
Proceedings Paper:
Hadley, Dawn orcid.org/0000-0001-5452-5265, Leach, M, Maddock, S et al. (8 more authors) (2018) Recreating Sheffield's Medieval Castle in situ using Outdoor Augmented Reality. In: Virtual Reality and Augmented Reality:15th EuroVR International Conference, EuroVR 2018, London, UK, October 22–23, 2018, Proceedings. 15th EuroVR International Conference, 22-23 Oct 2018 Lecture Notes in Computer Science . Springer , GBR , pp. 213-29.

Recreating Sheffield's Medieval Castle *in situ* using Outdoor Augmented Reality

Matthew Leach(✉)[1][0000−0002−8901−5609], Steve Maddock[1][0000−0003−3179−0263], Dawn Hadley[1][0000−0001−5452−5265], Carolyn Butterworth[1], John Moreland[1], Gareth Dean[1], Ralph Mackinder[1], Kacper Pach[1], Nick Bax[2], Michaela Mckone[2], and Dan Fleetwood[2]

[1] University of Sheffield, UK
{mileach1,s.maddock,d.m.hadley,c.butterworth,
j.moreland,g.dean,r.mackinder,kpach1}@sheffield.ac.uk
[2] Human, Sheffield, UK
{nick,michaela,dan}@humanstudio.com

Abstract. Augmented Reality (AR) experiences generally function well indoors, inside buildings, where, typically, lighting conditions are stable, the scale of the environment is small and fixed, and markers can be easily placed. This is not the case for outdoor AR experiences. In this paper, we present practical solutions for an AR application that virtually restores Sheffield's medieval castle to the Castlegate area in Sheffield city centre where it once stood. A simplified 3D model of the area, together with sensor fusion, is used to support a user alignment process and subsequent orientation tracking. Rendering realism is improved by using directional lighting matching that of the sun, a virtual ground plane and depth masking based on the same model used in the alignment stage. The depth masking ensures the castle sits correctly in front of or behind real buildings, as necessary, thus addressing the occlusion problem. The *Unity* game engine is used for development and the resulting app runs in real-time on recent high-spec *Android* mobile phones.

Keywords: Augmented Reality · Outdoor Augmented Reality · Mobile Augmented Reality · Location-based Augmented Reality · Smartphones · Occlusion culling · Cultural heritage

1 Introduction

Sheffield's medieval castle is long gone, destroyed during the English Civil War in the mid-seventeenth century. However, the legacy of the castle endures in the landscape of the city: the location of the castle, Castlegate, was developed for industry and then for various markets. It now lies abandoned, after Castle Market was relocated in 2013, and awaits redevelopment. This paper presents research on using Augmented Reality (AR) to visualise a 3D model of medieval Sheffield Castle embedded in the Castlegate site.

Outdoor AR experiences which attempt to embed 3D content into an environment are more complex than AR experiences inside buildings. Potential

solutions are complicated by real world complexities such as dynamic environments (e.g. people and traffic movement and lighting changes) and solving the occlusion problem, i.e. showing a 3D model with some parts in front of and some parts behind different buildings. Specialist hardware, with depth cameras, can help, as can remote server power, but real-time SLAM (simultaneous localisation and mapping) is beyond consumer mobile phones for outdoor AR.

This paper presents a set of practical solutions to the challenges of producing an outdoor AR experience in a city centre site. The scale of the problem is constrained by using prior knowledge of the site, a user-controlled alignment process and the fusion of a range of sensor data. GPS is used to locate the user at one of a few set viewing points, which helps to optimise subsequent rendering speed for the 3D castle model. A virtual model of the 3D area is then overlaid on the mobile phone's video feed and the user aligns the model with the real world, giving a solid fix on position and orientation, before the virtual castle is displayed. The 3D model of the area is also used to address the occlusion problem. This knowledge-based depth masking process means that the castle sits in front of and behind different buildings, accordingly, based on user position. The mobile phone's sensors (GPS, gyroscope and accelerometers) are used to deal with continuous viewing changes; the compass sensor is also used as part of initial orientation setting. In addition, the sun's approximate position is used to change the lighting for the virtual castle, thus better integrating it into the real world environment. Whilst previous solutions have dealt with the occlusion problem for AR, our research work uses a virtual object (the castle) that is much bigger than its surrounding buildings, and, at the same time, deals with partial occlusion by those buildings in real-time on a consumer smartphone.

The remainder of this paper is organised as follows. Section 2 will consider related work, looking at the range of issues that affect outdoor AR experiences. Section 3 will present the system, covering the data required (models of the castle and the relevant area of the city and photographs of landmark buildings), the user processes (alignment and viewing) and rendering, including the approach for solving the occlusion problem. Section 4 will present the results and discussion. Finally, Section 5 presents conclusions.

