[POSTER] A Particle Filter Approach to Outdoor Localization using Image-based Rendering

Christian Poglitsch† Clemens Arth‡ Dieter Schmalstieg§ Jonathan Ventura§
Graz University of Technology University of Colorado Colorado Springs

ABSTRACT
We propose an outdoor localization system using a particle filter. In our approach, a textured, geo-registered model of the outdoor environment is used as a reference to estimate the pose of a smartphone. The device position and the orientation obtained from a Global Positioning System (GPS) receiver and an inertial measurement unit (IMU) are used as a first estimation of the true pose. Then, multiple pose hypotheses are randomly distributed about the GPS/IMU measurement and use to produce renderings of the virtual model. With vision-based methods, the rendered images are compared with the image received from the smartphone, and the matching scores are used to update the particle filter. The outcome of our system improves the camera pose estimate in real time without user assistance.

Index Terms: I.3.7 [Computer Graphics]: Virtual Reality; I.4.8 [Image Processing and Computer Vision]: Scene Analysis-Tracking; J.6 [Image Processing and Computer Vision]: Monte Carlo; C.5.3 [Computer System Implementation]: Microcomputers-Portable Devices (e.g., laptops, personal digital assistants)

1 INTRODUCTION
Outdoor Augmented Reality (AR) applications today rely mostly on the built-in sensors to provide localization. Today's smartphones achieve self-localization by Global Positioning System (GPS), along with a digital compass and inertial orientation sensors like gyroscope or accelerometer. Due to limited accuracy, AR applications cannot rely solely on the sensors of smartphones. Therefore, alternative methods must be used to replace or assist the sensors.

Vision-based methods for global localization are the most promising, because of their high precision in estimating the camera pose and their ability to run in real time on modern hardware [17]. In this work we apply image-based rendering to evaluate hypotheses from a particle filter and determine the most likely camera pose.

2 RELATED WORK
There are two dominant approaches to image-based localization. The first requires knowledge of the scene beforehand [6, 4, 8, 14]. In this approach, vision-based localization is solved by (a) using a collection of images or (b) a 3D model as reference. The second approach is Simultaneous Localization and Mapping (SLAM), where the camera is localized while the system builds a map of the unknown scene.

Structure from Motion (SfM) approaches enable the creation of large-scale 3D models of urban scenes. These compact scene representations can be used for accurate image-based localization. As the scene gets larger, recognizing unique landmarks becomes more challenging. This difficulty is overcome by using sophisticated image features such as Scale-Invariant Feature Transform (SIFT) [9], but these are too expensive to compute in real-time [14, 6, 8, 18].

Reitmayr and Drummond [11, 12] present an edge-based tracking system using textured 3D building models for accurate tracking on a smartphone. They make use of the video image as well as gyroscope and measurements of gravity and magnetic field. This work mainly addresses frame-to-frame tracking, but also includes an approach for re-localization using features extracted using the FAST corner detector [13].

Aubry et al. [2] demonstrate a technique that can reliably align images of architectural sites, like drawings, paintings or historical photographs, to a 3D model of the site. To align different query inputs like drawings to a 3D model, local feature matching based on interest points, e.g., SIFT, often fails to find correspondences across paintings and photographs. Sibbing et al. [15] explore how point cloud rendering techniques can be used to create virtual views from which one can extract features that match image-based features as closely as possible.

Our approach also uses rendering of a textured model from many possible poses; however, in contrast to Reitmayr and Drummond [11, 12], we use pixel-wise cost functions instead of edge search and apply our cost functions to continuous particle filtering, rather than one-shot localization. Thus, our approach unifies localization and continuous tracking in a single method.

3 SYSTEM OVERVIEW
Our approach requires matching of a video stream from a smartphone to a virtual model. Therefore, our system uses a virtual model and sensor readings from a smartphone as input data.

The rendering of the virtual environment requires a geo-registered model of the environment and geo-registered panorama images for texturing the model.

Data collected from the smartphone includes the video stream and device pose as estimated by the non-visual sensors. The device delivers latitude and longitude from the GPS unit in the WGS84 datum and orientation as yaw, pitch and roll angles from the inertial measurement unit (IMU). In our experiments we run the particle filter algorithm on a desktop computer.

