Tracking-by-Synthesis Using Point Features and Pyramidal Blurring

Gilles Simon

Tracking-by-Synthesis Using Point Features and Pyramidal Blurring

Gilles Simon*

Nancy-University, MAGRIT INRIA Research Team

ABSTRACT

Tracking-by-synthesis is a promising method for markerless vision-based camera tracking, particularly suitable for Augmented Reality applications. In particular, it is drift-free, viewpoint invariant and easy-to-combine with physical sensors such as GPS and inertial sensors. While edge features have been used succesfully within the tracking-by-synthesis framework, point features have, to our knowledge, still never been used. We believe that this is due to the fact that real-time corner detectors are generally weakly repeatable between a camera image and a rendered texture.

In this paper, we compare the repeatability of commonly used FAST, Harris and SURF interest point detectors across view synthesis. We show that adding depth blur to the rendered texture can drastically improve the repeatability of FAST and Harris corner detectors (up to 100% in our experiments), which can be very helpful, e.g., to make tracking-by-synthesis running on mobile phones. We propose a method for simulating depth blur on the rendered images using a pre-calibrated depth response curve. In order to fulfil the performance requirements, a pyramidal approach is used based on the well-known MIP mapping technique. We also propose an original method for calibrating the depth response curve, which is suitable for any kind of focus lenses and comes for free in terms of programming effort, once the tracking-by-synthesis algorithm has been implemented.

Index Terms: I.2.10 [Vision and Scene Understanding]: Motion— [I.3.7]: Three-Dimensional Graphics and Realism— Color, shading, shadowing, and texture

1 INTRODUCTION

Tracking-by-synthesis is a promising method for camera tracking that is particularly suitable for Augmented Reality (AR) applications [20]. This method is based on the knowledge of a *textured* 3D model of the scene. Figure 1 illustrates the four steps of one iteration of the algorithm. An *approximate* camera pose and the camera intrinsic parameters are assumed known at the beginning of the iteration. The approximate pose can be obtained using some physical sensors (GPS, inertial, ...) or taken as the pose obtained at previous iteration. From these data, the 3D textured model can be rendered onto the image plane (step 1). Some features (edges, corners, ...) are then matched between the rendered image and the camera image (step 2). As the virtual camera pose corresponding to the rendered image is known *exactly*, features in the rendered image, and therefore corresponding features in the camera image, can be back-projected onto the 3D model (step 3), providing a set of 3D-2D correspondences between the model and the camera image, from which the current pose can be obtained (step 4).

This method has several advantages over other markerless vision-based tracking methods [27, 14, 6, 12, 18]: (i) unlike recursive methods such as planar tracking [27] or visual SLAM [6, 12], errors do not accumulate from frame to frame and tracking is drift-free, (ii) as features are matched between close images, there is no need for scale or affine invariant features [15, 17, 2, 18] that are generally slow to compute or storage consuming, (iii) the system automatically performs detail culling and only searches for features that are likely to be visible at the current scale and (iv) merging with other sensor data is natural and can be used to get the initial pose [23], which is a recurring problem in AR.

*e-mail: Gilles.Simon@loria.fr

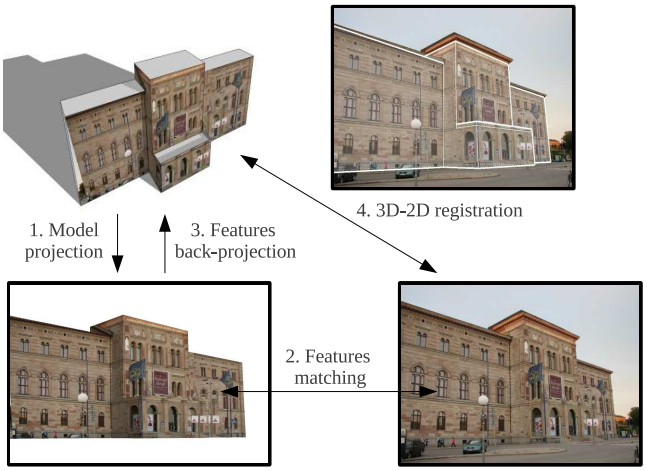


Figure 1: One iteration of the tracking-by-synthesis algorithm.

Despite these advantages, tracking-by-synthesis has surprisingly not been widely studied in the literature. In [20], a coarse, textured 3D model of an urban environment is used and edgels are extracted from the rendered view using a standard edge detector. These edgels are then projected back onto the model to obtain the 3D coordinates of the sample points. The edge search is conducted using an appearance-based model of the edge instead of a simple edge detector. This method has proven accurate enough for AR requirements. However, edges can be difficult to match on occluding contours, and while these features are often available on façades of buildings, these may be more rare on other kind of surfaces like grass-soils, carpets, and so on. Using point features, or combining point features with edgels, within the same framework, may therefore be advantageous in terms of robustness, accuracy and adaptability of the system.

However, up to our knowledge, point features have never been used within the tracking-by-synthesis framework as described above (see section 2.1). As our experiments show, this is probably due to the fact that state-of-the-art real-time corner detectors like FAST [22] or Harris [9] are weakly repeatable between a camera image and a rendered texture, especially when the texture was captured from a distant viewpoint. Indeed, point features detection can be sensitive to several factors such as texture resampling, depth blur, motion blur and illumination changes between the time when the texture was captured and the time when it is rendered. In this paper, we tackle the depth blur issue and show that simply adding depth blur to the rendered texture can drastically improve the repeatability of corner detection (up to 100% in our experiments).

The main contributions of the current paper are: (i) we com-

pare the repeatability of commonly used FAST, Harris and SURF interest point detectors across view synthesis (section 6), (ii) we propose a method for simulating depth blur on the rendered images using a pre-calibrated depth response curve associated with the camera (section 4). In order to fulfil the performance requirements, a pyramidal approach is used based on the well-known MIP mapping technique [29]; (iii) we propose an original method for calibrating the depth response curve of the camera (section 5), which is suitable for any kind of focus lenses (fixed, manually adjustable and motorized) and comes for free in terms of programming effort, once the tracking-by-synthesis algorithm has been implemented, (iv) we compare the repeatability of interest point detection across view-and-blur simulation (section 6). We show that adding blur greatly improves the repeatability of FAST and Harris corner detectors, leading to performance similar to SURF but with much faster running times, which can be very helpful to make tracking-by-synthesis running on mobile devices.

