Augmented Reality in the Control Tower: a Rendering Pipeline for Multiple Head-Tracked Head-Up Displays

Nicola Masotti[1], Francesca De Crescenzio[1], Sara Bagassi[1]

Department of Industrial Engineering
University of Bologna
Bologna, Italy

{nicola.masotti, francesca.decrescenzio,
sara.bagassi}@unibo.it

Abstract. The purpose of the air traffic management system is to accomplish the safe and efficient flow of air traffic. However, the primary goals of safety and efficiency are to some extent conflicting. In fact, to deliver a greater level of safety, separation between aircrafts would have to be greater than it currently is, but this would negatively impact the efficiency. In an attempt to avoid the trade-off between these goals, the long-range vision for the Single European Sky includes objectives for operating as safely and efficiently in Visual Meteorological Conditions as in Instrument Meteorological Conditions. In this respect, a wide set of virtual/augmented reality tools has been developed and effectively used in both civil and military aviation for piloting and training purposes (e.g., Head-Up Displays, Enhanced Vision Systems, Synthetic Vision Systems, Combined Vision Systems, etc.). These concepts could be transferred to air traffic control with a relatively low effort and substantial benefits for controllers' situation awareness. Therefore, this study focuses on the see-through, head-tracked, head-up display that may help controllers dealing with zero/low visibility conditions and increased traffic density at the airport. However, there are several open issues associated with the use of this technology. One is the difficulty of obtaining a constant overlap between the scene-linked symbols and the background view based on the user's viewpoint, which is known as 'registration'. Another one is the presence of multiple, arbitrary oriented Head-Up Displays (HUDs) in the control tower, which further complicates the generation of the Augmented Reality (AR) content. In this paper, we propose a modified rendering pipeline for a HUD system that can be made out of several, arbitrary oriented, head-tracked, AR displays. Our algorithm is capable of generating a constant and coherent overplay between the AR layer and the outside view from the control tower. However a 3D model of the airport and the airport's surroundings is needed, which must be populated with all the necessary AR overlays (both static and dynamic). We plan to use this concept as a basis for further research in the field of see-through HUDs for the control tower.

Keywords: Air·Traffic·Control·Tower·Augmented·Reality·Head-Up·Display.

The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-40621-3_23

1 Motivation

With the aim of increasing the air transport system efficiency and throughput, Europe has made plans for operating in Instrument Meteorological Conditions (IMC) as safely and efficiently as in Visual Meteorological Conditions (VMC). [1–3]. From a pilot perspective, the research on all-weather operations cockpits is already far advanced. Indeed, the integration of Head-Up Displays (HUDs) into modern civil flight decks has demonstrated many advantages. In modern cockpits, HUDs can be supplemented by Enhanced Vision Systems (EVS) and Synthetic Vision Systems (SVS) and a combination of these, the so-called Combined Vision Systems (CVS), is already being studied in the Single European Sky ATM Research (SESAR) for landing, take-off and taxi [4]. On the ground, a task that is still largely dependent on the visual observation of the surrounding area is the provision of Air Traffic Control (ATC) service by the control tower. Indeed, results of controllers' task analyses have shown the importance of the outside view for enhancing controllers' Situation Awareness (SA) [5–8]. Depending upon weather and lighting conditions, the visual contrast of controlled objects varies substantially, with possible detrimental impact on controllers' performances [9]. In particular, when bad weather, fog, smoke, dust or any other kind of environmental occlusion impairs the visibility from the control tower, the airport capacity is reduced and Low Visibility Procedures (LVP) must be applied. In addition, it is also possible for the airport, the surrounding airspace, and the controlled vehicles to be obscured by buildings, high-glare conditions and the cover of night [9]. LVP may include constraints, such as mandatory use of a Surface Movement Radar (SMR), taxiways that cannot be used, block spacing, limitation in pushback operations and use of a predefined runway. Consequently, as long as the operational capability is reduced, both carriers and Air Navigation Service Providers (ANSPs) incur in heavy financial losses. In [10], Shackelf and Karpe refer to large fuel savings and financial benefits if stable rates of airport capacity could be maintained in all visibility conditions. This also implies a higher arrival and departure rates and a more uniform and productive Air Traffic Flow Management (ATFM). Further, the increased reliability of the surface management service would improve metrics for taxi-times, departure queues, ground-delays, ground-holds and cancellations [10].

