Bostanci *et al. Hum. Cent. Comput. Inf. Sci.* (2015) 5:20
DOI 10.1186/s13673-015-0040-3

Human-centric Computing and Information Sciences
a SpringerOpen Journal

RESEARCH Open Access

Augmented reality applications for cultural heritage using Kinect

Erkan Bostanci[1*], Nadia Kanwal[2] and Adrian F Clark[3]

*Correspondence:
ebostanci@ankara.edu.tr
[1] Computer Engineering Department, Ankara University, Ankara, Turkey
Full list of author information is available at the end of the article

Abstract

This paper explores the use of data from the Kinect sensor for performing augmented reality, with emphasis on cultural heritage applications. It is shown that the combination of depth and image correspondences from the Kinect can yield a reliable estimate of the location and pose of the camera, though noise from the depth sensor introduces an unpleasant jittering of the rendered view. Kalman filtering of the camera position was found to yield a much more stable view. Results show that the system is accurate enough for in situ augmented reality applications. Skeleton tracking using Kinect data allows the appearance of participants to be augmented, and together these facilitate the development of cultural heritage applications.

Keywords: Pose estimation, Kinect, Augmented reality, Cultural heritage

Introduction

Augmented Reality (AR) combines real world imagery with synthetic content generated by a computer. The first comprehensive review of AR [1] identified a broad range of applications of this technology, including medicine, manufacturing and robot path planning. Subsequently, AR has been applied in cultural heritage [2] such as the reconstruction of ancient Olympia in Greece [3, 4]. Although it is reported that AR enriches human perception [5] in general, the principal reason for performing AR reconstructions in cultural heritage is that the owners of sites are usually reticent to permit physical reconstructions in situ so that the archaeology remains undisturbed for future generations [3].

Developments in multimedia technology facilitate the learning experience in cultural heritage [6] with the aid of improved user interaction methods. Developed models or virtual tours in reconstructions of archaeological sites (e.g. [7, 8]) provide entertaining means of learning. However, ex situ reconstructions such as models and movies are difficult to visualize in the context of the archaeological remains. AR reconstructions can be produced in situ with minimal physical disturbance, an attractive property, even though they may take a significant time to develop [9].

There are several forms that AR reconstructions may take, and the work reported here is directed towards a kind of 'historical mirror,' in which virtual buildings are built around flat surfaces visible in the real world and human participants are clothed appropriately for the historical period. The literature presents examples of using Kinect for cultural heritage [10–12]. It was used as a 3D scanner in the work given in [10] and as a

motion tracker to navigate in virtual reality reconstructions in [11, 12]. User experience for large screens with Kinect in a cultural heritage setting, such as in an exhibition, has been demonstrated in [13].

This paper uses Kinect to establish 3D world and 2D image correspondences and, from them, determine camera pose. It then finds planar objects within the real world and augments them, the aim being to render the appearance in antiquity in front of real-world features. The advantage of this work over the system in [14], which tries to augment synthetic objects over camera images, is that no offline phase is needed to create a map of the environment: data from the Kinect is used *in real time* to augment the appearance of flat surfaces in the real world. Moreover, this can be achieved without GPU programming, particularly promising for systems that require low power or are mobile.

Using the human tracking functionality of the OpenNI library [15], humans within the environment are identified and their appearances are also augmented; existing AR systems in the cultural heritage domain do not attempt the latter, yet it is essential if participants are truly to experience a sense of presence following the ideas presented in [16] and [17].

The rest of the paper is structured as follows: "Background" presents the use of vision-based methods for augmented reality and gives examples from the literature making use of Kinect for cultural heritage. "In situ augmentation" describes a method for determining the camera pose and registering 3D models with real world objects. "Augmenting participants" then describes the use of skeleton tracking to augment participants with clothes etc. "Creating 3D models" describes the modelling phase, while "Results" assesses the effectiveness of the in situ augmentation algorithm and presents the results of augmentation. Finally, conclusions are drawn in "Conclusions".

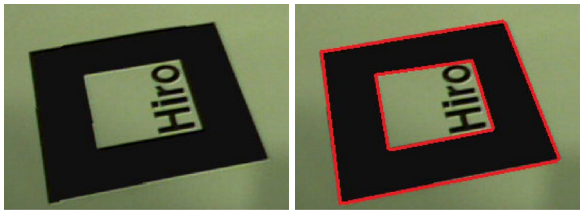
Background

The principal problem faced in AR applications concerns the accuracy of user tracking and, consequently, the registration of 3D models with real-world features [1, 18, 19]. When the 3D structure of the environment (feature positions, and internal/external parameters of the camera, etc.) is known, sufficient accuracy can be obtained to match virtual and real objects seamlessly; as a rule of thumb, this involves identifying the direction of gaze to ~1° in azimuth and elevation, and locating the position to ~0.1 m.

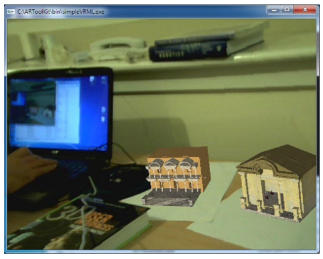
It is sensible to use a camera to perform tracking since the images from the real-world can be used both for rendering the synthetic content and finding the viewpoint of the user [20]. This sort of vision-based tracking can be classified into two groups namely marker-based and markerless methods [19].

A marker is a distinguishable element placed in the environment so it can be identified apart from other objects [21]. Once the marker is detected then the pose of the camera can be obtained and models can be rendered on the camera image. An example of this usage is shown in Figure 1 where ARToolkit[a] is used to track a marker to place 3D models in the scene.