2 Related Works

2.1 Tracking-by-Synthesis Using Point Features

Point features have been used in visual panorama tracking where a purely rotational motion is assumed [8, 23, 28]. A pure rotation does not create a parallax effect and hence the environment can be mapped onto a closed two-dimensional surface, such as a cube, sphere or cylinder. The tracked video stream is used to create the environment map on the fly. Once the map is built, interest points in the camera image are compared against their counterpart in the map. In [23, 28], locations of the interest points are selected using the FAST corner detector [22]. Point correspondences are obtained using normalized cross correlation (NCC) between pixel patches in the camera image and pixel patches warped from the map. In [8], the Shi and Tomasi's good features operator [25] is used to find the candidate points. These are tracked from frame to frame using a pyramidal version of Lucas and Kanade's optical flow algorithm [16] and features that are tracked for a number of consecutive frames are added to the map and used to combat drift. Dot products between normalized SURF descriptor vectors [2] are used to match features in the map with features in the camera images. A similar approach is used in [3], though an Inertial Measurement Unit (IMU) is used to (re-)initialize the pose. Morevover, vision measurements and orientation data obtained from the IMU are fused using an Extended Kalman Filter.

6-degrees-of-freedom (6-dof) camera tracking based on pixel patches associated with a CAD model is presented in [14]. During an off-line stage, a small set of images, called keyframes, representing the scene or the object from different viewpoints, is chosen and calibrated. When the projection matrix is known for every keyframe, the system performs interest point detection using the Harris corner detector [9] and back-projects the 2D pixel patches around the points that lie on the object surface. Viewpoint invariant descriptors of these patches are obtained by rerendering each keyframe from different viewpoints around the camera position computed for the keyframe. Template matching is performed using the eigen image methodology.

Actually, all these techniques are tracking-by-synthesis in spirit, but not in letter, because once a feature is added to the map (or keyframe), it is used for the rest of the process. In tracking-by-synthesis as described above, the complete set of model features is replaced at each iteration by the set of features detected in the rendered image. This is the crucial step of this method, which makes it possible to handle large scale and perspective changes of the scene during camera tracking. It may be noticed that strict 6-dof tracking-by-synthesis based on point features has been studied in [24], though using purely synthetic models instead of image-based textured models.

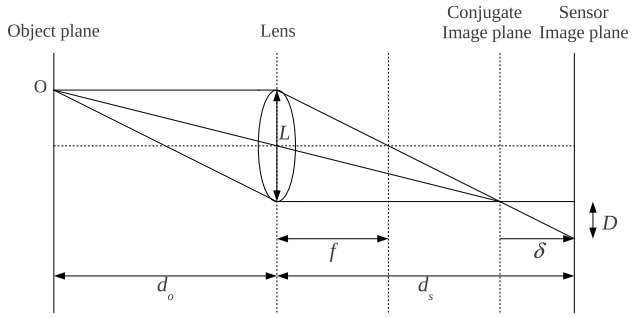


Figure 2: The ideal thin lens model.

2.2 Rendering With Depth of Field

Rendering with depth of field at interactive frame is an important and challenging task in computer graphics [21, 1, 7, 13]. Comprehensive surveys of available techniques are provided, for example in [7] and [1]. Using MIP mapping for fast generation of depth of field effects was first published by [21] and is particularly well suited for GPU-based approaches as noted by [7]. However, our problem is easier in some ways, because we only consider planar surfaces. But it is also more difficult in other ways, because the simulated blur must fit as closely as possible the depth blur really observed in the camera images, while only a visual effect is the goal in the referenced papers.

3 Preliminaries

3.1 Blurring Model

Depth blur is modeled using the ideal thin lens model [19], illustrated in figure 2: all the light rays emitted by an object point O in the scene and intercepted by the lens are refracted so that they converge to a point on the conjugate image plane. If the sensor image plane is not aligned with the conjugate image plane, the image point becomes a spot. For a circular diaphragm, the spot is a circle of diameter D called *circle of confusion* (CoC) or *blur circle*. Some simple geometric considerations provide the formula:

$$d_o = \frac{f d_s}{d_s - f - \frac{\delta}{|\delta|} D \frac{f}{L}}, \tag{1}$$

where d_o is the depth of the object point, f the focal length, L the diameter of the lens and d_s the distance between the lens and the sensor image plane. δ is the algebraic distance between the conjugate image plane and the sensor image plane, so that $\delta/|\delta| = 1$ or -1 depending on whether the sensor image plane is in front of or (resp.) behind the conjugate image plane.

The blurring effect finally obtained in the digital image depends on the transfer function of the optical system into the spatial domain. Traditionally in computer vision, this function is modeled as a Gaussian, so that:

$$I_b(i,j) = I_s(i,j) * G(i,j), \tag{2}$$

where I_b is the blurred image, I_s the sharp (focused) image, $*$ the bi-dimensional convolution product and G the bi-dimensional Gaussian function with spread parameter (standard deviation) σ:

$$G(i,j) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{i^2+j^2}{2\sigma^2}\right). \tag{3}$$

The relation between σ and the diameter D of the CoC is simply given by:

$$\sigma = kD, \tag{4}$$

where k is a proportionality constant depending on the camera. Introducing equation (4) in equation (1) and rearranging shows that the spread parameter σ is piecewise linear in the inverse depth $1/d_o$ of the object point :

$$\sigma(d_o) = \left| A\frac{1}{d_o} + B \right|, \tag{5}$$

where $A = kLd_s$ and $B = kL(1 - d_s/f)$. In practice, parameters k, L, d_s and f can not easily be obtained, but calibration methods exist to compute parameters A and B (see section 5).