In recent years, many advances in Air Traffic Management (ATM) have come in the form of visualization tools for tower controllers. Movement maps, conformance monitoring, conflict detection and others Advanced Surface Movement Guidance & Control System (A-SMGCS) based solutions are a few examples of these tools. But there is a paradox in developing visual tools in order to increase the tower controllers' situation awareness (SA), which is that their sight is pulled away from the outside view and the head-down time is increased. Previous studies have already proven that tasks requiring frequent shifts of gaze back and forth between the outside and the inside view may become significantly slow and fatiguing, particularly after the fortieth years of age [11].

In other words, a constant refocusing between the far view and the head-down equipment contributes to the operator's workload and reduces his or her SA. The use of augmented reality tools (AR) that can safely enable tower operations in zero/low visibility conditions may be able to address this paradox.

2 Augmented Reality for the Airport Tower

The topic of Augmented Reality (AR) appears in the human factors' literature with increasing frequency, usually in conjunction with the more familiar subject of Virtual Reality (VR) [12]. Lloyd Hitchcock of the FAA firstly proposed the concept of using AR technology in the control tower over 25 years ago [13]. At that time, no prototype construction was attempted and little was published, though many recall Mr. Hitchcock speculating on several methods that could aid tower controllers [5]. For instance, he suggested that AR displays could provide air traffic controllers with useful status information, such as aircraft identification, barometer settings, wind conditions and runway/gate assignments. More recent studies suggest that also other spatially conformal information, such as flight tags, warnings, shapes and layouts, can be presented on AR displays [12, 14–21]. Displayed information may be extracted and synthesized from multiple data sources, such as radar-based surveillance systems (e.g. Airport Surveillance Radar and Surface Movement Radar), Differential Global Positioning System (DGPS), 3D digital maps and other ground based sensors (e.g. video or infrared cameras). Other information that can be displayed to the controller includes System Wide Information Management (SWIM) data, such as weather conditions, wind direction and speed, wind shear and wake vortexes visualization [22, 23]. These could be used to optimize separations between approaching and departing aircrafts, leveraging weather in a similar manner to what controllers did in SESAR Operational Service and Environment Definition (OSED) 06.08.01 – Time Based Separation [24]. In any case, a 3D airport model must be developed providing precise positioning for infrastructures and objects (both aerial and terrestrial). Similar technology was developed in SESAR Operational Focus Area 06.03.01 (Remote Tower) particularly in SESAR Project 06.09.03 (Remote & Virtual TWR) where visual overlays have been used to introduce or highlight relevant information on the out of the window view [25]. However, in this case, the AR layer has been placed on top of the video surveillance feed of a remote airport location instead of the actual tower's windows.

2.1 Expected Impacts

Using AR in the airport tower means that controllers will be no longer limited by what the human eye can see out of the tower's windows. Consequently, constraints in LVC could be reduced. For instance, when relying on visual augmentations, an exclusive use of taxiway blocks may not be necessary. Therefore, an aircraft could use a segment of

a taxiway before the preceding aircrafts has left such segment. In other words, those tasks that can be negatively affected by poor visibility conditions will become weather-independent and the risk of creating bottlenecks in the traffic flow management system will be reduced.

AR overlays can also aid users by substantially reducing the amount of visual scanning needed to integrate various sources of information. This contrasts with the current practice of scanning multiple devices (screens, windows, flight strips, etc.), filtering the essential information from data that may not be relevant. As a result, the head-down time should be reduced.

On the whole, significant benefits are expected for the entire air traffic system, including (a) increased safety for passengers, (b) financial savings for carriers and ANSPs, (c) environmental pollution reduction, and (d) increased efficacy (and resilience) of the control tower IT system (Fig. 1) [5, 9, 10, 12, 19]. Also, the maintenance of operational capacity in all weather conditions should result in positive social impact on tourists, business travellers and the community living in the airport surrounding.

Finally, the development of AR tools will provide a technology bridge between the current tower systems and the 21st Century 'Remote & Virtual Tower' (R&VT) concept foreseen in both SESAR and NGATS (Next Generation Air Transportation System) visions. Over the last few years, several concepts for the provision of air traffic service from a distant/remote location have been proposed, including video-surveillance based systems (remote towers), and VR facilities in which a photo-realistic real-time rendering recreates a 360-degree tower view (virtual towers) [8, 9, 26, 27]. The first concept is actually far advanced in SESAR and has been proven ready for industrialization (leading to operational deployment). As for the second, this may take decades to refine. Nevertheless, there are strong financial reasons to develop this technology. One of the open issues associated with virtual towers is the assessment of the extent to which the 'digital world' can be trusted to resemble the referenced real world. In this sense, AR may become of critical importance for the R&VT research. If an augmented reality tool became certified and operational in the next several decades, it is expected that the community of tower controllers would generate discrepancy reports each time there is a mismatch between the real world that they observe and the virtual world that is presented via the AR Tower Tool [5]. Conversely, the inability of controllers to detect such discrepancies would become valuable data for the validation, verification and certification of R&VTs [5]. In this sense, AR towers will provide a suitable development path for designing the fully immersive virtual tower of the future [9].