2 Related Work

AR works best indoors, with various toolkits available to support the creation of indoor AR experiences: *ARToolKit* [3], *ARKit* and *ARKit 2*[4] [2], *ARCore*[5], *Vuforia*[6], and *Wikitude*[7]. Both marker-based and markerless tracking are supported, with ground plane detection being a key part of markerless solutions [15]. However, markerless tracking is difficult to achieve on outside scales, as the ground

[3] https://www.hitl.washington.edu/artoolkit/
[4] https://developer.apple.com/arkit/
[5] https://developers.google.com/ar/
[6] https://www.vuforia.com/
[7] https://www.wikitude.com/

is often uneven and may have obstacles in the way which frustrate the detection process. Nonetheless, there has been successful outdoor AR work. Verykokou et al [16] use a tablet PC in their computer-vision based work, but they only detect a specific almost-planar object in the scene before augmentation. Seo et al [14] use an image registration technique but further work is needed for the method to be applicable to smartphones. The ideal solution for outdoor tracking is a process known as simultaneous localisation and mapping (SLAM) [4]. This family of methods uses computer vision to build a virtual map of the surroundings, in which features are detected and tracked to position and orientate the user. The approach is commonly used in robotics, but only works well with specialist hardware such as depth sensors and also requires complex computer vision processing, which would be too slow on a consumer grade mobile device.

Practical AR applications can be produced on consumer mobile devices, albeit with compromises. Perhaps the best known example of this is *Pokémon GO*[8], which became wildly popular across the UK and in many other countries after its release in 2016 [13]. This takes advantage of a multiscale approach, where the map view only uses GPS to roughly locate the user and then markerless detection is used to place a virtual Pokémon on the ground level in front of the user. As the locations are controlled and the Pokémon only appear near to the user, it is (reasonably) certain that the ground plane will be easy to detect and that there isn't much integration required to make the Pokémon appear as part of the scene.

Cirulis and Brigmanis [3] also make use of a phone's GPS. They compute the relative position of virtual buildings and display them based on the GPS location, however, with GPS results being relatively inaccurate this could easily cause alignment issues and jittering. Huang et al [8] use a virtual model of an area with dedicated hardware to perform outdoor registration, but they compromise on precise tracking, instead focusing on information display. Vlahakis et al [17] also use GPS, and enhance this with a Differential GPS beacon located at a known position to improve accuracy. CityViewAR [11] uses GPS for geolocation of city buildings. This works outdoors but is constrained by not dealing with anything in front of the virtual building.

Marker based techniques have been used in outdoor applications [12, 10]. For example, Kim et al [10] use the *Vuforia AR plugin*[9] for *Unity*[10] to provide information about three Korean cultural heritage sites. All of the sites have good features for marker based detection, although they are focusing on information display rather than augmenting the sites themselves. As such they only need to detect whether one of the sites is visible, rather than obtain any solid tracking information.

An issue common to many AR experiences is producing correct occlusion of virtual objects. Without depth information, even if a virtual object should appear behind a real world object, it will still appear in front of it. Techniques

[8] https://www.pokemongo.com/
[9] https://unity3d.com/partners/vuforia
[10] https://unity3d.com/

for obtaining depth information either rely on stereo/depth cameras, or using prior knowledge of the scene combined with location of the user in a virtual environment. At present, depth cameras have insubstantial range for outdoor use, and produce low resolution data. This requires further processing to construct an environment mesh from point cloud data. This can be done using traditional mesh reconstruction algorithms, although more recently neural network based approaches are being experimented with and producing promising results [7]. In outdoor applications, the prior knowledge approach is more commonly used [5, 9]. We also use prior information, which is a 3D model of the environment that the virtual object is embedded into. The virtual object is a castle displayed at real scale. Occlusion with surrounding, smaller, real buildings is also addressed.

3 Data and Methods

Figure 1 shows the various components of our AR application and the data required for each stage. Stage 1 provides instructions to the user, including a map of the area and recommended viewing points. Stage 2 includes the alignment process where, after an initial coarse check on viewing position and direction using the GPS and compass sensors on the smartphone, the user aligns a virtual 3D model of the area containing various 'landmark buildings' with the real world view. Stage 3 is the viewing stage, where the castle is seen in situ using AR, correctly aligned and positioned relative to the user. Thereafter, tracking of orientation is done using the smartphone's gyroscope and accelerometer sensors. The app is developed using the *Unity* game engine and built for *Android*. The phone used for development and testing was a Motorola Moto Z (Snapdragon 820 processor, 1.8 GHz Quad-core CPU, Adreno 530 integrated GPU). The following subsections will describe the components of the system.