The depth of the scene points that are in focus on the sensor image plane is called the *focus distance*. The focus distance depends on the distance d_s between the lens and the sensor image plane, and is limited by mechanical constraints. In the following, the curve $\sigma(d_o)$ obtained for a certain focus distance will be called the *depth response curve* (DRC) of this focus distance. The red curves in figure 4(b) show the DRCs obtained for different focus distances, using a standard webcam equipped with adjustable focus. Figure 4(b) also shows a horizontal line that corresponds to the maximum permissible CoC, that is the maximum diameter D_{max} of the CoC for which the image is "acceptably" sharp (see section 6). Intersecting this line with the DRCs shows that each focus distance leads to a range of depths of scene points that appear acceptably sharp in the image. This range of depths is called the *depth of field* (DoF) in the literature. As it is shown in figure 4(b), the DoF increases with the focus distance. We call *hyperfocal distance* the nearest focus distance at which the DoF extends to infinity, that is at which $\sigma(d_o) < D_{max}$ for all depths d_o greater than a certain depth.

3.2 MIP Mapping

MIP mapping is a well-known technique in computer graphics, intended to speed-up texture rendering and reduce aliasing artifacts [29]. It is implemented in the OpenGL 3D graphics API and is also available in OpenGL ES, a subset of OpenGL designed for embedded devices such as mobile phones, PDAs and video game consoles. The principle of this technique is as follows. When a two-dimensional texture map is applied (mapped) to a polygon, a pyramid of N prefiltered texture maps of decreasing resolutions, called *mipmap images*, is built. Typically, the original texture is at bottom of the pyramid (level 0) and the image size at level $i (1 \leq i \leq N-1)$ is half of the image size at level $i-1$ in each coordinate direction. When the textured polygon is viewed, the renderer switches to a suitable mipmap image, or interpolate between the two nearest, according to the apparent size of the texture and the perspective distortion.

More precisely, when the polygon is rendered, a pixel in the final image rarely corresponds to a pixel in the original texture image (called a *texel*): a rendered pixel can be part of a texel (the textel is enlarged), or, conversely, contain several texels (the texels are reduced). Suppose a unit square corresponding to one pixel inside the rendered polygon is warped to the original texture according to the perspective distortion. Let ρ be the size of the largest side of the bounding box of the obtained parallelepiped, and let λ be a real number defined by:

$$\lambda = \log_2 \rho. \tag{6}$$

When $\lambda > 0$, a texel reduction occurs, and the nearest integer value of λ is the level number of the nearest mipmap image in which the reduced texels can be extracted. The value of λ can also be used to interpolate between the two nearest mipmap images. In our OpenGL-based implementation, a trilinear interpolation is performed between the 2×4 nearest texels in the two nearest views.

4 Pyramidal Blurring

In this section, we assume that (i) the intrinsic parameters of the camera are known, (ii) the depth response curve of the camera has been calibrated (see section 5) and the focus is not changed during the application, (iii) a textured planar surface has been identified in one camera image (called the *original image* in the following) and (iv) the position and orientation of the plane with regard to the camera are known for that frame. In practice, these data can be obtained automatically using a fiducial marker, or interactively using such techniques as those described in [26].

A pyramid of square blurred images suitable for tracking-by-synthesis is generated from these data using the following process. Let's call *image i* the image at level i of the pyramid and let $r_i \times r_i$ be the dimension of image i. For any $i (1 \leq i \leq N-1)$, we take $r_i = r_{i-1}/2$. The choice of r_0 and N depends on the available memory size of the device and determines the scene depth range that will be properly handled by the algorithm.

4.1 Particular case

We first consider the particular case where the following two conditions are satisfied: (i) the scene plane is fronto-parallel to the camera at a depth d_{ori} and (ii) the scene plane is in focus in the original image, that is d_s is such that $\delta = 0$ and $d_o = d_{ori}$ in equation (1).

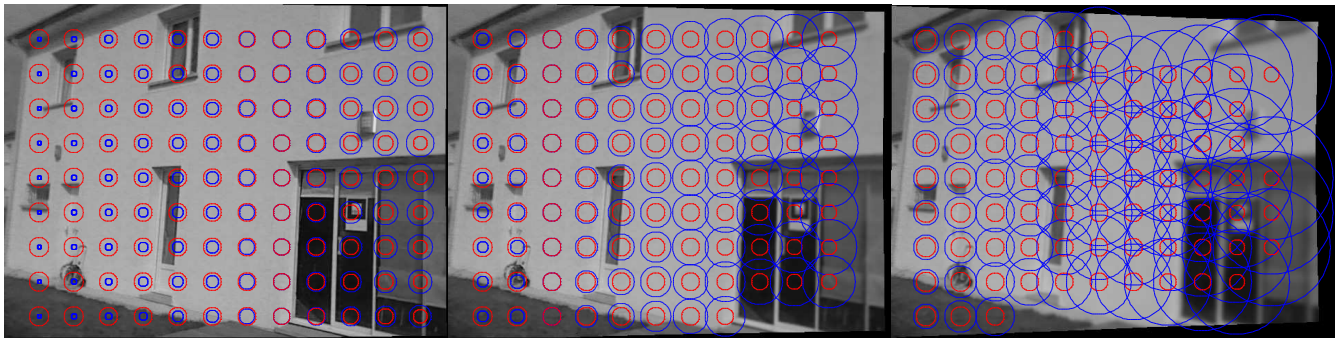
In that case, building the image at level i of the pyramid is straightforward: let $w \times h$ be the size of the original image and s_i the scale applied to that image so that the scaled image is fully contained in image i: $s_i = r_i / \max(w,h)$. As the scene plane is fronto-parallel to the camera, the scaled image can be seen as the image of the plane observed at distance $d_i = d_{ori}/s_i$. Then, as the plane is in focus in the original image, the amount of blur that has to be applied to each pixel of image i is simply given by $\sigma(d_i)$ where $\sigma()$ is defined in equation (5). Actually, in order to apply blur before resampling, a Gaussian blur of standard deviation $\sigma_i = \sigma(d_i)/s_i = \sigma(d_{ori}/s_i)/s_i$ is applied to the original image and the resulting image is resampled using bilinear interpolation.