4 PARTICLE FILTER
A particle filter is a simulation-based method for tracking a system with partially observable state. We briefly review particle filters in this section. The reader is referred to Thrun et al. [16] for a more detailed review. A particle filter maintains a weighted and normalized set of sampled states, $S_t = \{x^i_t, w^i_t\}_{i=1,\ldots,N}$, called particles $p_i^t$, where $x^i_t$ is the state of the $i^{th}$ particle and $w^i_t$ is its weight. The weight of a particle can be thought of as the probability that the particle’s state corresponds to the true state of the system. In our case, the state is the camera pose. In our system, one iteration is processed for every new frame when it arrives. The steps of one particle filter iteration are as follows:

\*e-mail:chris01@sbox.tugraz.at
\*e-mail:christina.chlo@tugraz.at
§e-mail:arth@icg.tugraz.at
‡e-mail:schmalstieg@tugraz.at
§e-mail:chris01@sbox.tugraz.at
‡e-mail:jventura@uccs.edu
1. Each particle is propagated according to our propagation model.
2. Their weights are computed by comparing rendered and real
   world frames.
3. The top ranked particles are re-sampled to obtain a new set
   of equally weighted particles, which approximates the new
   posterior distribution.

4.1 Motion model
In this discussion, we refer to the device pose delivered by the
internal GPS/IMU sensors as the “sensor pose.”

Each particle \( p^i \) is formed by perturbing a prior pose by a sample
from a statistical motion model. Similar to the work of Klein and
Murray [7], we sample these motions from a Gaussian noise model.

For the initial step we propagate particles as follows:

\[
p^i_{t,d} = c_{i,d} + \mathcal{N}(0, \sigma_d)
\]

where \( p^i_{t,d} \) is the particle, \( c_{i,d} \) the \( d \)-th dimension of the sensor pose \( c_i \)
of the \( p \)-th frame and \( \sigma_d \) the standard deviation of the noise added for
that dimension. For the following re-sampling steps, \( c \) is replaced by
the state of one of the most probable particles from the previous
iteration.

\[
p^i_{t,d} = p^i_{t-1,d} + \mathcal{N}(0, \sigma_d)
\]

where \( p^i_{t,d} \) is a particle for the current frame, \( p^i_{t-1,d} \) a probable
particle of the previous frame, \( d \) the dimension and \( \sigma_d \) the standard
deviation of the noise added for that dimension.

For each frame, changes in the sensor data are measured. The
most probable particles \( p^i_{t-1,d} \) are duplicated and modified with
the sensor changes between the current and the previous frame.

\[
p_t = p^i_{t-1,d} + (c^+_i - c^-_i)
\]

where \( c^-_i \) is the sensor pose of the previous frame and \( c^+_i \) the sensor
pose of the current frame. The unmodified sensor pose is also added
as a particle.

In summary, for maximum robustness, our particle re-sampling
includes (a) the most likely particles from the previous iteration (b)
the most likely particles, modified with changes between the current
and last frame according to the sensors and (c) the unmodified
sensor pose.

4.2 Calculating particle weight
To compute the weight of a particle, we compare an image rendered
at the particle’s pose to the current image from the device camera.

We use two different weighting approaches in order to capture
color, texture and edge information.

The image sum of squared distances (SSD) method compares the
rendered image to the image captured from the smartphone, with a
mask applied. We mask out the part of the rendering corresponding

to the sky (pixels not belonging to any scene geometry), specified as
a binary mask \( m_j \in \{0, 1\} \). We compute the SSD for each particle
\( p_i \) as follows:

\[
e^i_j = \sum_j m_j \cdot ||I_i - V_{i,j}||^2 \sum_j m_j
\]

where \( e^i_j \) is the error, \( I \) is the camera image, and \( V_j \) is the rendered
image for particle \( i \).

The gradient magnitude SSD method computes the average
difference between the gradient magnitudes of the rendered image
and the masked gradient magnitudes of the camera image as follows:

\[
e^i_G = \sum_j m_j \cdot (||G^i_j|| - ||G^i_j'||)^2 \sum_j m_j
\]

where \( e^i_G \) is the error, \( G^i_j \) contains the gradient magnitudes of the
camera image, \( G^i_j' \) contains the gradient magnitudes of the rendered
image for particle \( i \), and \( m' \) is the binary mask after applying a dilation
operator to avoid masking out the contours of the buildings.

The gradients are computed using the Sobel operator applied to the
rendered and the real world image.