3.1 Data

The data required for the application consists of a model of the Castlegate area, photographs of the front of specific landmark buildings, and a model of the castle. The 3D model of the Castlegate area was produced by MArch students at The University of Sheffield's School of Architecture and is illustrated in Figure 2 using *SketchUp*[11]. The model is made up of approx. 55,000 triangles. This relatively small memory footprint lessens the burden on the smartphone's processor and is sufficient to support the user alignment process and the depth masking aspects of the AR application. The initial model is untextured – it is the geometry that is important for the depth masking stage. However, to support the user alignment process (see later), photographs of the fronts of 'landmark buildings' (buildings that are easily distinguishable within the Castlegate area) are added to relevant parts of the model as texture maps, as illustrated in Figures 3 and 4.

[11] https://www.sketchup.com/

Program Flow **(User Processes)**	**Data/Hardware Requirements**
App Start (Press Continue)	– User Interface Icons
Alignment Screen (User lines up virtual landmark buildings with real world scene) (Press Aligned)	– GPS location – Landmark building photos – Known locations for landmark buildings – Gyroscope data – Accelerometer data
Landmark buildings hidden, castle displayed (UI options)	– Castle model – Gyroscope data – Accelerometer data

Fig. 1. An overview of the components of the system.

The model of Sheffield Castle (Figure 5) was created by Human[12], a Sheffield-based creative agency. It is based on archaeological and historical evidence for what the castle was like, drawn from research on the unpublished archives from mid-twentieth-century excavations, with inspiration also drawn from surviving castles of similar type (Richmond, Helmsley and Barnard), for the architectural details. The castle is modelled as a set of distinct pieces as shown in Figure 6 so as to support only rendering those that are visible during rendering. Each of the pieces is hidden or shown depending on the viewing location. The complete model

[12] http://humanstudio.com/

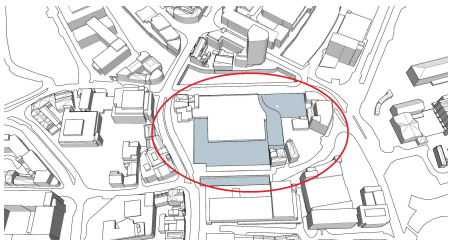

Fig. 2. The model of the Castlegate area viewed in *Sketchup*. Castlegate, which is where the castle was situated, is oulined in red.

Fig. 3. Texture map of a landmark building in the scene.

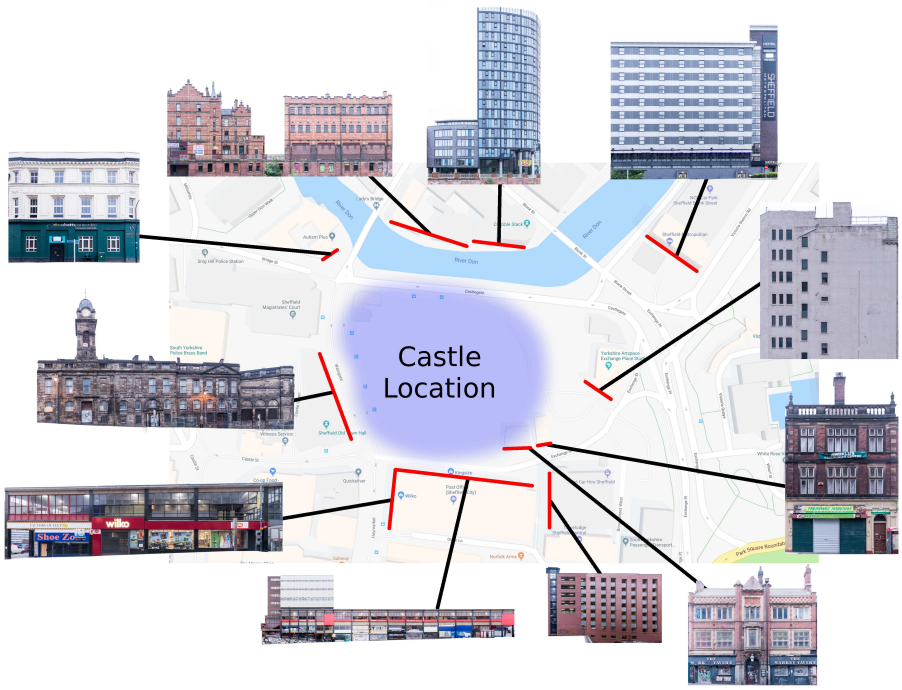

Fig. 4. The various landmark buildings and their locations. Map data ©2018 Google.

consists of 3100 triangles and uses 69 textures, 50 with resolution 2048x2048, 16 with resolution 1024x1024 and 3 with resolution 512x512. Figure 7 shows how the castle sits in the Castlegate model from Figure 2. A key part of this stage is to make sure the ground level of the two models is aligned – this is important for later stages. The ground heights of each of the area and castle models follows the current land height for the Castlegate area, although the castle model includes a moat.