2.2 Technologies

Many types of AR displays exist, all having specific areas of application. In [28] Bimber and Raskar provide a classification based on the AR display position along the optical path between the observed object and the viewer's eyes. Their classification includes head-attached displays (a.k.a. head-mounted displays), hand-held displays, spatial displays (a.k.a. head-tracked displays or head-up displays) and object-projected AR displays (head-attached, hand-held and spatial). Apart from the latter, that is so called because the virtual image is projected directly onto a real object, AR displays can be either of the see-through or the video-combined category. See-through AR displays combine the real and virtual images by means of mirrors, lenses, transparent screens or other optical components. This leaves the view of the real world nearly intact. Video-combined AR displays use cameras to covert the real view into a video feed, which is later merged with the virtual image. Spatial AR displays are typically fixed in space (e.g. attached to a desk, fixed on the floor or hung from the ceiling) and can be made to coincide with the tower windows (Fig. 2). These are often at an angle with each other and slope toward the tower at the base to avoid internal reflections. See-Through Head-Mounted Displays (ST-HMD) are worn by the user, resulting in a flexible but intrusive equipment. For both these systems to function properly, a head-coupled or eye-coupled perspective is needed. Thus, at any time, the underlying application (i.e. the one that generates the AR overlay) must know where the controller is and where s/he is looking. Indeed, the content of the screen is determined by the point of view of the observer, usually by tracking his or her head position and orientation. This differs from an aircraft HUD, where the displayed information is adjusted to the aircraft's perspective rather than the one of the pilot [29].

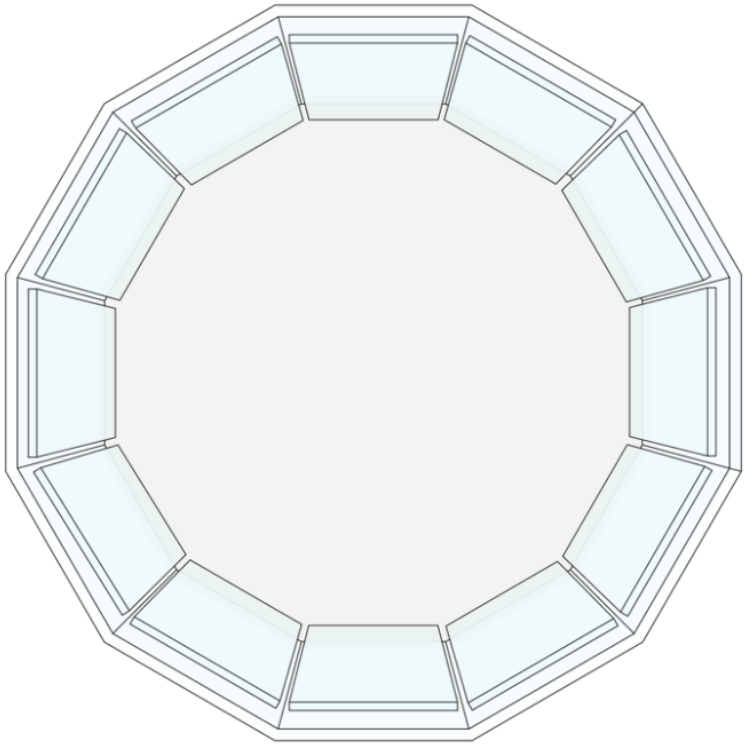

Fig. 1. A possible 360° deployment of spatial AR displays in the airport tower.

Also, because of its potentially large form factor, a spatial AR display can provide a considerably larger Field Of View (FOV) compared to a typical aircraft HUD.

2.3 Application area

In this work, the application area is defined as panoramic, i.e. an egocentric environment where the observer is confined in a limited volume but experiences a panoramic view of the environment surrounding that volume [30]. Examples of panoramic environments include ATC towers and other control or supervision positions where complex visual tasks are performed from a distance (e.g. life guarding). Also, this work will focus on spatial see-through AR displays, primarily because they are less intrusive than head-attached (i.e. head-mounted) or hand-held displays. Object-projected AR will not be considered in this work because it doesn't fit the application area we have defined (real objects are far away, and potentially obstructed by other objects, therefore it is not possible to project images on them).