Placing markers is not always practical especially when the concerns on preserving the original structure of the ruins are taken into account. In this case, markerless methods can be used to extract features which are already present in the environment. These features can be points [22], lines [23] or higher-level geometric structures such as



a Fiducial marker **b** Marker detected



c Augmented view

Figure 1 Stages of augmentation using markers.

planes [24]. It is important to note that these natural features should be robust and stable so that the tracking can be performed accurately.

Estimation of the camera pose in 3D is a widely-studied research topic [25, 26], normally obtained using vision-only methods such as the Perspective-n-Point (PnP) solution, which aims to recover camera pose using the positions of 3D features and their projections. For instance, [14] recently used OpenCV's PnP functionality for real-time AR: In a preliminary (offline) phase, a map of features was first created from known 3D positions in the real world, with each point being described by a SURF descriptor [27]. Subsequently, an online system matched SURF features from the camera to those of the map, and then the camera pose was calculated. The time taken for calculating and matching SURF features is significant, so this had to be performed on a GPU.

The advent of the Kinect [28] permits both colour and depth of a scene to be sensed concurrently. This allows extracting the 3D structure of the surrounding and creating a representation of the environment and the use of Kinect data with the PnP algorithm has already been investigated [29] for image-based registration. The analysis involved determining relative and absolute accuracies of the Kinect and another sensor according to the camera pose obtained by using Efficient PnP (EPnP) algorithm [30].

Kinect has also been used in the context of cultural heritage. For instance, Remondino [10] presented a review of using different types of imaging and depth sensors including Kinect to perform 3D scanning of archaeological objects for the purposes of digital recording, historical documentation and preservation of cultural heritage. Richards-Rissetto et al. [11] used Kinect's body motion detection features to perform navigation in a 3D reconstructed model of an ancient Maya city using virtual reality.

In addition to the usages described above, this innovative sensor also provides new opportunities in terms of on-site learning described in [6] by presenting the learners an interesting visualization of structures overlaid on top of the ruins as we aim in this study.

In situ augmentation

Since the viewpoint information is required for augmentation, the first step is to establish correspondences between 2D image features and their depth since, from them, the camera pose can be calculated. With that knowledge, rectangular features in the image are identified and augmented with columns, as this is a common requirement in cultural heritage reconstructions in the eastern Mediterranean as shown in Figure 2.

Generating 3D–2D correspondences

The distance between the two sensors of the Kinect results in different projected locations in the RGB and the depth images for the same 3D point. To compensate for this, the calibration parameters of both sensors must be estimated, or errors will ensue [33]. These calibration parameters can be computed using the Kinect Calibration Toolbox [34], or obtained using a method [35] similar to stereo calibration [36]. The first method provides a semi-automatic way of calibrating the camera, by both allowing the user to select corners manually from a calibration target (a chess-board pattern) in the RGB image and the plane where the calibration target is placed in the depth image; these selections are used as an initial guess for the calibration parameters for a set of RGB and depth images (~30 images). A non-linear minimization method is then used to reduce the projection errors. This work uses the calibration parameters obtained by the second method [35], which again finds corners on the calibration target and performs the calibration only for the RGB camera. Then the depth camera is also calibrated by manually finding corresponding corners in the depth image.

The transformations (**R** and **T**) between the depth and RGB sensors of the Kinect are used to align the two sensors and are obtained as calibration results along with the intrinsic parameters (focal lengths and distortions) for both sensors. The camera matrices $\mathbf{K_D}$ and $\mathbf{K_C}$ for the depth and RGB sensors respectively are:

$$\mathbf{K_D} = \begin{bmatrix} f_{x_D} & 0 & c_{x_D} \\ 0 & f_{y_D} & c_{y_D} \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$\mathbf{K_C} = \begin{bmatrix} f_{x_C} & 0 & c_{x_C} \\ 0 & f_{y_C} & c_{y_C} \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where f_x and f_y are the focal lengths in the x and y directions and c_x and c_y are the centre coordinates of the relevant sensor.

a Temple in ruins **b** A reconstruction **c** 3D model

Figure 2 Views of the Hadrian Temple in Ephesus in Turkey. **b** Obtained from [31, 32].

Figure 3 illustrates the process of back-projecting raw depth values into 3D coordinates and then projecting them onto the RGB image.

Given raw depth (d) values from the depth image captured by the Kinect, the world coordinates of a 3D point ($\mathbf{p} \equiv (\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z)^T$) are calculated using

$$\begin{aligned} \mathbf{p}_x &= \left(x \times c_{x_D}\right) \times \frac{d_m}{f_{x_D}} \\ \mathbf{p}_y &= \left(y \times c_{y_D}\right) \times \frac{d_m}{f_{y_D}} \\ \mathbf{p}_z &= d_m \end{aligned} \tag{3}$$

where x and y denote a position in the depth data. d_m is a conversion from raw depth values (d) to depth in meters [37] and is calculated as:

$$d_m = 0.1236 \times \tan\left(\frac{d}{2842.5} + 1.1863\right) \tag{4}$$

Unreliable values (those having the maximum raw disparity value of 2047 [29], as illustrated in Figure 3) are discarded and the remainder stored.