The error value for each method is normalized so that

\[
\sum_{i=0}^{N} e^i_j = 1, \quad \sum_{i=0}^{N} e^i_G = 1
\]

The weights are then calculated as

\[
w^i_j = \exp(-e^i_j), \quad w^i_G = \exp(-e^i_G)
\]

and normalized so that

\[
\sum_{i=0}^{N} w^i_j = 1, \quad \sum_{i=0}^{N} w^i_G = 1
\]

The final weight is calculated simply as the sum of the weights from
the two methods:

\[
w^i = w^i_j + w^i_G
\]

The particle with the highest combined weight \( w^i \) is considered
the best estimate of the device pose, i.e., the pose to be used for AR
rendering.

Before computing the error functions and weights, all images un-
der consideration are downsampled. This helps to avoid or reduce
artifacts and to improve performance.
Table 1: Noise parameters for the initialization step.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>$\sigma_d$</th>
<th>Approx. range</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{\text{eye}}, y_{\text{eye}}$</td>
<td>5.5</td>
<td>up to 9 meters</td>
</tr>
<tr>
<td>$x_{\text{center}}, y_{\text{center}}$</td>
<td>0.25</td>
<td>up to 15 degrees</td>
</tr>
<tr>
<td>$x_{\text{center}}, x_{\text{up}}, y_{\text{up}}, z_{\text{up}}$</td>
<td>0.085</td>
<td>up to 5 degrees</td>
</tr>
</tbody>
</table>

Table 2: Noise parameters for the update steps.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>$\sigma_d$</th>
<th>Approx. range</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{\text{eye}}, y_{\text{eye}}$</td>
<td>0.36</td>
<td>up to 60 centimeters</td>
</tr>
<tr>
<td>$x_{\text{center}}, y_{\text{center}}$</td>
<td>0.01</td>
<td>up to 0.5 degrees</td>
</tr>
<tr>
<td>$x_{\text{center}}, x_{\text{up}}, y_{\text{up}}, z_{\text{up}}$</td>
<td>0.01</td>
<td>up to 0.5 degrees</td>
</tr>
</tbody>
</table>

5 Experiments

The test areas for our particle filter system are the Hauptplatz and Tummelplatz in Graz, Austria. For recording GPS, IMU data and the video stream we used an Apple iPhone 4. To process the particle filter, we used a desktop computer with an Intel Core i7-4770, 16 GB RAM and an AMD Radeon R9 280X.

5.1 Implementation notes

This section provides some details about the particle filter configuration. Each frame has a width of 100 pixels, whereas the height depends on the aspect ratio of the smartphone image. Our system is initialized with 1000 particles. The chosen noise parameters for the initial particle distribution governed by Equation 1 are given in Table 1.

In the subsequent update iterations, the three most probable particles and the sensor pose $c_i$ are used for re-sampling. The chosen noise parameters for the particle motion model governed by Equation 2 are given in Table 2. In total, the update step uses only 20 particles. For future work we consider to use Kullback-Leibler distance (KLD)-sampling to decide if the resampling step is necessary. This approach adjusts the number of samples based on the likelihood of observations [5].

5.2 Results and discussion

We tested our system with many images and videos recorded in our test areas. Figure 2 shows that we achieve accuracy within 50 cm on a geo-referenced image. After initialization (669 ms), the system runs at 30Hz. The rate of convergence would likely be increased by using more particles in the update steps, but in our tests we used the maximum number of particles possible to achieve real-time rates with our current unoptimized implementation.

Figure 3 shows sampled frames from the test videos alongside renderings of the virtual environment, comparing the best pose from the particle filter to the uncorrected sensor pose. As can be clearly seen in these examples, our approach significantly improves the camera pose and offers much better real-virtual alignment than the raw camera pose delivered by the GPS and IMU sensors. We also found that our system is robust to image artifacts like shadows and the presence of objects that are not in the virtual environment, like cars or people. Examples of occluding objects are shown in the second row of Figure 3, where occluding people, street cars, and street carts are present in the query image.

In most cases, the rendering of the best particle closely matches the input camera image. However, in Figure 2, some jitter in the pose can be observed. We found that, because the 3D model is textured with projective texture mapping, the localization result is negatively affected if the camera’s viewing angle differs greatly from the viewing angle of the images used to make the 3D model. The nature of the random sampling process in the particle filter can also cause jittering in stationary scenes, for example, when two particles have similar weights, but slightly different poses.