2.4 Open issues

In the field of AR the concept of spatially matching the real and the virtual objects according to the user perspective is known as registration [30]. Alternate designations include 'object alignment', 'object connectivity', and 'conformal' or 'scene-linked' symbology [20, 21, 30]. As already mentioned, in order to achieve registration, one crucial factor is to have accurate spatial data (tracking) of the observed object, display device and observer (at all instances). This may be accomplished by means of depth from stereo, infrared tracking or many others techniques. Inaccurate measurements or latency in the tracking methodology lead to registration errors, which can seriously affects the system usability [30]. However, tracking is a widely researched topic [31, 32] and will not be discussed further in this paper. Eventually, the tracking process must result in the head/eyes coordinates being fed (in real time) to the rendering pipeline.
Even assuming that an accurate tracking result is continuously fed to the AR content generator, there are still a number of significant issues in getting the real and virtual imagery to blend naturally. For instance, a digital model of the airport and the airport surroundings must be developed and populated with all the necessary overlays for both aerial and terrestrial object. Also, because in the airport tower multiple HUDs may be arbitrary oriented, any attempt to use a standard projection model would fail [33].
What we propose is a modified rendering pipeline that can be used in a first person, head-tracked (or eye-tracked), multi-screen, non-planar, panoramic environment, such as the AR control tower of the future. If all others prerequisites are met (i.e. accurate tracking and modelling) our algorithm will generate a spatially registered (i.e. conformal) overlay for an arbitrary number of anyway oriented HUDs. In other words, this is a flexible mechanism for generating AR contents that are (a) consistent with the display orientation and (b) constantly overlaid with the real view (i.e. spatially registered).

3 The standard projection model

The majority of Virtual/Augmented Reality (V/AR) applications operate on some variant of the pinhole camera metaphor, i.e. a camera object exists in the virtual environment, which regularly takes bi-dimensional snapshots of a computer-generated scene, to be displayed on a physical device (Fig. 3). According to this model, programmers may simply select a horizontal FOV, specify an aspect ratio, declare the distances from the near clipping plane and the far clipping plane, and build the projection matrix.
For instance, the OpenGL[1] function *gluPerspective* [34] sets up a perspective projection matrix based on four user specified parameters (*r*, *t*, *n* and *f*). This entails the use a

[1] OpenGL (Open Graphics Library) is a cross-language, multi-platform Application Programming Interface (API) for rendering 2D and 3D vector graphics

symmetrical frustum[2] such as the one represented in Fig. 3. You may find extensive information about the OpenGL projection matrix and *gluPerspective* input parameters either on the Internet [34, 35] or in the OpenGL Programming Guide, alias The Red Book[35, 36]. Also, be aware that alternatives to the OpenGL Application Programming Interface (API) exist [37]. However, as long as a 3D content must be displayed on a 2D media, rest assured that a projection matrix exists.

When *gluPerspective* is invoked, it builds a projection matrix that looks like this:

$$P = \begin{bmatrix} \frac{n}{r} & 0 & 0 & 0 \\ 0 & \frac{n}{t} & 0 & 0 \\ 0 & 0 & \frac{n+f}{n-f} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \qquad (1)$$

Where *r* and *t* represent half of the horizontal and vertical near clip plane extents respectively, while *n* (nearVal) and *f* (farVal) refer to the distances between the viewpoint (i.e. the eye-space origin) and the near and far clipping planes respectively.
When (1) is used, a few underlying assumptions have been made, and that is that (a) the viewer is positioned in front of the screen, (b) facing perpendicular to it and (c) looking at the centre of it. This is also known as the 'on-axis' projection model (Fig. 3). As long as the projection matrix does not change, relative movements between the eyes and the screen (e.g. back and forward movements) are forbidden, as they modify the physical FOV whereas the projection model (i.e. the projection matrix) remains the same. In order to free up the viewpoint position from the screen normal[3], OpenGL provides a second function (*glFrustum*) that sets up the projection matrix as follows:

$$P = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{n+f}{n-f} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \qquad (2)$$

Where *l*, *r*, *b* and *t* denote the distances between the near clipping plane edges and the straight line that goes from the camera origin to the plane itself (in a perpendicular manner). Again, you may find extensive information about *glFrustum* input parameters

---

[2] A *frustum* is a six-sided truncated pyramid that originates sectioning the shape the virtual camera FOV by means of two user-defined clipping planes. These are known as the 'far clipping plane' and the 'near clipping plane'. The latter is the one on which the scene must be projected as a necessary step of the rendering pipeline.