Using the transformation and distortion (used for modelling radial and tangential distortions arising from the inaccuracies in the manufacturing process) parameters for the Kinect sensor [24], the 3D points are projected onto the RGB image [36]. For a 3D point $\mathbf{p}$, first the transformation parameters ($\mathbf{R}$ and $\mathbf{T}$) are applied to $\mathbf{p}$ to find $p' \equiv (p'_x, p'_y, p'_z)^T$:

$$p' = \mathbf{R}\mathbf{p} + \mathbf{T} \tag{5}$$

from which the projected coordinates (p'_x, p'_y) are calculated (p'_z is ignored) using

$$\begin{aligned} p'_x &= \frac{p'_x}{p'_z} \\ p'_y &= \frac{p'_y}{p'_z} \\ p'_z &= \frac{p'_z}{p'_z} = 1 \end{aligned} \tag{6}$$

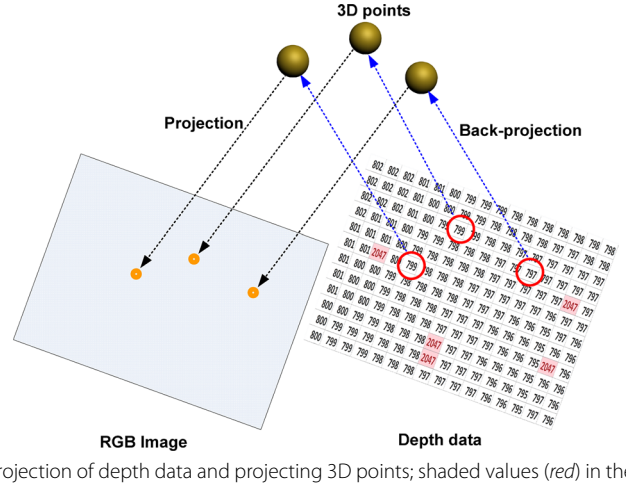


Figure 3 Back-projection of depth data and projecting 3D points; shaded values (*red*) in the depth data show unreliable raw disparities.

The radial (k_1, k_2, k_3) and tangential (p_1, p_2) distortion parameters of the RGB camera are applied to get $p'' \equiv (p''_x, p''_y)^T$:

$$\begin{aligned} p''_x &= p'_x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ &\quad + 2p_1 p'_x p'_y + p_2\left(r^2 + 2{p'_x}^2\right) \\ p''_y &= p'_y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ &\quad + p_1\left(r^2 + 2{p'_y}^2\right) + 2p_2 p'_x p'_y \end{aligned} \tag{7}$$

where $r = \sqrt{{p'_x}^2 + {p'_y}^2}$. Finally, the pixel locations are obtained using the camera matrix $\mathbf{K_C}$:

$$p''' = \begin{bmatrix} p'''_x \\ p'''_y \end{bmatrix} = \mathbf{K_C} \begin{bmatrix} p''_x \\ p''_y \end{bmatrix} \tag{8}$$

The 3D points resulting from this back-projection of the depth sensor and their corresponding projection points are stored in a data structure (Table 1) that allows the 3D position corresponding to a 2D pixel (p''') to be obtained quickly.

Finding the camera pose

The camera position must be found relative to the 3D points obtained from the Kinect sensor, and that is best performed by tracking reliable image features over several frames. The set of points used for tracking needs to be stable in order to achieve robust localization of the camera. To achieve this, the FAST feature detector [38] was used to find features in the RGB image. This detector produces features that are repeatable [39] and widely scattered across the image [40], important characteristics if the resulting homography matrix is to be accurate, and the system is to operate at video rate. The key-points obtained from FAST were described using the BRIEF descriptor [41], which is also known to be robust and operate at video rate. The binary structure of the BRIEF descriptor allows two descriptors to be matched using *XOR* instructions, and is therefore rapid to execute.

After the initial set of features has been obtained, these points are matched against the features detected in subsequent frames, outliers being rejected using RANSAC [42]. 3D information for matched points is obtained from data structure (Table 1) alluded to in the previous section. There can be cases where a pixel position may not have an associated 3D datum, perhaps due to reflection of the infra-red beam or the disparity between the depth sensor and camera; when this happens, the closest 2D point in the

Table 1 Data structure to store point data

Index	Data
$\left\langle p'''_{x_1}, p'''_{y_1} \right\rangle$	$\mathbf{p_1}$
$\left\langle p'''_{x_2}, p'''_{y_2} \right\rangle$	$\mathbf{p_2}$
⋮	⋮
$\left\langle p'''_{x_n}, p'''_{y_n} \right\rangle$	$\mathbf{p_n}$

data structure is used and the corresponding depth calculated as the mean of depths within a 21×21-pixel region.

Algorithms for calculating the position and orientation of the camera are well-established [43, 44], provided that the intrinsic parameters (focal length etc.) are known through calibration. Correspondences between 3D points and their 2D projections are used in order to recover the camera pose. For three correspondences, four possible solutions can be found, whereas a unique solution can be found for six or more correspondences [45]. The corresponding 2D and 3D points stored in Table 1 are used to calculate the camera position using a recent PnP solution known as EPnP [30, 46] as the initial estimate.

As discussed in [44], PnP solutions are easily affected by noise, and this manifests itself as an unpleasant jittering of the camera position and hence rendered imagery. To reduce this, the estimate of the camera pose is filtered. As will be shown below, a sliding window filter of size 15 reduces this effect but does not eliminate it, while a Kalman filter [47] was found to be more effective. The state of this filter comprises the x, y, z coordinates of the camera position and their velocities V_x, V_y and V_z. Measurements of (x, y, z) are obtained from the EPnP algorithm discussed above. The velocities were initialized to 0.5 *units*/*frame* for the three axes, reasonable for a user standing and observing ancient ruins (i.e., little or no motion). The transition matrix F is:

$$F = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

At each frame, first the transition matrix F is applied to the current camera position as the prediction step (Δt is the time passed between two consecutive frames). Then the measurements for the camera pose calculated using EPnP are used to refine this prediction for updating the filter, and the state is used for the viewpoint of augmentation. It was found that this filter reduced jittering to the point where it was imperceptible.

Finding objects for augmentation

After determining the camera pose, the next step is to find planar objects, to be augmented, in this work, by synthetic column models. Whether columns are cylindrical or fluted, their projection in the 2D image will be rectangular, and such shapes are relatively easy to detect: Any visual noise in the image was reduced by Gaussian smoothing, then the Canny edge detector [48] was applied to each of the colour channels independently. Contours were extracted from the edge images using an approximation method [49]; contours that contain four vertices with angles between pairs of lines joining these vertices close to 90° form reliable rectangles (Figure 4). Small rectangles and rectangles having inappropriate aspect ratios for columns were rejected.