3.2 Alignment Processes

This section focusses on stage 2 in Figure 1. An initial viewpoint is established using GPS, followed by a user alignment process between the model and the real world view. Orientation tracking is also required.

GPS for standard smartphones is only accurate to approx. 5-8.5m in good conditions [19]. In an urban environment, particularly when the scene is being viewed from pavement level, tall buildings may be close to the user and lead to even worse performance. We solve this issue by defining specific viewpoints where the user should stand. The active viewpoint is chosen by selecting the viewpoint with the minimum Euclidian distance to the reported GPS location.

Fig. 5. The textured, full resolution model of the castle. The model includes a surrounding landscape and moat (in grey in this image).

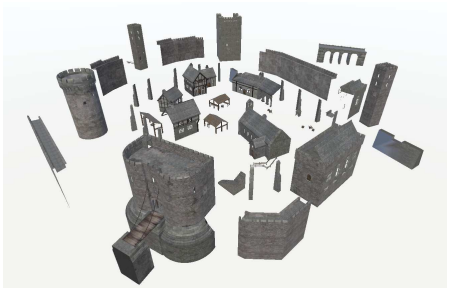

Fig. 6. An exploded view of the separate parts of the castle.

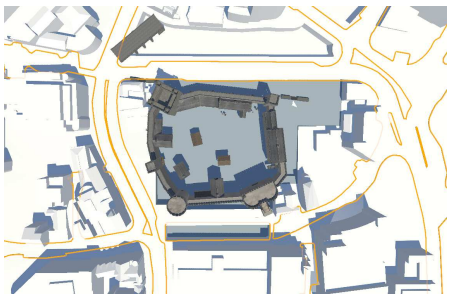

Fig. 7. The model of Sheffield Castle positioned at its historic location in the Castlegate model.

This selection process takes place when the app is started, and every 10 seconds thereafter – a continuous update would use unnecessary resources, both computationally and in terms of battery power on the user's device. A time of 10 seconds is considerably shorter than it would take to walk between any of the defined viewpoints, ensuring that when the user reaches such a point, the app will have updated their location.

Having established a viewpoint, the next step is a user alignment process. This involves detecting view orientation and the direction the user is facing. In theory, the compass could be used to detect the direction the user is facing. However, mobile device compasses are not particularly accurate, and are also affected by surrounding magnetic fields. During testing, the reported angle was often found to be up to twenty or thirty degrees away from the true angle. Thus an accurate bearing could not be found. However, it gave an initial guess for the orientation, which roughly aligns the viewpoint direction with the Castlegate model so that relevant landmark buildings are in view. Using this initial information, the landmark buildings can then be used to refine the alignment.

Figures 3 illustrates one of the landmark buildings, as described in Section 3.1. From the user's position, the landmark buildings are displayed in their correct position relative to the virtual Castlegate model – the white Castlegate area model is not visible, only the landmark buildings. The user can swipe on the screen to rotate the scene until the relevant virtual landmark buildings line up with the real ones, based on the current viewpoint position. The Castlegate model is now aligned with the real world from the user's viewpoint. However, other processes are happening in parallel – user orientation and perspective correction – which must be considered before the alignment process is complete.

The user's orientation is tracked so that the correct view of the castle model can be presented in relation to the real world view. The smartphone's gyroscope and accelerometer sensors are used for this. The gyroscope sensors give a very accurate reading for the angular velocity of the device around each of the main axes. By integrating this we can determine the total angle the device has moved through. An issue, however, is that error accumulation in the integration causes drift. Initial experiments used a Kalman filter [18] to mitigate this. This worked well for correcting pitch measurements, but absolute heading values from the compass were incorrect and, since the Kalman filter used these to update its state, the results were poor, converging to the wrong angle and producing jittery behaviour. Instead, a complementary filter [6] was used with a small timestep in the Euler integrator. Since only minor drift corrections were required, this worked well. In addition, the complementary filter has a lower performance impact than a Kalman filter requiring only a simple multiplication and addition, rather than an iterative matrix solve or approximation. This was two orders of magnitude faster in testing.

