj.**) ambient illumination. The RMSE of the translation is below 1 mm and even smaller than for the

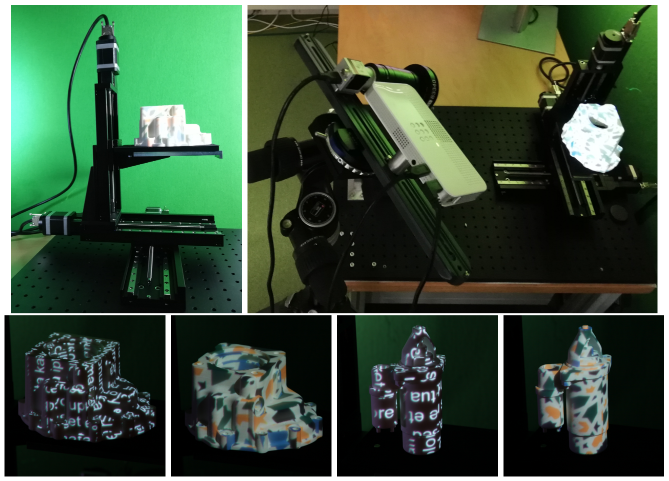
Fig. 10. First row: Visualization of the test setup. The target object is placed on a motorized linear translation stage (left) and is illuminated by the projector-camera system (right). Second row: Object 1 (left) and Object 2 (right), each illuminated two different textures.

contour-based approach without an active projection (1.29 mm). In addition, the rotation error is smaller than one degree. The translation error of the combined approach (**Comb. + Amb.**) differs minimally, but the rotation is slightly more stable (compared to Proj. + Amb). The contour-based tracker only works without shader lamp illumination and otherwise drifts after reaching position four on the path (**Cont. + Amb.**).

For Texture 2, the tracking can be successfully maintained over the complete sequences with and without ambient illumination. We observe that the diagonal movements introduce a small rotational error, which is automatically recovered after reaching a turning point. Interestingly the recovery works better for the approach using only the projection, than for the combined approach, which fails during the last movement. Even if the projection is slightly off, a large part of the object is covered under the projection, providing enough information to estimate the correct movement direction. The bright border of the projection introduces an error in the contour-based terms.

	Texture 1		**Texture 2**	
Test	t_{err}	r_{err}	t_{err}	r_{err}
Proj.	0.78	0.45	0.99	0.51
Proj. + Amb.	0.84	0.97	1.16	0.95
Comb. + Amb.	0.85	0.79	fails after Pos. 7	
Cont. + Amb.	fails after Pos. 4		fails after Pos. 4	

Table 3. RMSE of rotation and translation (r_{err}, t_{err}) for the first object in presence of shader lamp illumination.

Object 2 has a cylindrical shape and it is not as wide as the first object. A larger part of the projection misses the object for a movement of the same size. Using Texture 1, the object can be tracked with the projector-based model with and without ambient illumination. A constant offset is observable during the movement from position 4 to 5. The offset is not present using the combined tracker. The offset can be explained by the approximately cylindrical shape of the object. A movement of the object exactly along the main axis of a cylinder will not change the image of the projection locally in the camera image and an error tends to be introduced in that direction. Using the additional contour equations restricts this false estimation. For Texture 2, this phenomenon also arises. Maintaining the tracking till the end of the sequence without the contour terms is possible if we reduce to speed to $4mm$ per second ($t_{err} = 3.32$, $r_{err} = 1.83$). This is not necessary using the combined approach. Still the obtained errors are lower ($t_{err} = 1.46$, $r_{err} = 0.67$). Fig. 11 shows the correcting effect of the combined

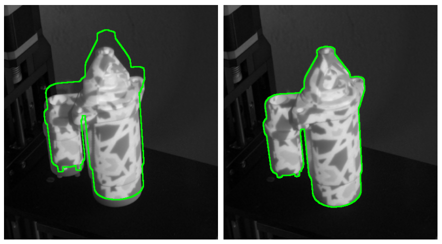
Fig. 11. The estimated pose (green line) drifts along the main axis of a cylindrical object during a tracking sequence for the projection-based tracking (left). The combined tracking approach (right) resolves ambiguities.

approach. As expected, the tracking fails if the projector-based terms are ignored.

While we performed all experiments with two pyramid levels and a maximum resolution of $1224x1024$, we also tested to use only one or three pyramid levels. For three levels, the large amount of texture detail in relation to the size of the object in the image (especially if the object is far away from the camera) causes pose instabilities in the coarsest resolution. On the other hand, the highest resolution alone only allows the compensation of small offsets. If the input image is subsampled to a resolution of $612 * 512$, larger offsets can be compensated, but the pose is not as accurate as for two levels. Therefore, the chosen settings are a good compromise between speed and accuracy.