The rectangles detected using the method described above tend to disappear and reappear from frame to frame due to the changes in lightning conditions. For this reason, the vertices of these rectangles were tracked using the *Condensation* algorithm [50], essentially a particle filter for visual tracking supporting a multi-modal estimate. For a

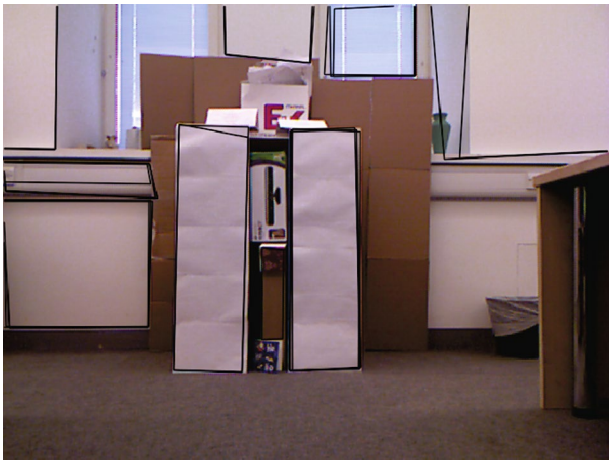


Figure 4 Rectangular features (marked in *black*) located within an image.

rectangle centred at $\mathbf{X} = (x_c, y_c)^T$, the initial state is initialized from the rectangle detector and the particles were scattered within a 20-pixel radius of that point. The state $\mathbf{X}$ was updated using two models, one dynamical and the other observation.

The dynamic model $\mathbf{A}$ generates the predicted (hypothetical) state $\hat{\mathbf{X}}$ and is initialized to the identity matrix for simplicity:

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{10}$$

At each frame the predicted state is calculated for all particles $(1 \ldots n)$:

$$\hat{\mathbf{X}}_n = \mathbf{A}\mathbf{X}_n \tag{11}$$

Particle weights are updated using the observation model. Measurements $(\mathbf{M} = (m_{c_x}, m_{c_y}))$ from the rectangle detection algorithm are used to refine the particle confidences using

$$\begin{aligned} w_{i_x} &= e^{\frac{-1}{2\sigma_x^2}(m_{cx} - p_{i_x})^2} \\ w_{i_y} &= e^{\frac{-1}{2\sigma_y^2}\left(m_{cy} - p_{i_y}\right)^2} \\ w_i &= w_{i_x} \times w_{i_y} \end{aligned} \tag{12}$$

where w_i denotes the weight of particle i and σ_x^2 and σ_y^2 are the variances of the samples for the x and y coordinates of the centre points. The hypothetical values of the rectangle centre are stored in p_{i_x} and p_{i_y}. From (12), it can be seen that a Gaussian probability distribution is used for the update of the particle confidences.

This process is followed by re-sampling, in which the new confidences will be used to create a new set of particles of the same size [36]. The number of particles for the algorithm was selected as $N = 50$, which was found to be sufficient (i.e., not causing particle deprivation [51]). At each frame, the particles are updated using the result of the rectangle detection algorithm described above, resulting in the centre of the rectangle being tracked robustly as depicted in Figure 5. After the re-sampling update is performed, the

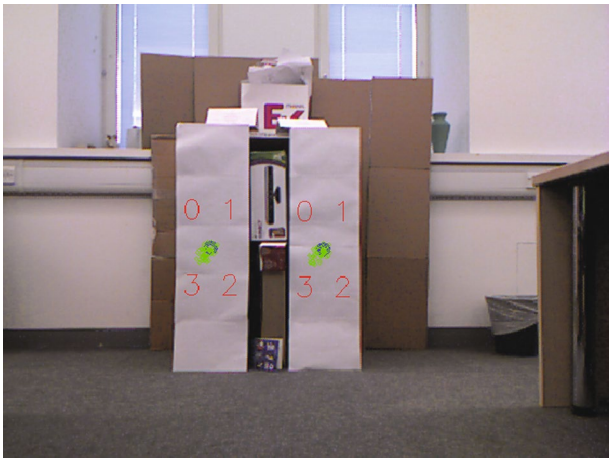


Figure 5 Tracking selected *rectangles*. *Small circles* show the estimates of their centres whereas the *large circle* is the particle with largest weight.

particle with the largest weight is selected both for augmentation and propagation into the next state. The 3D position of each rectangle's centre is retrieved and the column models are rendered at these 3D positions.

The complete augmentation algorithm is given in Algorithm 1.

Algorithm 1 *In situ* augmentation

Require: I_C: RGB image, I_D: Depth image, 3D models for augmentation.
Load models and camera calibration parameters.
Initialize the Kalman filter and rendering environment.
for all frames **do**
  if first frame **then**
    Extract the initial set of features for tracking using FAST detector and BRIEF descriptor.
    Find the initial number of rectangles for augmentation, initialize the Condensation filter.
  else
    Compute 3D points from the depth data.
    Calculate projections and create the hash-map.
    Find feature matches by detecting, describing and matching features from the RGB image for the new frame.
    Calculate camera pose using EPnP.
    Update the Kalman filter for camera pose using measurements and set viewpoint for augmentation.
    Detect rectangular objects.
    Update Condensation filter.
    Retrieve centre coordinates for augmentation.
    Render the view.
  end if
end for

Augmenting participants

This section describes the use of the skeleton tracking features of Kinect to augment a participant with clothes from ancient times. The OpenNI library [15] processes depth

information from Kinect and performs detection of body parts identified as joints, as shown in Figure 6. In this work, three 'joints' are used: the head, torso and right hand, for superimposing a *galea* (Roman helmet), toga and a sword respectively.