In order to accelerate this procedure, cascade convolutions can also be applied to the original image [5], up and down the pyramid starting from the image having minimal σ_i. However, as the pyramid is built once and for all at the beginning of the tracking process, we do not necessary have to reach real-time for this process. By contrast, during tracking-by-synthesis, MIP mapping is performed in real-time using this pyramid, providing approximate but (as shown experimentally) realistic non uniform depth blur, without having to compute explicitly per-pixel ray depths.

4.2 General case

The two conditions mentioned in section 4.1 may be too restrictive for practical use. For instance, in [26], the scene is modeled by performing camera rotations at a fixed position and clicking the vertices of the faces through the screen center. The textured planed obtained using this procedure are generally not fronto-parallel. Moreover, using a fixed-focus lens makes it impossible to focus a plane at arbitrary distance. Even an adjustable focus lens has a maximal focus distance which can be smaller than the distance required to acquire, for instance, the whole façade of a large building. Our algorithm has therefore to be modified in order to be able to build the mipmap pyramid from a defocused non-fronto-parallel texture.

Without lose of generality, we assume that the camera pose $\mathbf{R}, \mathbf{t}$ associated with the original image is expressed in a world coordinate system where the equation of the plane is $z = 0$. Let $\mathbf{K} = \begin{pmatrix} f & 0 & u_0 \\ 0 & af & v_0 \\ 0 & 0 & 1 \end{pmatrix}$ be the intrinsic camera matrix, $\mathbf{P} = \mathbf{K}(\mathbf{R}|\mathbf{t})$ the projection matrix and $\tilde{\mathbf{P}}$ the 3×3 matrix made of the first two and the last columns of $\mathbf{P}$. $\tilde{\mathbf{P}}^{-1}$ is a planar homography that warps (rectify) the original image to the world plane, up to a scale. Let (x_{min}, y_{min}) and (x_{max}, y_{max}) be the lower left and (resp.) upper right corners of the bounding box of the shape obtained when applying $\tilde{\mathbf{P}}^{-1}$ to the boundaries of the plane in the original image. Let $\mathbf{S}_i$ be



$\Rightarrow 1024 \times 1024$ ($d_i = 8.6$m) $\Rightarrow 512 \times 512$ ($d_i = 17.2$m) $\Rightarrow 216 \times 216$ ($d_i = 34.5$m)

Figure 3: Blurring of the first image of the *House* sequence before perspective rectification for three levels of the mipmap pyramid. Red circles are proportional to the amount of blur already present in the original image, blue circles to the expected amount of blur.

a translation/scaling matrix defined by:

$$\mathbf{S}_i = \frac{r_i}{m} \begin{pmatrix} 1 & 0 & -x_{min} \\ 0 & 1/a & -y_{min}/a \\ 0 & 0 & m/r_i \end{pmatrix},$$

where $m = \max(x_{max} - x_{min}, (y_{max} - y_{min})/a)$. A rectified image at resolution r_i can be built using the planar homography $\mathbf{H}_i$ given by:

$$\mathbf{H}_i = \mathbf{S}_i \tilde{\mathbf{P}}^{-1}. \quad (7)$$

Let $\mathbf{Q}_i$ be the 3×3 matrix defined by

$$\mathbf{Q}_i = (\mathbf{q}_1 | \mathbf{q}_2 | \mathbf{q}_3) = \mathbf{K}^{-1} \mathbf{S}_i.$$

The matrix $\mathbf{Q}_i$ is, up to a scale s, the 3×3 submatrix (first two and last column) of the camera pose associated with the rectified image at resolution r_i. As $\frac{1}{s}\mathbf{q}_1$ and $\frac{1}{s}\mathbf{q}_2$ are columns of a rotation matrix, the scale is $s = ||\mathbf{q}_1|| = ||\mathbf{q}_2|| = r_i/(mf)$ and the depth associated with all pixels of this image is $d_i = \mathbf{q}_{33}/||\mathbf{q}_1||$, that is:

$$d_i = \frac{fm}{r_i}. \quad (8)$$

Homography $\mathbf{H}_i$ and depth d_i can now be used to compute the non uniform blur that has to be applied to the original image so that it is transformed to a uniform blur of standard deviation $\sigma(d_i)$ in the rectified image. Let us consider any pixel p in the original image. If the depth blur around this pixel is approximated by a Gaussian blur of standard deviation σ, then the depth blur around the pixel in the rectified image will be approximated by a Gaussian blur of standard deviation $\sigma_w = \rho\sigma$, where ρ is the size (as defined in section 3.2) of the warped pixel. As the convolution of two Gaussians of variance σ_1^2 and σ_2^2 results in a Gaussian of variance $\sigma_3^2 = \sigma_1^2 + \sigma_2^2$, the variance σ_p^2 of the blur to be applied to pixel p is therefore taken as:

$$\sigma_p^2 = \left(\frac{\sigma(d_i)}{\rho}\right)^2 - \sigma(d_p)^2, \quad (9)$$

where d_i is given by equation (8), $\sigma()$ is the DRC of the camera defined in equation (5) and d_p is the depth of the pixel p, obtained by back-projecting the pixel ray onto the world plane, using the pose corresponding to the original image. The term $\sigma(d_p)$ in equation (9) corresponds to the amount of blur we assume already present in the original image.

Equation (9) can only be used when the required amount of blur $\sigma(d_i)/\rho$ is greater than the already present amount of blur $\sigma(d_p)$. In the contrary case, no blur is applied to the original image at pixel p and the amount of blur obtained in the rectified image is higher than it should be. Tackling this issue is out of scope of this paper, and will be discussed in conclusion.