[3] For the sake of readability, we refer to the straight line being orthogonal to the screen and passing by the centre of it simply as the screen normal.

– namely *l* (left), *r* (right), *b* (bottom), *t* (top), *n* (nearVal) and *f* (farVal) – either on the Internet [38] or in the in the OpenGL Programming Guide [36].

Most importantly, a projection matrix such as (2) allows for asymmetric frusta to be used. In other words, the viewpoint position is freed from the screen normal. This is known as the off-axis projection model (Fig. 4). As a matter of fact, the projection model delivered by (2) is much more flexible than the one provided by (1). E.g., the frustum extents can be determined separately for each eye-screen pair, resulting in an much more accurate projection model for stereovision implementation [33]. However, there are still a few constraints. For instance, *glFrustum* assumes that the near clipping plane is orthogonal to the virtual camera depth axis (i.e. the eye-space coordinate system e_n axis). Also, relative movements between the screen and the viewpoint position are still forbidden (unless accounted for by the tracking system).

Eventually, the field of AR introduces circumstances under which the assumptions of both *glFrustum*() and *gluPerspective*() fail and the resulting incorrectness is not tolerable [33, 39, 40]. For instance, when dealing with very large format HUDs for the airport tower, the overlay between the scene-linked symbols and the background view strongly depends on the viewer's eyes position with respect to the HUD and on the HUD orientation. Hence, a far more generic perspective model is needed.

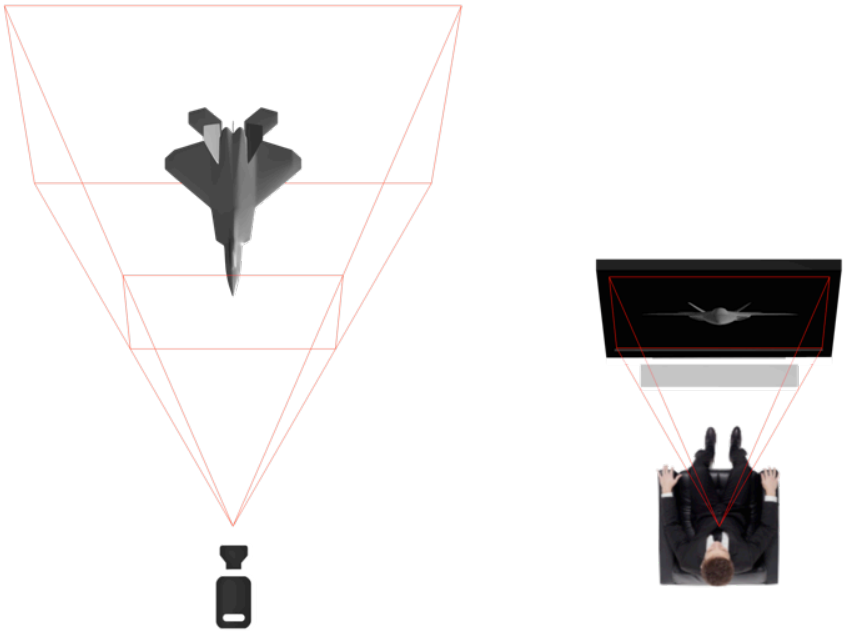

Fig. 2. The symmetrical frustum projection model, a.k.a. on-axis projection model.

4 Formulation of a custom rendering pipeline

Our first objective is to develop formulas allowing us to compute the parameters of a standard 3D perspective projection matrix (*l*, *r*, *b* and *t*) based on the relative position and orientation between the viewer's eyes and the screen. In order to constantly feed these parameters to the projection matrix a constant link between the tracking system and the projection matrix is needed. This is depicted in Fig. 8. Also it is mandatory to know the exact transformation between the tacker-space coordinate system and the world space-coordinate system. However, this can be easily determined once the location and the orientation of the tracking device(s) are fixed and known.