Camera perspective must also be considered. From the defined viewpoints process, the user's location is known. The roll, pitch and yaw of the smartphone are tracked by the user orientation process. The roll and pitch can be determined purely from the accelerometer and gyroscope data, whilst the user has completed the alignment process to ensure the correct yaw. These transformation values are applied to the *Unity* camera. With the real and virtual cameras' positions matching, the fields of view must be matched to ensure the same view is seen by both cameras. To match the field of view, the device camera's field of view and aspect ratio are queried. This combined with the aspect ratio of the screen is sufficient information to produce the correct perspective matrix. As the screen and camera aspect ratios do not match, the actual camera image is cropped which affects the field of view. For a screen of wider aspect ratio than camera, the updated field of view can be computed according to the following formula:

$$\theta_c \frac{(W_c/W_s)}{(H_c/H_s)} \tag{1}$$

where θ_c is the reported vertical camera field of view, W_c is the width of the image returned from the camera, W_s is the width of the screen, H_c is the height of the image returned from the camera and H_s is the height of the screen. For our test smartphone, the screen is 16:9, whilst the camera is 4:3, or 16:12. As

such the vertical component is scaled by three quarters to match the 16:9 screen aspect. This also reduces the effective vertical field of view by 3/4. The device camera reports a 50 degree vertical field of view, so three quarters of this, 37.5, is used for the vertical field of view of the *Unity* camera.

Figure 8 shows the alignment process in progress. For the building on the right in Figure 8a, both the virtual landmark building and its real counterpart are visible. The user then swipes to rotate the virtual scene until it matches the real scene as seen in Figure 8b. The user is free to look around during this process to also compare other buildings for alignment. When the user is happy with the alignment, a tap on the smartphone screen reveals the virtual castle model correctly augmented into the real world scene. One final part of the alignment process worth noting is the slowest part of the whole process was using *Unity*'s WebCamTexture class to handle the video feed. Performance was improved significantly by using code to natively access the camera.

(a) pre-alignment (b) post-alignment

Fig. 8. User alignment with the frontages of the Market Tavern and the building to its right.

3.3 Rendering

After user alignment, the virtual castle model is displayed. This involves three aspects. First, to give a real sense that the castle is augmenting the real world, real buildings that are behind the castle should not be seen, and real buildings in front of the castle should obscure the virtual model. Second, the ground planes of the virtual model and the real world should be aligned. Third, the lighting of the virtual model should consider the position of the sun in the real world, so as to better match the lighting of the surrounding real world buildings.

Correctly embedding the virtual 3D model into the real world required a process to detect what should be in front of, rather than behind, the virtual object. Standard smartphone cameras do not report any depth information. Our solution for the occlusion problem makes use of the user's location and orientation and knowing what the user's view should be in the real world, based on the earlier user alignment process between the virtual Castlegate model's

landmark buildings and the real world. Since we have the full virtual Castlegate model aligned with the real world, and we know the castle's position within the Castlegate model, we can use the Castlegate model to create a depth mask to stop portions of the virtual castle from being drawn, making it appear hidden by buildings in the foreground. We call this location knowledge-based depth masking. It is similar in concept to the approach used in [5, 9], but is extended to use 3D models. They are only interested in occluding small, ground height markers, however, for an object the size of a castle, only portions of it may be occluded, and it is larger than the occluding objects, so their ray based approach is not sufficient. Our method allows occlusion of only parts of objects.

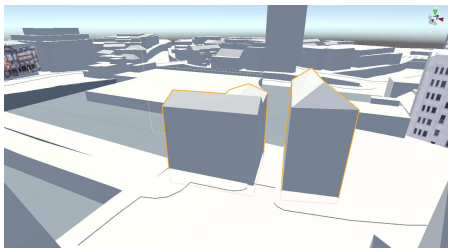

Fig. 9. The Castlegate model buildings outlined in orange will act as a mask for the castle.

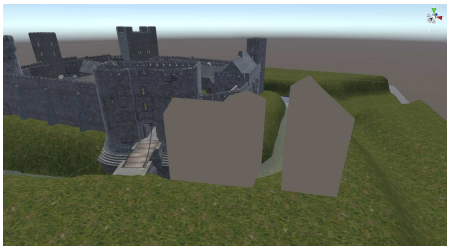

Fig. 10. The castle masked by the Castlegate model buildings.

A multi-pass rendering approach is used. Initially, the depth and colour buffers are cleared. In the first pass, the video feed is rendered full screen. This ensures that a full background is available. In the second pass, occluding buildings are rendered using a shader which writes only to the depth buffer. This masks out regions where the castle should not be drawn because buildings are present in front of it, as illustrated in Figure 9, and the actual buildings will be displayed in the correct location in the video feed assuming the alignment process was carried out correctly. In the final pass, the castle is rendered, with any masked portions failing the depth test, essentially cutting a hole in the castle model, as illustrated in Figure 10. Thus the castle will appear to be behind real foreground buildings.