Once each pixel in the original image has been blurred, the resulting image is rectified using $\mathbf{H}_i$ (a bilinear interpolation is performed). Figure 3 illustrates the blurring procedure at three levels of the pyramid (images are shown before rectification). The red circles are proportional to the amount of blur $\sigma(d_p)$ already present in the original image and the blue circles to the expected amount of blur $\sigma(d_i)/\rho$. For the image to be warped at resolution 216×216 ($d_i = 34.5$m), all the red circles are inside the blue circles, which means that some amount of blur has to be added in the whole image, with increasing amount of blur from the background to the foreground. For the images to be warped at resolutions 512×512 ($d_i = 17.2$m) and 1024×1024 ($d_i = 8.6$m) the farthest part of the building is not blurred (blue circles are inside the red circles). The non-blurred region represents about a quarter of the image for $r_i = 512$ and about half of the image for $r_i = 1024$.

5 Calibration of the Depth Response Curves

Many methods have been proposed in the depth-from-focus/defocus literature [19, 30] to calibrate the DRC of a camera, that is, to compute the parameters A and B of equation (5). Usually, a linear regression is performed on several pairs of values $\{1/d_i, \sigma_i\}$, obtained by moving the calibration target over different places. The distances between the camera and the calibration target being controlled, the problem amounts to measuring the value of the spatial constant σ_i obtained in the resulting images. Different methods have being used, such as measuring the slope of the Laplacian at sharp discontinuities or comparing two images across different apertures e.g. using Fourier transforms [19]. When using camera motor systems, equation (5) can be arranged in order to put it in terms of motor counts, which are measurable and controllable, and therefore make the calibration of the DRC easier [30].

Our method differs from the literature in that (i) it can deal with any kind of lenses: fixed focus, manually adjustable focus and motorized focus, and (ii) it is free in terms of programming effort once the tracking-by-synthesis algorithm based on the method described in section 4 has been implemented. In a nutshell, rather than measuring σ_i directly in the images, we resynthesize each image i with different amounts of blur and take σ_i that provides the nearest image, in terms of matched features, to the real image.

5.1 Fixed-Focus Lenses

More precisely, the following procedure is used to compute the DRC of a fixed-focus lens. An image sequence is produced by moving the camera at different distances from a fronto-parallel textured plane containing an ARToolkit marker [11]. The first frame

of the sequence is assumed in focus and is used for texture mapping with different amounts of blur (typically between 0.05 and 3 with a step of 0.05). The marker is used to obtain that camera-plane distances d_i as well as the camera poses used to generate the rendered views of the blurred textures. Harris corners [9] are detected in both the real and the rendered images and feature matching is performed using NCC. For each standard deviation of the Gaussian blur, we count the number of inliers obtained by RANSAC computation of a planar homography, and take for σ_i the standard deviation that maximizes this number. As a result, we get a set of pairs $\{1/d_i, \sigma_i, \}$ from which a RANSAC linear fitting is performed. Actually, equation (5) is piecewise linear because of the absolute value. We therefore use values $\{1/d_i, \sigma_i\}$ when $d_i \geq d_f$ and $\{1/d_i, -\sigma_i\}$ when $d_i < d_f$, d_f being the distance of the focused (first) image of the sequence.

In this procedure, we need a point detector that is *not* robust against blur, otherwise the number-of-inliers criterion is not discriminating enough to enable determining σ_i. That is the reason why we chose to use the Harris corner detector, that has proven not robust against blur in our experiments (see section 6).

5.2 Adjustable-Focus Lenses

When using an adjustable-focus lens, we must be able to recompute the parameters A and B of the DRC each time the focus setting of the camera, that is the distance d_s in figure 2, is changed. From equation (5), we know that

$$A = Cd_s, \tag{10}$$

$$B = C - A/f, \tag{11}$$

where $C = kL$ is a camera constant. Parameters C and f can be estimated by computing the DRC parameters A_1, B_1 and A_2, B_2 of two different focus settings of the camera (obtained using the procedure described in section 5.1) and solving a system of two equations and two unknown provided by equation (11). Once C and f are known, it is possible to estimate the values A and B corresponding to any value of d_s, using equations (10) and (11).

Actually, it is usually not possible to get the distance d_s directly, but if the focus distance d_f between the camera and the scene object on which it is focused is known, writing equation (1) with $d_o = d_f$ and $\delta = 0$ provides:

$$d_s = \frac{d_f f}{d_f - f}. \tag{12}$$

It is therefore possible to recompute the DRC of the camera each time the focus setting is changed, assuming this change corresponds to a focus on a scene object whose distance to the camera can be estimated. In our implementation, an Artoolkit marker is used to estimate the focus distance, though any other measurement procedure could be used depending on the application context. Moreover, we show in section 6 that the repeatability of point detection can be significantly improved even with approximate focus distances and DRCs.

6 Experimental Results

All our experiments were performed using a Logitech QuickCam E3500 at resolution 640×480, equipped with a manually adjustable focus lens.

6.1 Calibration

As seen above, the calibration of the C and f values of the camera can be done using a single Artoolkit marker and two sequences starting at different focus distances. However, in order to assess the predictive property of the procedure described in section 5.2, we produced three sequences starting at different focus distances (10cm, 27cm, 65cm, resp.) and used three markers of different sizes (2cm, 8cm, 26.5cm, resp.). The small and medium markers were surrounded with postcards put on a table and the large marker was surrounded with posters taped on a closet (see Fig. 6, first two rows). The camera was moved closer and farther from the marker (depth ranges are 7-24cm, 15-67cm, 50-250cm, resp.) and for each sequence, a DRC was obtained using the procedure described in section 5.1. Figure 4(a) shows the pairs of values $(1/d_i, \sigma_i)$ obtained on the small marker sequence as well as the RANSAC fitting. Figure 4(b) shows, in red, the measured DRCs of the three sequences and, in green, the DRCs predicted at each focus distance, using the DRCs of the other two sequences as explained in section 5.2. In blue is represented the DRC corresponding to the maximum focus distance of the camera $d_{fmax} = 2050$mm, which is below the hyperfocal distance if we consider a conventional maximum CoC of 1/1250 of the image width (red horizontal line).