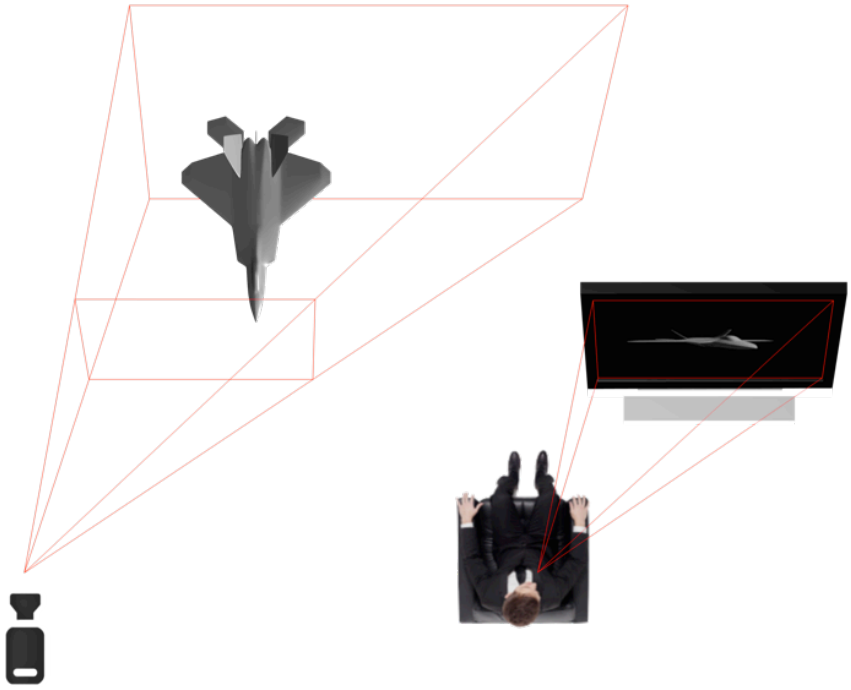

Fig. 3. The skewed frustum projection model, a.k.a. off-axis projection model.

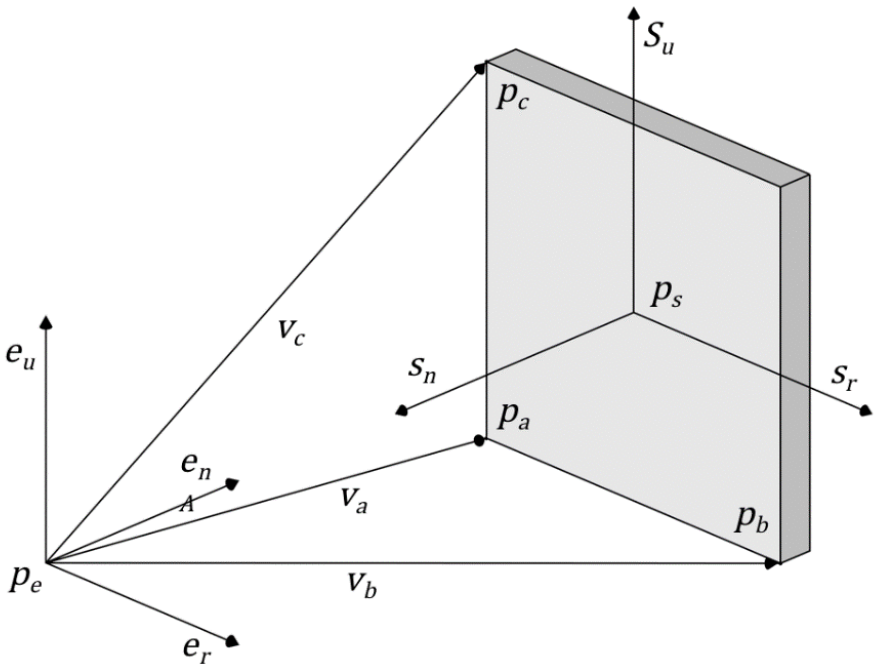

Fig. 4. The eye-space coordinate system (origin p_e), the screen-space coordinate system (origin p_s) and the screen corners vectors v_a, v_b and v_c.

Let's start reviewing the main characteristics of the AR system. These are the coordinates of the display corners, the origin of screen-space coordinate system, and the distance from the eye-space coordinate system origin to the screen (Fig. 5 and 6).
The coordinates of the head-up display corners, namely p_a (lower left corner), p_b (lower right corner), and p_c (upper left corner) are expressed with respect to world-space coordinate system. Assuming that flat screen is used, the position of the fourth point is implicit. Together these points encode the size of the screen, its aspect ratio, its position and orientation. Also, they can be used to compute an orthonormal basis for the screen-space coordinate system[4]. We refer to this basis as the triad of vectors composed by s_r (the vector toward the right), s_u (the vector pointing up), and s_n (the vector normal to the screen, pointing in front of it).