In conclusion, we can say that the projection-based approach is suitable to precisely track objects with different geometric shape, illuminated with different textures. The experiments show that the registration is possible in cases where a purely contour-based tracker fails. Due to the repeatable movement with the linear stage, we could identify situations where the combined tracker using projection and contour leads to an additional benefit.

A video showing that our approach is able to track a manually moved object is part of the supplementary material. The hand of the user introduces a partial occlusion of the projection and the free movement covers rotations as well as translations. For slow movements, as shown in the video, the CPU implementation is sufficient. To track faster movements, more frames per second have to be evaluated, for example by using a GPU variant of the algorithm (see Section 3.6) in future work.

6 Conclusion

In this work, we presented a new approach for pose estimation of 3D objects by directly aligning a simulated image of a spatial AR projection with the camera image. The appearance of dynamically moving objects can be manipulated using a shader lamp projection, while at the same time the distortion of the projection is used for pose estimation. We derived the necessary motion model and presented an optical flow-based pose estimation algorithm making use of the principle. We omitted a radiometric calibration by using binary edge images with adaptive thresholds and a very fast initial illumination estimation. The developed linear optimization algorithm can be extended to optimize the pose of the projector instead of the pose of the object. This allows a fast refinement of the extrinsic parameters of a projector-camera system, using a reference model with known 3D shape. A refinement of the intrinsic parameters of the projector using more input poses is an interesting topic for future research. We have proven in tests with synthetic and real data, that the algorithm is suitable to track moving objects with and without ambient illumination and with different projected textures. Many AR applications in industry and design can potentially benefit from the presented techniques, already in the current state of development. In future, we want to invest time in improving the real-time performance of our implementation and further explore the possibilities of combining the projection distortion-based estimation with other motion models.

ACKNOWLEDGMENTS

This work is supported by the German Federal Ministry of Education and Research (EASY-COHMO project, grant no. 03ZZ0443E) and the German Federal Ministry of Economic Affairs and Energy (DigitalTWIN project, grant no. 01MD18008B).

REFERENCES

[1] R. S. Andersen, O. Madsen, T. B. Moeslund, and H. B. Amor. Projecting robot intentions into human environments. In *Proc. RO-MAN*, pp. 294–301. IEEE Computer Society, Washington DC, USA, 2016. doi: 10.1109/ROMAN.2016.7745145

[2] H. Asayama, D. Iwai, and K. Sato. Fabricating diminishable visual markers for geometric registration in projection mapping. *IEEE Transactions on Visualization and Computer Graphics*, 24(2):1091–1102, Feb. 2018. doi: 10.1109/TVCG.2017.2657634

[3] S. Audet, M. Okutomi, and M. Tanaka. Direct image alignment of projector-camera systems with planar surfaces. In *Proc. CVPR*, pp. 303–310. IEEE Computer Society, Washington DC, USA, 2010. doi: 10.1109/CVPR.2010.5540199

[4] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–466, Sept. 1995. doi: 10.1145/212094.212141

[5] A. H. Bermano, M. Billeter, D. Iwai, and A. Grundhöfer. Makeup Lamps: Live Augmentation of Human Faces via Projection. *Computer Graphics Forum*, 36(2):311–323, May 2017. doi: 10.1111/cgf.13128

[6] L. Besharati Tabrizi and M. Mahvash. Augmented reality-guided neurosurgery: accuracy and intraoperative application of an image projection technique. *Journal of neurosurgery*, 123(1):1–6, Mar. 2015. doi: 10.3171/2014.9.JNS141001

[7] M. Chae, D. Ganhewa, D. Hunter-Smith, and W. Matthew Rozen. Direct augmented reality computed tomographic angiography technique (ARC): an innovation in preoperative imaging. *European Journal of Plastic Surgery*, 41(4):415–420, Jan. 2018. doi: 10.1007/s00238-018-1395-2

[8] C. W. Everitt. Projective texture mapping. `http://www.cse.unsw.edu.au/~cs9018/readings/projective_texture_mapping.pdf`, 2001. [Online; accessed 10-June-2019].

[9] N. Gard and P. Eisert. Markerless closed-loop projection plane tracking for mobile projector-camera systems. In *Proc. ICIP*, pp. 3363–3367. IEEE Computer Society, Washington DC, USA, 2018. doi: 10.1109/ICIP.2018.8451038

[10] K. M. Guttag and S. Hurley. Laser + LCOS projection-technology revolution. *Journal of the Society for Information Display*, 20(5):279–285, May 2012. doi: 10.1889/JSID20.5.279

[11] N. Hashimoto and D. Kobayashi. Dynamic spatial augmented reality with a single ir camera. In *Proc. SIGGRAPH Posters*, pp. 5:1–5:1. ACM, New York, NY, USA, 2016. doi: 10.1145/2945078.2945083

[12] B. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1–3):185–203, Aug. 1981. doi: 10.1016/0004-3702(81)90024-2