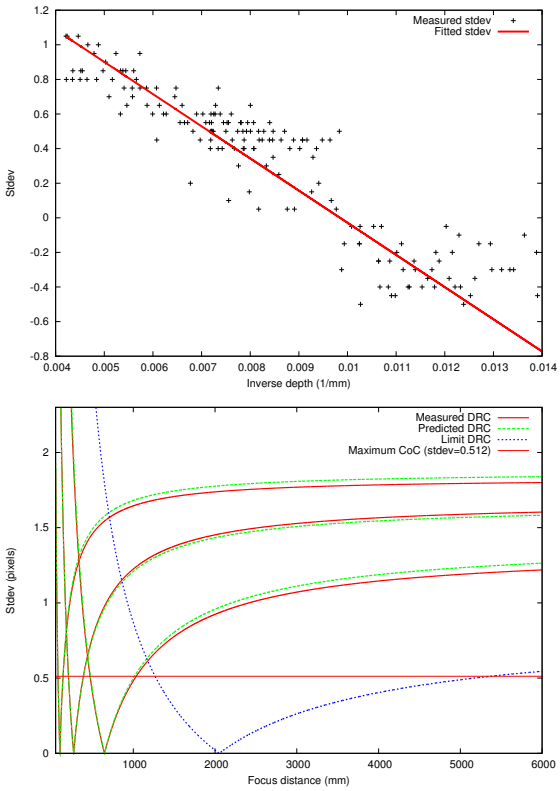


Figure 4: Calibration of the DRC prediction parameters. *Top to bottom:* (a) Linear fitting of pairs of values $(1/d_i, \sigma_i)$ obtained on the small marker sequence. (b) DRCs of the three calibration sequences (red), predicted DRCs based on the other two DRCs (green) and DRC corresponding to the maximum focus distance of the camera (blue). The red horizontal line corresponds to the estimated maximum CoC.

In the following, we will refer as "optimal DRC-prediction parameters" to parameters C and f obtained by solving equation (11) with the parameters A and B obtained in the three marker sequences.

6.2 Sensitivity against DRC accuracy

Sensitivity of point repeatability against accuracy of the DRC was estimated using a part of the large marker sequence, which is called the *Posters* sequence in the table 1. Five different DRCs have been tested: DRC 1 is the DRC directly measured from the large marker sequence (see section 6.1), DRC 2 is the DRC predicted at focus distance $d_f = 65$cm, using the prediction parameters obtained from

the small and medium marker sequences, DRC 3 is the DRC predicted at focus distance d_f, using the optimal DRC-prediction parameters. DRCs 4 and 5 correspond to erroneous DRCs that may be obtained in practical situations, due to the difficulty to visually assess whether the camera image is perfectly focused. Actually, the error that can be obtained is given by the intersection of the maximum-CoC line with the correct DRC, as shown in figure 5(a): at a focus distance of 65cm, we get an image that is visually similar, in term of sharpness, to the images obtained at distances $d_{min} = 47$cm and $d_{max} = 103$cm (47-103cm is the depth of field). DRC4 and DRC5 are therefore the DRCs predicted at focus distances d_{min} and (resp.) d_{max}, using the optimal DRC-prediction parameters.

A pyramid of blur is built for each DRC, and RANSAC matching of Harris corners is used for tracking-by-synthesis, according to the procedure described in the introduction of the paper. Table 1 (lines 4-8) shows the mean recall obtained on the whole sequence without blurring and with blurring, as well as the improvement ratio between these two values. The *recall* of a point detection between two images is defined as the number of points matched (that is repeated) between the two images, divided by the minimum number of points detected in these images. These results show that blurring the rendered texture is always beneficial, even with poorly estimated DRCs. The improvement ratios are between 50 and 60 % for all five DRCs, including the distant DRC 4 and 5. Figure 5(b) shows in more details the recall values obtained over the sequence with DRCs 3, 4 and 5 and without blurring. In order to make this figure more legible, the recall curves are drawn using very thin lines, and superimposed with thick smooth curves obtained by low-pass filtering of the curves in Fourier domain. One can see that the gap between the red curve (without blurring) and the other three curves is relatively high in all images of the sequence.

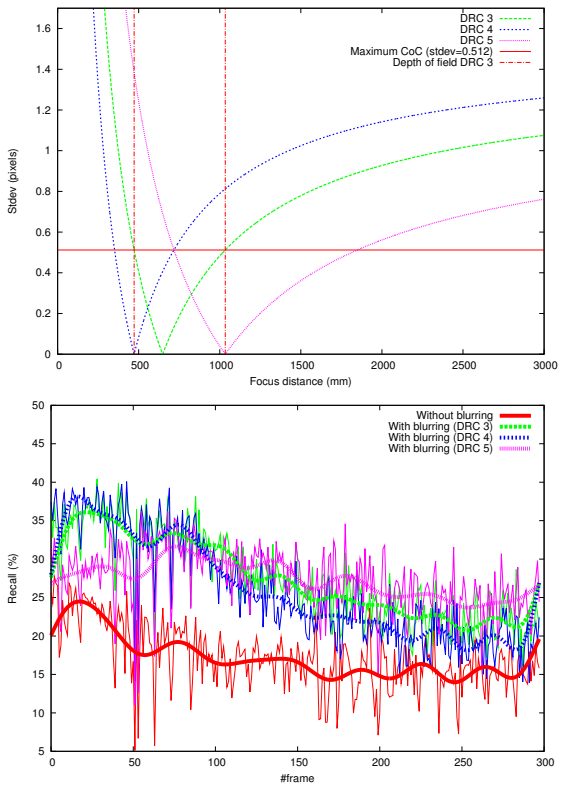


Figure 5: Sensitivity against DRC accuracy. *Top to bottom:* (a) DRCs obtained at the extremities of the depth of field (DRCs 4 and 5) and at focus distance (DRC 3) of the *Poster* sequence. (b) Recall values obtained over the *Poster* sequence using DRCs 3, 4 and 5 and without blurring.