As each joint is recognized by OpenNI, its position and orientation are returned. These transformations are absolute, not relative, and so can be used directly for rendering, with the exception that the rotation matrix for orientation R_M must converted to a vector R using Rodrigues' formula:

$$R = \begin{bmatrix} R_y \\ R_x \\ R_z \end{bmatrix} = \begin{bmatrix} \sin^{-1} R_{M_{0,2}} \\ \frac{\sin^{-1} -R_{M_{1,2}}}{\cos R_y} \\ \frac{\sin^{-1} -R_{M_{0,1}}}{\cos R_y} \end{bmatrix} \tag{13}$$

When using optical see-through displays, images from the real environment are obtained automatically using a mirror in the display device; when the graphics are rendered, augmentation comes at no cost [52]. However, when using video see-through displays or a camera as the input source, the images of the real environment must also be rendered together with other graphics in order to generate the final image. The approach commonly adopted is to convert camera images to the texture format of the rendering engine (Irrlicht in our case); then, at each call to the function that draws the whole scene, the camera image is first rendered and then the computer models are superimposed on it.

Creating 3D models

The approach followed for creating the models used for AR applications is presented briefly in this section.

Modelling and optimization

The ancient columns and other models were created using 3D Max [53]. Buildings were modelled [8] in accordance with reconstruction images prepared by

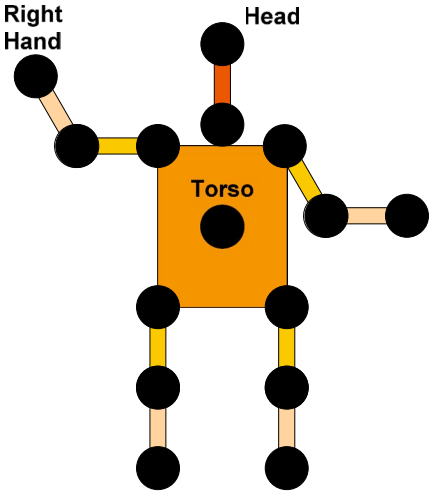


Figure 6 Skeleton joints identified by OpenNI.

archaeologists [31, 32]. AutoCAD [53] was used to draw 2D profiles of buildings and other structures such as columns as shown in Figure 7a. These profiles were later exported to 3D Max. 3D objects are created using modifiers such as bevel profile which rotates the given profile around a boundary (square, circle etc.) to get a 3D object (Figure 7b).

3D models having a large number of vertices (hence and hence faces) reduce the frame rate of a real time application. For this reason, created models were optimized using the *MultiRes* modifier, which works by first computing the number of vertices and faces in a model, then allows the user to eliminate some of them manually. This method proved to be effective for 3D models consisting of thousands of faces.

Texture baking

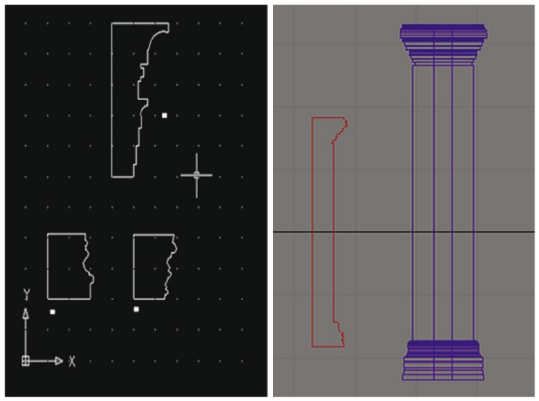
Texture baking (also known as "render to texture") [54] is the process of creating a single texture map from multiple materials that have been applied to a model. There are several ways to perform this; the following method was found to produce the best results.

To produce a single texture from several materials applied to a model, one first uses an *Unwrap UVW* modifier to store the current material map—this defines how the texture must wrap around an object that has a complex structure. The next step is to unwrap all the individual faces of the model as shown in Figure 8 and render the material information into the output texture.

When rendering, a diffuse map was selected (instead of the complete map model, which included maps for lighting or surface normals and failed to create the texture for the invisible side of the model). Later, this single texture was applied to the model again, by removing the previous material. Finally, the stored map must be applied to obtain the original look of the model before texture baking.

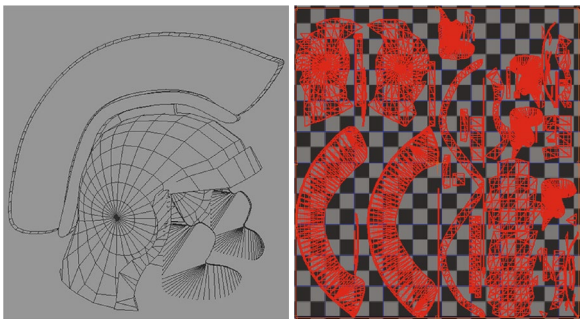
Results

This section first presents the created models and then presents a quantitative analysis of the camera pose estimation. It finally shows the effects of the complete augmentation process which can run at video rates (25 frames per second).



a Column profiles **b** 3D column created from the profiles

Figure 7 Creating 3D models of the columns in front of the Hadrian temple by spinning its profile around a circle based on the structure of the ruins and the reconstruction image in Figure 2.

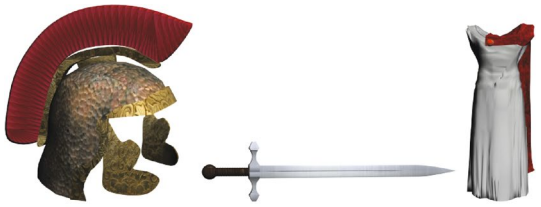


a Wireframe view of the model **b** Faces unwrapped

Figure 8 Unwrapping the faces of a model for texture baking.