Making AR objects appear as though they are correctly integrated with the ground is challenging. For small objects, a simple shadow surrounding it may be sufficient, but for a large object this is very difficult as correctly computing how the shadow should appear on the video feed is non-trivial. In addition, without proper depth information, even portions of the castle model that might be under the ground are rendered on top. To solve this, we use a virtual ground plane. The ground is modelled to match the layout of the real land. This approach means inclusion of the moat is trivial.

To further integrate the castle into the real world, the sun's position must be considered so that the lighting of the castle appears to better match that of the

surrounding buildings. The sun's position can be computed from the date, time of day and longitude and latitude of the area. This calculation involves using astronomical formulae, based on those found in the *Astronomical Almanac*[13]. Initially, the date is converted to Julian days and centuries. From these, sidereal time is computed. Solar coordinates are determined from the results of the previous calculations, and these are used to calculate the right ascension and declination. Finally, these are transformed to Alt/Az coordinates. A more detailed explanation can be found in the Appendix.

The final aspect of rendering to consider is performance, since a mobile device has limited processing power. The castle is composed of individual pieces (Figure 6) to help increase performance. Only those parts that are visible from the defined viewpoints in the application need to be rendered; those pieces that are entirely blocked by others do not need to be rendered. In addition, mipmapping and level of detail techniques are used to further reduce the rendering time.

4 Results and Discussion

Figures 11 and 12 show the view when the user is aligning the Old Town Hall landmark building texture with its real world counterpart. Once this is aligned, the virtual castle model is then displayed, as shown in Figure 13. Note how the real old town hall building is seen beyond the virtual castle. The user can then rotate her mobile device to show other parts of the scene. Figure 14 shows the scene once the viewpoint is rotated to the right along the castle wall to show the main gate. The real Market Tavern (also one of the landmark buildings in the Castlegate model) is shown in front of the castle demonstrating the effect of the depth masking process. Another example of the alignment process is given in Figures 15 and 16. Here, the Metropolitan hotel is used as the landmark building in the alignment process, and the user must then rotate her camera to see the castle (Figure 17).

Over time, with continuous changes in orientation, some calculation drift can occur, since this is based on integration of gyroscope data. The virtual model and the real world can become slightly misaligned. In general, a misalignment of a few degrees is not an issue at this scale as the castle sits well within the area of land – a small difference in location won't be noticed. The drift does, however, create some problems in conjunction with the depth masking. The cutout in the castle model for a foreground building requires a good alignment, or the wrong portion of the video feed can be displayed in the hole (Figure 18). When this occurs, user alignment must be redone.

Figure 19 shows an enlarged portion of Figure 12. As can be seen, representing landmark buildings as textured planes works well for the flat frontage of the building. However, the tower of the old town hall is set back from the building face and, as such, appears in the wrong place for alignment, since it has been projected forwards into the textured plane. An alternative solution could use

[13] http://astro.ukho.gov.uk/nao/publicat/asa.html

Fig. 11. The view of the Old Town Hall during the alignment process.

Fig. 12. The view of the Old Town Hall after alignment. The row of shops on the left is also a landmark texture.

Fig. 13. The virtual castle viewed with the real Old Town Hall in the background.

Fig. 14. The depth masking technique cuts a hole in the castle, leaving the image of the real Market Tavern showing from the video feed.

Fig. 15. The Metropolitan hotel before alignment.

Fig. 16. The Metropolitan hotel after alignment.

Fig. 17. After user alignment with the Metropolitan hotel, the camera view is rotated to show the castle.

multiple textured planes, but it is unclear if this extra complication would be of benefit, since the current alignment process, based on the building frontage, seems to work well.

Fig. 18. Over time, with continuous changes in orientation, calculation drift can occur, producing a slight misalignment between the virtual model and the real world, which affects the depth masking process.

Fig. 19. An enlarged view of the Old Town Hall in Figure 12

5 Conclusions

A practical, working, outdoor AR system that runs on android phones with appropriate sensors has been presented. A user alignment process, together with the fusion of a range of sensors, produces a system that is stable and easy to use. The 3D model of the area is used both in the user alignment process and also as part of a depth masking process so that the 3D virtual castle is properly placed in the real world view. There are some drift issues over time, although these can be rectified by user re-alignment. Further work could consider how to retain the alignment for longer, perhaps using a lightweight version of SLAM, as well as how to remove the initial user alignment step.