6.3 Comparison of FAST, Harris and SURF across View Synthesis

Tracking-by-synthesis without and with pyramidal blurring was tested on several kinds of scenes, indoor and outdoor, from desktop to building size and with different amounts of texture. Figure 6 shows some snapshot of the used sequences, accompanied with information about the number of frames (from 298 to 796), focus distance (from 27cm to camera limit 2m) and scene depth range (from 10-44cm to 12-30m). The *Postcards* and *Posters* sequences are parts of the medium and (resp.) large marker sequences used to calibrate the DRC. They contain a lot of texture information and are easy to track. The *Carpet* and *Wall* sequences are much more difficult to track, the former due to the presence of repetitive patterns, the latter due to the rarity of texture information. The *House* and *Building* sequences show typical urban façades with relatively few texture information.

Tracking is initialized using an Artoolkit marker, and then is performed using the procedure described in the introduction. Point features are matched using NCC between Harris or FAST's pixel patches or SURF's description vectors. Outliers are discarded using a RANSAC computation of the homography between the two set of points. Only one plane is used in each experiment, whose texture is captured in the first image. Just before the process starts, the camera is focused as sharp as possible on the marker by turning the focus ring in the limit of camera capability. When the process starts, the focus distance d_f to the marker is computed using the marker, detected in the first image. If d_f is below the maximum focus distance d_{fmax}, the DRC is computed for d_f using the optimal DRC prediction parameters. Otherwise, the DRC corresponding to d_{fmax} is taken. It is important to note that the estimate of the DRC may also be done at a different time than just before tracking starts, providing that the focus ring is not changed between the two tasks.

The three interest point detectors that we chose to compare are FAST [22], Harris [9] and SURF [2], which are all widely used in vision-based camera tracking systems. FAST and Harris are corner detectors that are fast to compute (implementations exist on mobile phones) but only rotation invariant. SURF is a blob detector which is scale, rotation and almost viewpoint invariant, but slower to compute. Even if an efficient implementation of SURF has been proposed, bringing 30% speed-up to the original algorithm [4], performance is still not good enough for mobile phones. However, one main advantage of tracking-by-synthesis is that invariance to viewpoint is obtained without need of viewpoint invariant features, due to the fact that the camera and the rendered images to be matched are generally close to each other. It would therefore be very useful to be able to use FAST or Harris in tracking-by-synthesis, providing that the performance of these detectors is similar to that of SURF in term of repeatability across view synthesis.

Actually, this is not the case when no blur is added to the rendered views, as shown in table 1 (a dash is used when tracking failed using the related detector). The mean recall of SURF is always significantly higher than for the other two detectors, except for the *Building* sequence where it is comparable (though low) to the mean recall obtained by Harris. Actually, this result is not very surprising because it has been shown, e.g. in [10], that SURF is relatively repeatable across blur. Another result shown in table 1 is that FAST seems less resistant to blur than Harris. However, an important result of this paper is that when adding blur to the rendered image, the repeatability of FAST and Harris can be greatly improved. By contrast, the repeatability of SURF does not improve, which is, again, not surprising, for the same reason as just mentioned. As a result, Harris reaches mean recalls that are similar to those of SURF, and

Figure 6: Snapshots of the sequences used in our experiments.

even much greater for the *Building* sequence (though much lower for the *Wall* sequence). The recall of FAST generally stays significantly below the recall of Harris. Figure 7 shows the recall obtained in all images of the *Posters* sequence: results are conform, all along the sequence, to the general tendency we just described.

Finally, virtual cube was added in the scene and tracking-by-synthesis was performed using Harris corner detector and pyramidal blurring. The snapshots in figure 6 show that the cube is rendered properly over a variety of viewpoints, which demonstrates that our system performs drift-free tracking under large scale viewpoint and perspective changes, and is therefore suitable for AR applications.

7 Conclusion and Future Works

The results of this work can be considered as a preliminary study to demonstrate that tracking-by-synthesis using point features is viable, and can run at high speed by combining fast corner detection and pyramidal blurring. In particular, we have shown that using Harris corners with image blurring is equivalent, in term of repeatability across view synthesis, to using SURF blobs without blurring. In addition, we have proposed a practical method to calibrate the depth response curve of the optical blur at a given focus distance.

This work will be added to our interactive system [26] that allows online reconstruction of several textured planar surfaces, which are immediately used for camera tracking. Initialization of tracking-by-synthesis using a GPS and an inertial sensor will be tested. Moreover, as mentioned in section 4, a limit of pyramidal blurring is that shaper textures can not be obtained from more blurred textures. However, in an interactive system, we can envision updating the mipmaps on-the-fly as closer views of the textures become available. Nevertheless, this will have to be done carefully if we do not want to introduce drift in the process. Finally, repeatability across view synthesis may be improved by simulating other sources of differences between the camera images and the rendered images, such as illumination changes and motion blur.

Table 1: Comparison of the repeatability of FAST, Harris and SURF without and with addition of blur in the rendered images.