Models used in the experiments

The models created are shown in Figure 9. The optimization step resulted in a 10−25% reduction in the number of faces for the models. Usually, the more complex the model is, the higher the percentage of faces can be removed without a noticeable change in appearance; here, these optimizations were most efficient in terms of removing faces for



a *Galea* (Roman helmet) **b** Sword **c** Toga

d Top arch

e Column

Figure 9 Models created for the augmenting participants.

the *galea* model (Figure 9a) due to its roughly spherical shape (many faces are required for a smooth surface, hence many can be removed during optimization) whereas the column (Figure 9e) was unable to accommodate the removal of many vertices without losing detail of the capital (top) or pediment (base) (Figure 7a).

Performance of the in situ augmentation algorithm

The errors in the initial estimation are shown in Figure 10. The relative error is calculated for the initial set of 3D–2D correspondences by re-projecting 3D points using the estimated translation and rotation for the camera, while the true re-projection error is calculated using ground truth, obtained from the knowledge that the camera is stationary. A mean difference of $\simeq$0.26 pixels was obtained between the true and estimated re-projection errors, showing that the estimation is reasonably accurate.

Again using the ground truth, the calculated errors in rotation and translation and are given in Figure 11. (Calculating the rotation error involves converting the actual and estimated rotation matrices into quaternions and finding the distance between the two quaternions. For the translation error, the Euclidean distance is calculated between the actual and estimated translations.) Fluctuations are substantially higher for the translational error ($\sigma_{t_{error}} = 0.009$) compared to the rotational error ($\sigma_{r_{error}} = 0.0004$). As

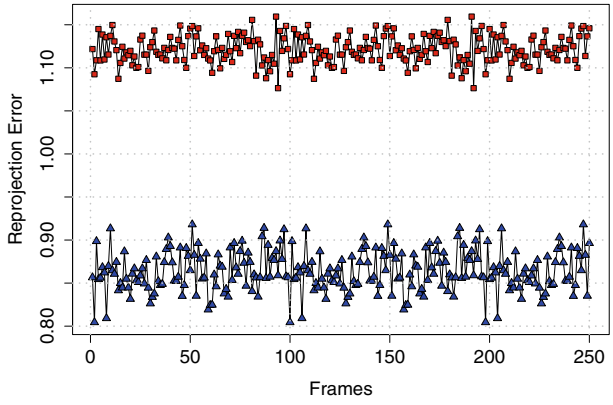


Figure 10 Estimated (*red*) and true (*blue*) re-projection errors calculated with EPnP.

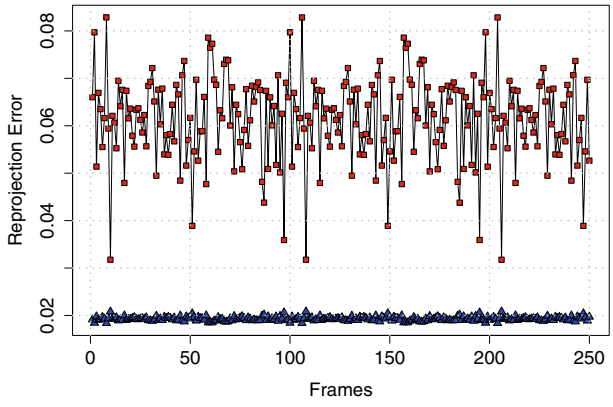
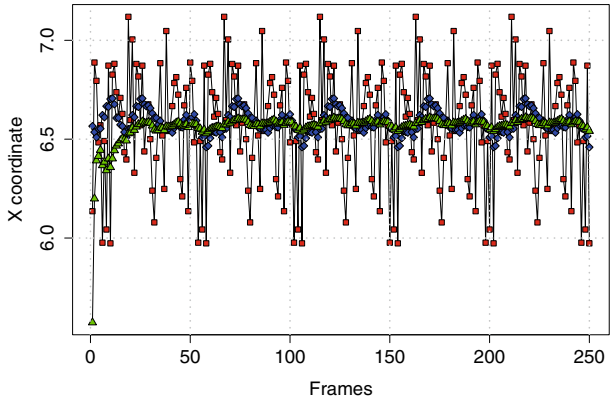


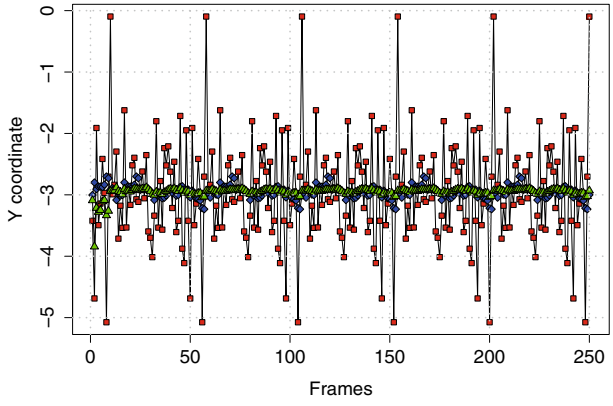
Figure 11 Rotational (*blue*) and translational (*red*) errors for the camera position estimate.

alluded to above, the magnitude of this translational error results in rendered augmentation being unstable ('jittering').

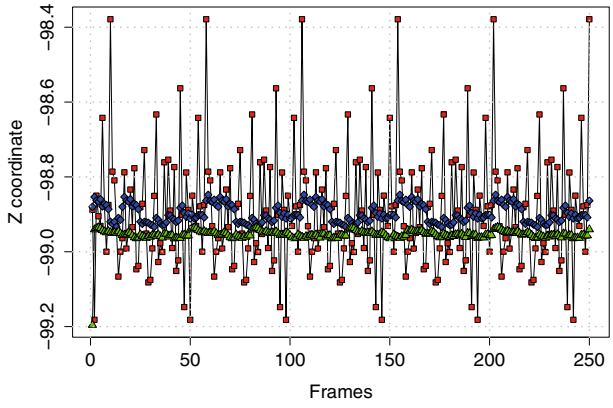
Figure 12 shows the results from sliding window and Kalman filters, and it is clear that the Kalman filter produces more stable positions. The jitter is reduced from $\bar{\sigma}_{raw} = 0.43$



a Estimated x coordinate



b Estimated y coordinate



c Estimated z coordinate

Figure 12 Camera coordinate estimations using raw estimation result (*blue*), sliding window (*red*) and Kalman filter (*green*).

to $\bar{\sigma}_{Kalman} = 0.07$, where $\bar{\sigma}_{raw}$ and $\bar{\sigma}_{Kalman}$ are the standard deviations of the camera coordinates for the initial estimate and Kalman filtered results.