Initial experiments have been done to add links to social media tools within the application, with the aim of allowing the general public to share their thoughts on the restored castle model (thus producing a reconstruction AR application, using Bekele's categorisation [1]). Also, since the Castlegate area will undergo redevelopment in the future, our intention is to be able to display the future 3D plans for the area as an alternative user option. We could also offer an option to display a model of the remaining underground chambers on the site which preserve some of the archaeological heritage. Both of these would be relatively straightforward since the models would be geolocated in the Castlegate model in the same way that the castle model was. This would give local people a chance to use an AR application to be involved in redevelopment of the site, and make their views known on both the future building plans and the site's cultural heritage.

Appendix

All trigonmetric functions listed operate in radians. Angles should be corrected to a range between 0 and 2π throughout unless otherwise noted.

Compute the number of Julian days and Julian centuries since J2000:

$$d_j = 367y - \left\lfloor \frac{7}{4}(y + \lfloor (m+9)/12) \rfloor \right\rfloor + \left\lfloor \frac{275m}{9} \right\rfloor + d - 730531.5$$

$$C_j = \frac{d_j}{36525}$$

where d_j is the Julian day, y is the year, m is the month in numerical form, d is the day in numerical form and C_j is the Julian century.

Compute the sidereal time:

$$S_h = 6.6974 + 2400.0513 C_j$$

$$S_{ut} = S_h + \frac{366.2422}{365.2422} h$$

$$S = 15 S_{ut} + L_o$$

where S_h is the sidereal time in hours at midnight, S_{ut} is the sidereal time in hours including the current time, S is the local sidereal time and L_o is the longitude.

Update to fractional (including time of day) Julian days and centuries:

$$d_j = d_j + \frac{h}{24} c_j = \frac{d_j}{36525}$$

Compute solar coordinates:

$$G_{MeanLong} = \frac{2\pi}{360}(280.466 + 36000.77 C_j)$$

$$G_{MeanAnom} = \frac{2\pi}{360}(357.529 + 35999.05 C_j))$$

$$E_{cen} = \frac{2\pi}{360}((1.915 - 0.005 C_j)\sin(G_{MeanAnom}) + 0.02 \sin(2 * G_{MeanAnom}))$$

$$L_{eliptic} = G_{MeanLong} + E_{cen}$$

$$O = \frac{2\pi}{360}(23.439 - 0.013 C_j)$$

where $G_{MeanLong}$ is the mean solar longitude, $G_{MeanAnom}$ is the mean solar anomaly, $E + cen$ is the equation of center, $L_{eliptic}$ is the eliptical longitude and O is the obliquity.

Compute right ascension and declination:

$$R = \text{atan2}(\cos(O)\sin(L_{eliptic}), \cos(L_{eliptic}))$$

$$D = \arcsin(\sin(R)\sin(O))$$

where R is the right ascension and D is the declination.

Compute horizontal coordinates. The hour angle, H, should be brought into the range $-\pi$ to π.

$$H = \frac{2\pi}{360}S - R$$
$$Alt. = \arcsin(\sin(\frac{2\pi}{360}L_a)\sin(D) + \cos(\frac{2\pi}{360}L_a)\cos(D)\cos(H))$$

where H is the hour angle and $Alt.$ is the altitude.

Compute the azimuth angle:

$$Az. = \arctan\left(\frac{-\sin(H)}{\tan(D)\cos(\frac{2\pi}{360}L_a) - \sin(\frac{2\pi}{360}L_a)\cos(H)}\right)$$

where $Az.$ is the azimuth angle.

Finally, adjust the azimuth angle to the correct quadrant.

Acknowledgements

This research was funded by a grant from the AHRC/EPSRC Immersive Experience scheme (grant no. AH/R009392/1). Research on the archaeological archives relating to Sheffield Castle was funded by the Pamela Staunton Bequest (University of Sheffield).

References

1. Bekele, M.K., Pierdicca, R., Frontoni, E., Malinverni, E.S., Gain, J.: A survey of augmented, virtual, and mixed reality for cultural heritage. J. Comput. Cult. Herit. **11**(2), 7:1–7:36 (Mar 2018). https://doi.org/10.1145/3145534, http://doi.acm.org/10.1145/3145534
2. Buerli, M., Misslinger, S.: Introducing ARKit-augmented reality for iOS. In: Apple Worldwide Developers Conference (WWDC17). pp. 1–187 (2017)
3. Cirulis, A., Brigmanis, K.B.: 3D outdoor augmented reality for architecture and urban planning. Procedia Computer Science **25**, 71–79 (2013). https://doi.org/10.1016/j.procs.2013.11.009
4. Durrant-Whyte, H., Bailey, T.: Simultaneous localization and mapping: part I. IEEE robotics & automation magazine **13**(2), 99–110 (2006). https://doi.org/10.1109/MRA.2006.1638022
5. Galatis, P., Gavalas, D., Kasapakis, V., Pantziou, G., Zaroliagis, C.: Mobile augmented reality guides in cultural heritage. In: Proceedings of the 8th EAI International Conference on Mobile Computing, Applications and Services. pp. 11–19 (2016). https://doi.org/10.4108/eai.30-11-2016.2266954
6. Higgins, W.T.: A comparison of complementary and Kalman filtering. IEEE Transactions on Aerospace and Electronic Systems (3), 321–325 (1975). https://doi.org/10.1109/TAES.1975.308081