[4] In linear algebra an orthonormal basis for an inner product space is a basis whose vectors are all unit vectors orthogonal to each other.

$$s_r = \frac{p_b - p_a}{|p_b - p_a|} \quad (3)$$

$$s_u = \frac{p_c - p_a}{|p_c - p_a|} \quad (4)$$

$$s_n = \frac{s_r \times s_u}{|s_r \times s_u|} \quad (5)$$

The origin of the screen space coordinate system is the intersection between the perpendicular line drawn from p_e to the screen, and the plane of the screen itself. Since neither $\underline{p_e}$ nor p_s are fixed in space, when the viewer moves with respect to the screen, the screen-space origin changes accordingly. If s/he moves far to the side of the screen, then the screen space origin may not fall within the screen at all.

The distance from the eye-space origin p_e and the screen-space origin p_s may be computed by taking the dot product of the screen normal v_n with any of the screen vectors. However, because these vectors point in quite opposite directions, their product must be negated.

$$d = -(s_n \cdot v_a) \quad (6)$$

In order to compute the frustum extents we need the vectors from the camera space origin (p_e) to the screen corners. Once again, these can be easily calculated using the screen corners.

$$v_a = p_a - p_e \quad (7)$$

$$v_b = p_b - p_e \quad (8)$$

$$v_c = p_c - p_e \quad (9)$$

Frustum extents may be interpreted (and computed) as distances from the screen-space origin to the edges of the screen (as shown in Fig. 6). However, because these are not specified at the near clipping plane, we must scale them back from their value at the plane of the screen, d units away from the eye-space origin, to their value at the near clipping plane, n units away from the eye-space origin.

$$l = \frac{(s_r \cdot v_a)\, n}{d} \quad (10)$$

$$r = \frac{(s_r \cdot v_b)\, n}{d} \quad (11)$$

$$b = \frac{(s_u \cdot v_a)\, n}{d} \quad (12)$$

$$t = \frac{(s_u \cdot v_c)\, n}{d} \quad (13)$$

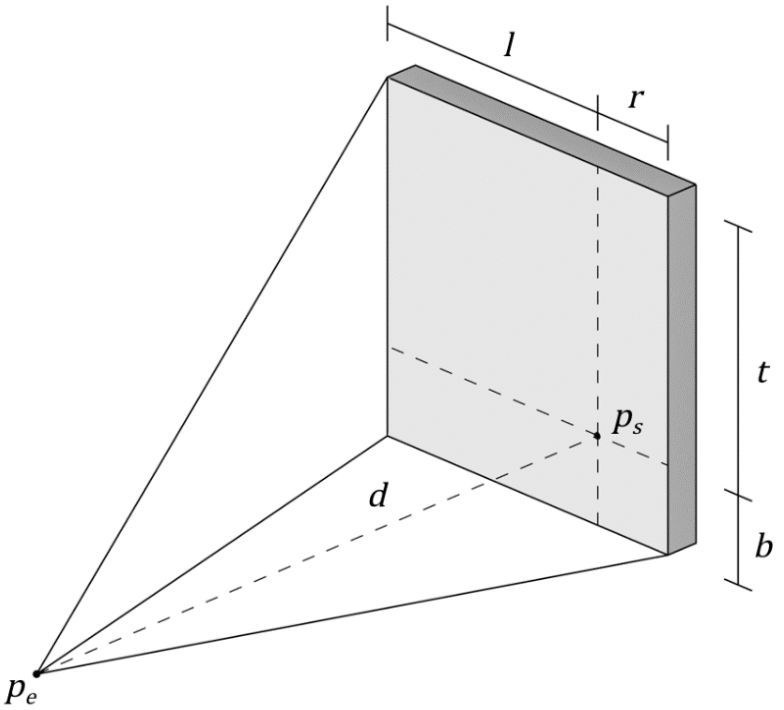

Fig. 5. The length of the frustum extents (*l*, *r*, *b* and *t*) at the plane of the screen.

Inserting these values into the standard perspective projection matrix allows us to build a head-tracked (or eye-tracked), off–axis projection (Fig 7). In other words, we now have the ability to create a skewed frustum for an arbitrary screen viewed by an arbitrary 'eye'.

There is one final limitation that we must work past, and that is that the near clipping plane is orthogonal to the eye-space depth axis (e_n), whereas the plane of the screen may not be. In practice, we need to free the projection plane from the orientation of the e_u-e_r plane. Unfortunately the way the standard perspective projection matrix was built simply disallows this. What we can do instead is to rotate the virtual world in order to line up the desired projection plane with the e_u-e_r orientation. As far as the projection outcome is concerned this is equivalent to rotating the viewing frustum aligning the near clipping plane to the plane of the screen. Note that this operation does not affect the frustum extents calculation.