Augmentation results

To illustrate the augmentation algorithms presented above, an application was developed to augment rectangular regions of a specific size and aspect ratio in the Kinect imagery. When the camera position and orientation had been found and the centres of suitable rectangles identified, the 3D column models described in the previous section were rendered in front of them. When rectangles were found at are particular distance apart, an arch could be placed above the columns to form an arch, as shown in Figure 13.

The result of augmenting users is shown in Figure 14a for a single user and in Figure 14b for two users. The general effect is acceptable for toga and sword but some minor registration errors are apparent for the galea model in the case of a single user. These registration errors become more severe when multiple users are present, and this appears to be due to the accuracy of skeleton tracking decreasing when the user is not centred in the field of view.

Conclusions

It is clear that the Kinect has great potential in AR applications for cultural heritage. Different studies have shown that Kinect can be used as an imaging and depth sensor in order to scan and record archaeological objects [10]. It can also be used as an interaction device tracking the movements of users [11].

In this paper, we presented two novel usages of Kinect within the context of cultural heritage. The first contribution is the in situ AR application using the algorithm described in "In situ augmentation" which allows the camera pose to be estimated reasonably precisely and robustly from Kinect imagery and depth values. Planar, rectangular regions can be identified robustly through condensation and augmented with features from historical buildings. The second contribution is the use of Kinect data to

Figure 13 Augmenting columns over *rectangles*.

a Augmenting a single user with toga, galea and sword

b Augmenting two users

Figure 14 Augmenting participants.

augment the appearance of humans, so that they can be clothed in a way that matches their surroundings.

The analysis and augmentation presented in this paper can be achieved in real time using a single computer equipped with a Kinect. The lack of any set-up phase (as opposed to [14]) and this speed of processing are important practical considerations for installations in museums that are to be used without supervision, or for use in educational games. Indeed, we believe that AR applications similar to ones presented in this paper will improve the on-site learning experience [6] and provide people with an incentive to learn about their and other people's past and protect our historical artefacts and monuments as a memory of the past.

Endnote

[a]http://www.hitl.washington.edu/artoolkit

Authors' contributions

EB designed and coded the application in addition to drafting the manuscript. NK helped in coding and testing the developed algorithm as well as designing the test cases. AFC supervised the project and improved the presentation of the paper. All authors read and approved the final manuscript.

Author details
[1] Computer Engineering Department, Ankara University, Ankara, Turkey. [2] Lahore College for Women University, Lahore, Pakistan. [3] School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK.

Compliance with ethical standard

Competing interest
The authors declare that they have no competing interests.

Received: 4 December 2014 Accepted: 10 July 2015
Published online: 22 July 2015

References

1. Azuma R (1997) A survey of augmented reality. Presence Teleop Virtual Environ 6:355–385
2. Papagiannakis G, Singh G, Magnenat-Thalmann N (2008) A survey of mobile and wireless technologies for augmented reality systems. Comput Anim Virtual Worlds 19:3–22
3. Stricker D, Kettenbach T (2001) Real-time and markerless vision-based tracking for outdoor augmented reality applications. In: International symposium on augmented reality
4. Dahne P, Karigiannis J (2002) Archeoguide: system architecture of a mobile outdoor augmented reality system. In: IEEE international symposium on mixed and augmented reality, pp 263–264
5. Ribo M, Lang P, Ganster H, Brandner M, Stock C, Pinz A (2002) Hybrid tracking for outdoor augmented reality applications. IEEE Comput Graph Appl 22(6):54–63
6. Hawkey R, Futurelab N (2004) Learning with digital technologies in museums, science centres and galleries. NESTA Futurelab Series, Futurelab Education
7. Chiara R, Santo V, Erra U, Scanarano V (2006) Real positioning in virtual environments using game engines. In: Eurographics Italian chapter conference
8. Koyuncu B, Bostanci E (2007) Virtual reconstruction of an ancient site: Ephesus. In: Proceedings of the XIth symposium on mediterranean archaeology, pp 233–236. Archaeopress
9. Wolfenstetter T (2007) Applications of augmented reality technology for archaeological purposes. Tech. rep., Technische Universität München
10. Remondino F (2011) Heritage recording and 3d modelling with photogrammetry and 3d scanning. Remote Sens 3(6):1104–1138
11. Richards-Rissetto H, Remondino F, Agugiaro G, Robertsson J, vonSchwerin J, Girardi G (2012) Kinect and 3d gis in archaeology. In: International conference on virtual systems and multimedia, pp 331–337
12. Pietroni E, Pagano A, Rufa C (2013) The etruscanning project: gesture-based interaction and user experience in the virtual reconstruction of the regolini-galassi tomb. In: Digital heritage international congress, vol 2, pp 653–660. IEEE
13. Ioannides M, Quak E (2014) 3D research challenges in cultural heritage: a roadmap in digital heritage preservation. Lecture notes in computer science/information systems and applications, incl. Internet/Web, and HCI. Springer, Berlin. https://books.google.com.tr/books?id=Mf-JBAAAQBAJ
14. Tam DCC, Fiala M (2012) A real time augmented reality system using GPU acceleration. In: 2012 ninth conference on computer and robot vision (CRV), pp 101–108
15. OpenNI: OpenNI library. http://openni.org/. Last access Nov 2013
16. Bostanci E, Clark AF (2011) Living the past in the future. In: International conference on intelligent environments, international workshop on creative science, pp 167–172
17. Bostanci E, Kanwal N, Clark AF (2013) Kinect-derived augmentation of the real world for cultural heritage. In: UKSim, pp 117–122
18. Noh Z, Sunar MS, Pan Z (2009) A review on augmented reality for virtual heritage system. In: Proceedings of the 4th international conference on e-learning and games: learning by playing. Game-based Education System Design and Development, pp 50–61
19. Bostanci E, Kanwal N, Ehsan S, Clark AF (2010) Tracking methods for augmented reality. In: The 3rd international conference on machine vision, pp 425–429
20. Bostanci E, Clark A, Kanwal N (2012) Vision-based user tracking for outdoor augmented reality. In: 2012 IEEE symposium on computers and communications (ISCC), pp 566–568
21. Johnston D, Fluery M, Downton A, Clark A (2005) Real-time positioning for augmented reality on a custom parallel machine. Image Vis Comput 23(3):271–286
22. Chia K, Cheok A, Prince S (2002) Online 6 dof augmented reality registration from natural features. In: International symposium on mixed and augmented reality, pp 305–313
23. Jiang B, Neumann U, You S (2004) A robust hybrid tracking system for outdoor augmented reality. In: IEEE virtual reality conference
24. Bostanci E, Kanwal N, Clark AF (2012) Extracting planar features from Kinect sensor. In: Computer science and electronic engineering conference, pp 118–123
25. Hartley R, Zisserman A (2003) Multiple view geometry in computer vision. Cambridge University Press, Cambridge
26. Ma Y, Soatto S, Kosecka J, Sastry SS (2004) An invitation to 3-D vision: from images to geometric models. Springer, Berlin
27. Bay H, Tuytelaars T, Gool LV (2006) SURF: speeded up robust features. EECV, LNCS: 3951, pp 404–417
28. Corporation M (2010) Kinect for xbox360. http://www.xbox.com/en-GB/Kinect. Last access Nov 2013
29. Weinmann M, Wursthorn S, Jutzi B (2011) Semi-automatic image-based co-registration of range imaging data with different characteristics. In: Photogrammetric image analysis PIA11, LNCS: 6952, pp 119–124