Figure 2: Result of particle filter iterations on a geo-referenced image.

Our experiments indicate that SSD is less effective if a significant amount of sky is visible, causing the number of pixels under consideration to decrease and leading to a high weight. Conversely, gradient magnitude SSD works best when parts of the building and parts of the sky are visible. This is because the highest gradients are at the contour of the buildings. Thus the two weights are complementary and it makes sense to combine them.

Although our system as tested is implemented on a desktop computer, we are confident that the approach could easily achieve real-time rates on mobile hardware. This is because most of the computation consists of either rendering of textured models with low polygon count or simple image processing operations with small filter size.

6 Conclusions and future work

Our work has demonstrated an accurate outdoor localization system that runs in real time. The particle filter is flexible in its ability to adapt to different computation environments and different desired levels of performance, accuracy and robustness, by simply adjusting the number of particles and propagation model used.

Another interesting property of our system is that both localization and continuous tracking are achieved with the same method. In addition, we do not make any use of slow-to-compute feature descriptors or large descriptor databases and, instead, only require a simple textured model of the target environment. Therefore, a mobile version of the application may be feasible in the near future.

A disadvantage of the proposed system is increased jitter in stationary scenes. This condition occurs when two particles with a slightly different pose have a similar weight. One possible solution is to back-project feature points to the synthetic model in order to obtain 3D-2D matches that can be used to run a much more stable and faster tracking [1].

One avenue of future work is to improve the particle propagation model. A more accurate distribution function could provide even further performance, because fewer particles would be necessary. For example, a pedestrian motion model could be used [10]. Another improvement would be view-dependent rendering [3] to increase the accuracy of the image-based rendering and the range of the localization.

Our work hints at the possibility that image-based localization
based on sparse descriptors may in the long run be replaced by
dense GPU-based methods. In particular, the increasing availability
of high-quality urban models created with structure-from-motion
methods, supported by large geo-data providers such as Google or
Microsoft, can be leveraged in GPU-based “tracking by synthesis”
approaches such as the one explored in this paper.

Figure 3: The input data for a pose query is an image from the smartphone (left) and sensor data from GPS and gyroscope. The device sensors
are used as an initial guess (right). Our system provides an improvement of the initially guessed camera pose (middle). Note the robustness to
issues such as occluding objects (second row) and lighting (third row).

ACKNOWLEDGEMENTS

This work was supported in part by the Christian Doppler
Gesellschaft (CD Labor Handheld Augmented Reality).

REFERENCES

tenance operations using automatic markerless augmented reality sys-
tem. In Mixed and Augmented Reality (ISMAR), 2011 10th IEEE In-
via discriminative visual elements. ACM Trans. Graph., 33(2):14:1–
14:14, Apr. 2014.
Image-Based Rendering with Projective Texture-Mapping. Rendering
camera tracking. In Computer Vision, 2009 IEEE 12th International
S. Becker, and Z. Ghahramani, editors, NIPS, pages 713–720. MIT
motion point clouds to fast location recognition. In Proceedings of
Computer Vision and Pattern Recognition (CVPR), 2009.
[8] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition us-
ing prioritized feature matching. In K. Daniilidis, P. Maragos, and
N. Paragios, editors, ECCV (2), volume 6312 of Lecture Notes in
localization in unmodified environments with inexpensive sensors. In
ing for outdoor augmented reality. In Mixed and Augmented Reality,
[12] G. Reitmayr and T. Drummond. Initialisation for visual tracking in
urban environments. In Mixed and Augmented Reality, 2007. ISMAR
[13] E. Rosten and T. Drummond. Fusing points and lines for high perfor-
International Conference on, volume 2, pages 1508–1515 V ol. 2, Oct
2005.
using direct 2d-to-3d matching. In Proceedings of International Con-
ference on Computer Vision (ICCV), 2011.
dering. In 3D Vision - 3DV 2013, 2013 International Conference on,
pages 56–63, June 2013.
2005.
[17] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. Global local-
ization from monocular slam on a mobile phone. IEEE Transactions
[18] J. Ventura and T. Hollerer. Wide-area scene mapping for mobile vi-
sual tracking. In Mixed and Augmented Reality (ISMAR), 2012 IEEE