7. Höft, N., Schulz, H., Behnke, S.: Fast semantic segmentation of RGB-D scenes with GPU-accelerated deep neural networks. In: Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz). pp. 80–85. Springer (2014). https://doi.org/10.1007/978-3-319-11206-0_9
8. Huang, W., Sun, M., Li, S.: A 3D GIS-based interactive registration mechanism for outdoor augmented reality system. Expert Systems with Applications **55**, 48–58 (2016). https://doi.org/10.1016/j.eswa.2016.01.037
9. Kasapakis, V., Gavalas, D.: Occlusion handling in outdoors augmented reality games. Multimedia Tools and Applications **76**(7), 9829–9854 (2017). https://doi.org/10.1007/s11042-016-3581-1
10. Kim, H., Matuszka, T., Kim, J.I., Kim, J., Woo, W.: An ontology-based augmented reality application exploring contextual data of cultural heritage sites. In: 2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS). pp. 468–475. IEEE (2016). https://doi.org/10.1109/SITIS.2016.79
11. Lee, G.A., Dünser, A., Kim, S., Billinghurst, M.: CityViewAR: A mobile outdoor AR application for city visualization. In: 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-AMH). pp. 57–64. IEEE (2012). https://doi.org/10.1109/ISMAR-AMH.2012.6483989
12. Murru, G., Fratarcangeli, M., Empler, T.: Practical augmented visualization on handheld devices for cultural heritage. In: Proc. 21st International Conference on Computer Graphics, Visualization and Computer Vision. pp. 97–103 (2013)
13. Paavilainen, J., Korhonen, H., Alha, K., Stenros, J., Koskinen, E., Mayra, F.: The Pokémon GO experience: A location-based augmented reality mobile game goes mainstream. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. pp. 2493–2498. ACM (2017). https://doi.org/10.1145/3025453.3025871
14. Seo, B.K., Kim, K., Park, J., Park, J.I.: A tracking framework for augmented reality tours on cultural heritage sites. In: Proceedings of the 9th ACM SIGGRAPH Conference on Virtual-Reality Continuum and its Applications in Industry. pp. 169–174. ACM (2010). https://doi.org/10.1145/1900179.1900215
15. Simon, G., Fitzgibbon, A.W., Zisserman, A.: Markerless tracking using planar structures in the scene. In: Proc. IEEE and ACM International Symposium on Augmented Reality, 2000 (ISAR 2000). pp. 120–128. IEEE (2000). https://doi.org/10.1109/ISAR.2000.880935
16. Verykokou, S., Ioannidis, C., Kontogianni, G.: 3D visualization via augmented reality: The case of the Middle Stoa in the Ancient Agora of Athens. In: Ioannides, M., Magnenat-Thalmann, N., Fink, E., Žarnić, R., Yen, A.Y., Quak, E. (eds.) Digital Heritage. Progress in Cultural Heritage: Documentation, Preservation, and Protection. pp. 279–289. Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-13695-0_27
17. Vlahakis, V., Ioannidis, M., Karigiannis, J., Tsotros, M., Gounaris, M., Stricker, D., Gleue, T., Daehne, P., Almeida, L.: Archeoguide: an augmented reality guide for archaeological sites. IEEE Computer Graphics and Applications **22**(5), 52–60 (2002). https://doi.org/10.1109/MCG.2002.1028726
18. Welch, G., Bishop, G.: An introduction to the Kalman filter: SIGGRAPH 2001 course 8. In: Computer Graphics. Annual Conference on Computer Graphics & Interactive Techniques, Los Angeles, CA, USA (August 12-17) (2001), https://sreal.ucf.edu/wp-content/uploads/2017/02/Welch2001.pdf
19. Zandbergen, P.A., Barbeau, S.J.: Positional accuracy of assisted GPS data from high-sensitivity GPS-enabled mobile phones. The Journal of Navigation **64**(3), 381–399 (2011). https://doi.org/10.1017/S0373463311000051