30. Lepetit V, Moreno-Noguer F, Fua P (2009) EPnP: an accurate O(n) solution to the PnP problem. Int J Comput Vis 81(2):155–166
31. Alzinger W (1972) Die Ruinen von Ephesos. Koska
32. Hueber F, Erdemgil S, Buyukkolanci M (1997) Ephesos. Zaberns Bildbande zur Archaologie. von Zabern
33. Khoshelham K (2010) Accuracy analysis of kinect depth data. GeoInform Sci 38(5/W12)
34. Castro D, Kannala J, Heikkila J (2011) Accurate and practical calibration of a depth and color camera pair. In: International conference on computer analysis of images and patterns, vol 6855, pp 437–445
35. Burrus N (2012) Kinect calibration method. http://nicolas.burrus.name/index.php/Research
36. Bradski G, Kaehler A (2008) Learning OpenCV: computer vision with the OpenCV library. O'Reilly
37. Magnenat S (2013) Kinect depth estimation approach. http://openkinect.org/wiki/Imaging
38. Rosten E, Drummond T (2006) Machine learning for high-speed corner detection. Comput Vis ECCV 2006:430–443
39. Rosten E, Porter R, Drummond T (2010) Faster and better: a machine learning approach to corner detection. IEEE Trans Pattern Anal Mach Intell 32:105–119
40. Bostanci E, Kanwal N, Clark AF (2012) Feature coverage for better homography estimation: an application to image stitching. In: Proceedings of the IEEE international conference on systems, signals and image processing
41. Calonder M, Lepetit V, Strecha C, Fua P (2010) BRIEF: binary robust independent elementary features. In: European conference on computer vision, pp 778–792
42. Fischler M, Bolles R (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 24:381–395
43. Haralick RM, Lee C, Ottenberg K, Nölle M (1994) Review and analysis of solutions of the three point perspective pose estimation problem. Int J Comput Vis 13(3):331–356
44. Szeliski R (2011) Computer vision: algorithms and applications. Springer, Berlin
45. Lepetit V, Fua P (2005) Monocular model-based 3D tracking of rigid objects: a survey. Found Trends Comput Graph Vis 1(1):1–89
46. Moreno-Noguer F, Lepetit V, Fua P (2007) Accurate non-iterative O(n) solution to the PnP problem. In: IEEE international conference on computer vision
47. Kalman RE (1960) A new approach to linear filtering and prediction problems. Trans ASME J Basic Eng 82(Series D):35–45
48. Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 8(6):679–698
49. Douglas D, Peuker T (1973) Algorithms for the reduction of the number of points required to represent a digitised line or its caricature. Can Cartograph 10:112–122
50. Isard M, Blake A (1998) Condensation—conditional density propagation for visual tracking. Int J Comput Vis 29(1):5–28
51. Thrun S, Burgard W, Fox D (2006) Probabilistic robotics. MIT Press, Cambridge
52. Piekarski W, Thomas BH (2004) Augmented reality working planes: a foundation for action and construction at a distance. In: International symposium on mixed and augmented reality
53. (2014) Autodesk: AutoCAD and 3D studio max. http://usa.autodesk.com/autocad/, http://usa.autodesk.com/3ds-max/. Last access Nov 2013
54. Apollonio FI, Corsi C, Gaiani M, Baldissini S (2010) An integrated 3D geodatabase for Palladio's work. Int J Archit Comput 8(2):111–133