[13] J. Kern, M. Weinmann, and S. Wursthorn. Projector-based augmented reality for quality inspection of scanned objects. *The international archives of photogrammetry, remote sensing and spatial information sciences*, 4(2):83–90, Sept. 2017. doi: 10.5194/isprs-annals-IV-2-W4-83-2017

[14] D. Kobayashi and N. Hashimoto. Spatial augmented reality by using depth-based object tracking. In *ACM SIGGRAPH Posters*, pp. 33:1–33:1. ACM, New York, USA, 2014. doi: 10.1145/2614217.2614226

[15] R. Koizumi, D. Kobayashi, and N. Hashimoto. Acceleration of dynamic spatial augmented reality system with a depth camera. In *Proc. CW*, pp. 50–53. IEEE Computer Society, Washington DC, USA, 2015. doi: 10.1109/CW.2015.42

[16] M. R. Marner, R. T. Smith, J. A. Walsh, and B. H. Thomas. Spatial user interfaces for large-scale projector-based augmented reality. *IEEE Computer Graphics and Applications*, 34(6):74–82, Nov. 2014. doi: 10.1109/MCG.2014.117

[17] D. Moreno and G. Taubin. Simple, accurate, and robust projector-camera calibration. In *Proc. 3DIMPVT*, pp. 464–471. IEEE Computer Society, Washington DC, USA, 2012. doi: 10.1109/3DIMPVT.2012.77

[18] G. Narita, Y. Watanabe, and M. Ishikawa. Dynamic projection mapping onto deforming non-rigid surface using deformable dot cluster marker. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1235–1248, Mar. 2017. doi: 10.1109/TVCG.2016.2592910

[19] M. K. Park, K. J. Lim, M. K. Seo, S. J. Jung, and K. H. Lee. Spatial augmented reality for product appearance design evaluation. *Journal of Computational Design and Engineering*, 2(1):38 – 46, Jan. 2015. doi: 10.1016/j.jcde.2014.11.004

[20] T. Petković, T. Pribanic, M. Donlić, and N. D'Apuzzo. Software synchronization of projector and camera for structured light 3D body scanning. In *Proc. 3DBST*, pp. 286–295. Hometrica Consulting, Ascona, Switzerland, 2016. doi: 10.15221/16.286

[21] R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao, and C. Forlines. ilamps: Geometrically aware and self-configuring projectors. In *Proc. ACM SIGGRAPH*, pp. 809–818. ACM, New York, NY, USA, 2003. doi: 10.1145/1201775.882349

[22] R. Raskar, G. Welch, K. Low, and D. Bandyopadhyay. Shader lamps: Animating real objects with image-based illumination. In *Proc. EGWR*, pp. 89–102. Springer, Berlin, Germany, 2001. doi: 10.2312/EGWR/EGWR01/089-101

[23] C. Resch, P. Keitler, and G. Klinker. Sticky projections-A model-based approach to interactive shader lamps tracking. *IEEE Transactions on Visualization and Computer Graphics*, 22(3):1291–1301, Mar. 2016. doi: 10.1109/TVCG.2015.2450934

[24] C. Resch, H. Naik, P. Keitler, S. Benkhardt, and G. Klinker. On-site semi-automatic calibration and registration of a projector-camera system using arbitrary objects with known geometry. *IEEE Transactions on Visualization & Computer Graphics*, 21(11):1211–1220, Nov. 2015. doi: 10.1109/TVCG.2015.2459898

[25] C. Seibold, A. Hilsmann, and P. Eisert. Model-based motion blur estimation for the improvement of motion tracking. *Computer Vision and Image Understanding*, 160:45–56, July 2017. doi: 10.1016/j.cviu.2017.03.005

[26] E. G. Steinbach, P. Eisert, and B. Girod. Model-based 3-d shape and motion estimation using sliding textures. In *Proc. VMV*, pp. 375–382. AKA GmbH, Berlin, Germany, 2001.

[27] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *Proc. CVPR*, pp. 2432–2439. IEEE Computer Society, Washington DC, USA, 2010. doi: 10.1109/CVPR.2010.5539939

[28] L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3D camera tracking. In *Proc. ISMAR*, pp. 48–57. IEEE Computer Society, Washington DC, USA, 2004. doi: 10.1109/ISMAR.2004.24

[29] M. Vo, S. G. Narasimhan, and Y. Sheikh. Texture illumination separation for single-shot structured light reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(2):390–404, Feb. 2016. doi: 10.1109/TPAMI.2015.2443775

[30] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Proc. RSS*. RSS Foundation, Berlin, Germany, 2018. doi: 10.15607/rss.2018.xiv.019

[31] Z. Zhang. Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting. *Image and vision Computing*, 15(1):59–76, Jan. 1997. doi: 10.1016/S0262-8856(96)01112-2

[32] F. Zheng, R. Schubert, and G. Weich. A general approach for closed-loop registration in AR. In *Proc. VR*, pp. 47–50. IEEE Computer Society, Washington DC, USA, 2013. doi: 10.1109/VR.2013.6549358