Sequence	Point detector	Mean recall (%) wo blurring	Mean recall (%) w blurring	% impr
Postcards	Harris	15.6	19.8	26.7
Postcards	FAST	12.2	9.3	-23.9
Postcards	SURF	24.0	23.9	-0.4
Posters - DRC 1	Harris	17.2	27.3	58.6
Posters - DRC 2	Harris	17.2	27.1	57.9
Posters - DRC 3	Harris	17.2	27.5	60.0
Posters - DRC 4	Harris	17.2	26.1	51.6
Posters - DRC 5	Harris	17.2	27.4	59.2
Posters	FAST	9.9	15.1	53.7
Posters	SURF	28.2	26.1	-7.3
Carpet	Harris	6.5	7.0	7.5
Carpet	FAST	4.8	5.0	3.8
Carpet	SURF	-	-	-
Wall	Harris	3.8	6.1	60.4
Wall	FAST	-	-	-
Wall	SURF	15.9	16.1	1.2
House	Harris	9.6	13.8	43.7
House	FAST	7.2	13.8	91.7
House	SURF	11.34	12.0	6.1
Building	Harris	7.0	14.6	109.0
Building	FAST	4.9	10.6	119
Building	SURF	7.6	7.0	-7.4

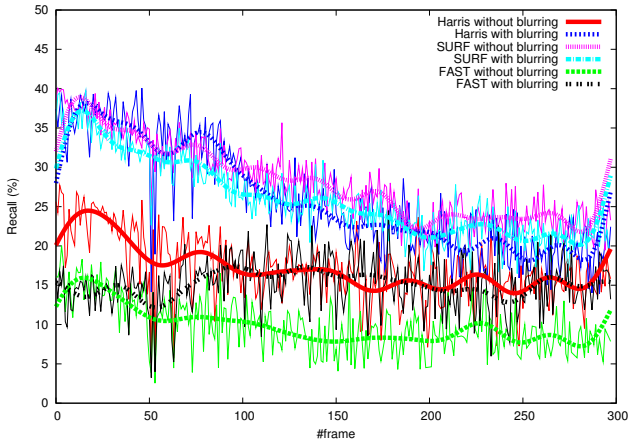


Figure 7: Recall values obtained in the *Posters* sequence.

REFERENCES

[1] B. A. Barsky, D. R. Horn, S. A. Klein, J. A. Pang, and M. Yu. Camera models and optical systems used in computer graphics: part i and ii. In *International conference on Computational science and its applications*, pages 256–265, Berlin, Heidelberg, 2003. Springer-Verlag.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *European Conference on Computer Vision*, vol. 3951 of *Lecture Notes in Computer Science*, pages 404–417, 2006.

[3] G. Bleser and D. Stricker. Advanced tracking through efficient image processing and visual-inertial sensor fusion. In *IEEE Virtual Reality Conference*, pages 137 –144, 2008.

[4] W.-C. Chen, Y. Xiong, J. Gao, N. Gelfand, and R. Grzeszczuk. Efficient extraction of robust image features on mobile devices. In *Int. Symposium on Mixed and Augmented Reality*, pages 287–288, 2007.

[5] J. Crowley and O. Riff. Fast computation of scale normalised gaussian receptive fields. In *Scale Space Methods in Computer Vision*, vol. 2695 of *Lecture Notes in Computer Science*, pages 584–598. 2003.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007.

[7] J. Demers. Depth of field: A survey of techniques. In *GPU Gems, Fernando R., (Ed.), Addison Wesley*, pages 375–390, 2004.

[8] S. Diverdi, J. Withert, and T. Hllerert. Envisor: Online environment map construction for mixed reality. In *10th IEEE International Conference on Virtual Reality*, 2008.

[9] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *4th Alvey Conference*, Cambridge, 1988.

[10] L. Juan and O. Gwon. A Comparison of SIFT, PCA-SIFT and SURF. *International Journal of Image Processing*, 3(4):143–152, 2009.

[11] H. Kato and M. Billinghurst. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. In *2nd International Workshop on Augmented Reality, San Francisco*, 1999.

[12] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Int. Symp. on Mixed and Augmented Reality*, 2007.

[13] M. Kraus and M. Strengert. Depth-of-field rendering by pyramidal image processing. In *Proceedings Eurographics*, 2007.

[14] V. Lepetit, L. Vachetti, D. Thalmann, and P. Fua. Fully Automated and Stable Registration for Augmented Reality Applications. In *Int. Symposium on Mixed and Augmented Reality*, pages 93–101, 2003.

[15] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 60(2):91–110, 2004.

[16] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *7th international joint conference on Artificial intelligence - Volume 2*, pages 674–679, San Francisco, 1981. Morgan Kaufmann Publishers Inc.

[17] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *Int. Journal of Computer Vision*, 60:63–86, October 2004.

[18] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):448–461, march 2010.

[19] A. P. Pentland. A New Sense for Depth of Field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):523–531, Apr. 1987.

[20] G. Reitmayr and T. W. Drummond. Going out: Robust tracking for outdoor augmented reality. In *International Symposium on Mixed and Augmented Reality*, pages 109–118, Santa Barbara, 2006.

[21] P. Rokita. Fast generation of depth of field effects in computer graphics. *Computers & Graphics*, 17(5):593–595, 1993.

[22] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, pages 430–443, 2006.

[23] G. Schall, D. Wagner, G. Reitmayr, E. Taichmann, M. Wieser, D. Schmalstieg, and B. Hofmann-Wellenhof. Global pose estimation using multi-sensor fusion for outdoor augmented reality. In *8th IEEE International Symposium on Mixed and Augmented Reality*, pages 153–162, Washington, 2009.

[24] M. Schumann, S. Achilles, and S. Müller. Analysis by Synthesis Techniques for Markerless Tracking. In *Virtuelle und Erweiterte Realität, 6. Workshop der GI Fachgruppe VR/AR, Braunschweig*, 2009.

[25] J. Shi and C. Tomasi. Good features to track. In *IEEE Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[26] G. Simon. In-Situ 3D Sketching Using a Video Camera as an Interaction and Tracking Device. In Eurographics 2010, Norrköping, 2010.

[27] G. Simon, A. Fitzgibbon, and A. Zisserman. Markerless tracking using planar structures in the scene. In *International Symposium on Augmented Reality*, pages 120–128, 2000.

[28] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg. Real-time panoramic mapping and tracking on mobile phones. In *IEEE Virtual Reality Conference*, pages 211–218, 2010.

[29] L. Williams. Pyramidal parametrics. In *10th annual conference on Computer graphics and interactive techniques, ACM SIGGRAPH*, pages 1–11, New York, 1983.

[30] Y. Xiong and S. A. Shafer. Depth from Focusing and Defocusing. In *IEEE Conference on Computer Vision and Pattern Recognition*, New York, pages 68–73, 1993.