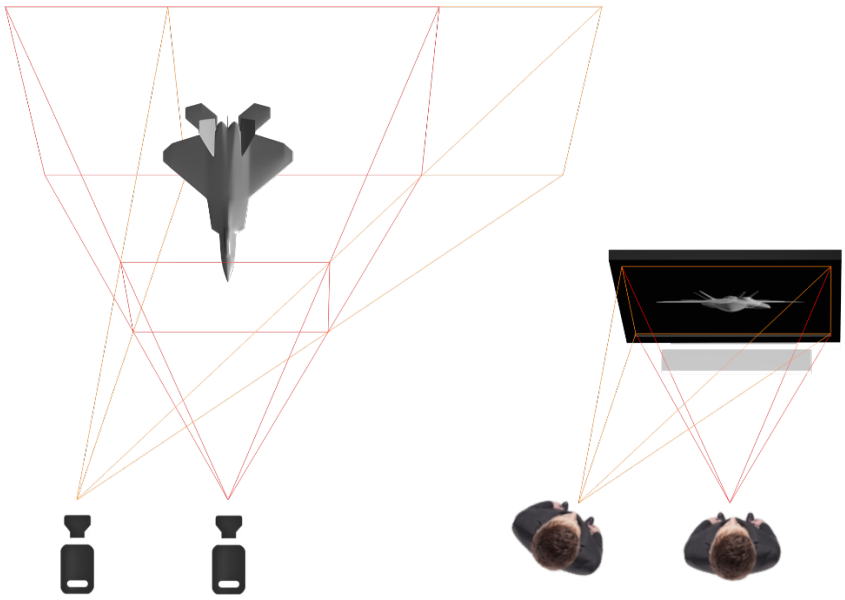

Fig. 6. Example of a head-tracked off-axis perspective.

We can build a transformation matrix that rotates the eye-space coordinate system so that its standard axis e_r, e_u and e_n match the orientation of the screen-space coordinate system like so:

$$R = \begin{bmatrix} s_{rx} & s_{ux} & s_{nx} & 0 \\ s_{ry} & s_{uy} & s_{ny} & 0 \\ s_{rz} & s_{uz} & s_{nz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

Which can be easily deduced from:

$$R \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = s_r \tag{15}$$

$$R \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = s_u \tag{16}$$

$$R \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = s_n \tag{17}$$

If something lies on the e_u-e_r plane, this transformation will align it to the plane of the screen. However, what we really need is the inverse mapping:

$$R^{-1}s_r = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{18}$$

$$R^{-1}s_u = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{19}$$

$$R^{-1}s_n = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{20}$$

Which is produced by the inverse of R:

$$R^{-1} = \begin{bmatrix} s_{rx} & s_{ry} & s_{rz} & 0 \\ s_{ux} & s_{uy} & s_{uz} & 0 \\ s_{nx} & s_{ny} & s_{nz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{21}$$

However, since R is orthogonal, R^{-1} is simply its transpose:

$$R^{-1} = R^T \tag{22}$$

Applying this transformation to all the objects in the virtual world will rotate the scene until the plane of the screen lines up with the e_u-e_r plane, which is exactly we needed. If we compose the standard projection matrix P with the rotation matrix R^T, the resulting matrix M covers everything we need and will work under any circumstances.

$$M = R^T P \tag{23}$$

With this matrix, we are finally able to render the scene in order to generate the AR layered.

5 Conclusion

Various analysts have estimated the benefits of using AR tools in control tower operations. However it is unclear which one between the head-worn AR technology and the spatial AR technology will prevail. Also, it has been rarely specified how such tools should be designed and operated. Our review confirms that many problems must be addressed before these tools become operational. However, there is ample reason to believe that, eventually, this will happen.

With regard to the use of very large format AR displays, we have developed an advanced rendering pipeline that is capable of generating registered overplays for multiple, arbitrary oriented head-up displays, defined together in a common coordinate system (Fig 8). Our concept is based on the excellent work by Robert Kooima (Electronic Visualization Laboratory, University of Illinois) [39]. However, because of a different target and development framework, the here-proposed implementation turned out to be a quite different thing. For instance, we do not consider the eye-space coordinate system to be bound to the world-space coordinate system (which probably seems quite reasonable to game engine developers).

In this document, the algorithm description has been deliberately mathematical, (i.e. not linked to any specific development framework or programming language). However, at our facilities, we have developed (and tested) a Python/C# code using an open source game engine and a Kinect™ for Windows™ tracking sensor. We plan to use this software as a basis for further research in the field of spatial see-through HUDs for the control tower.

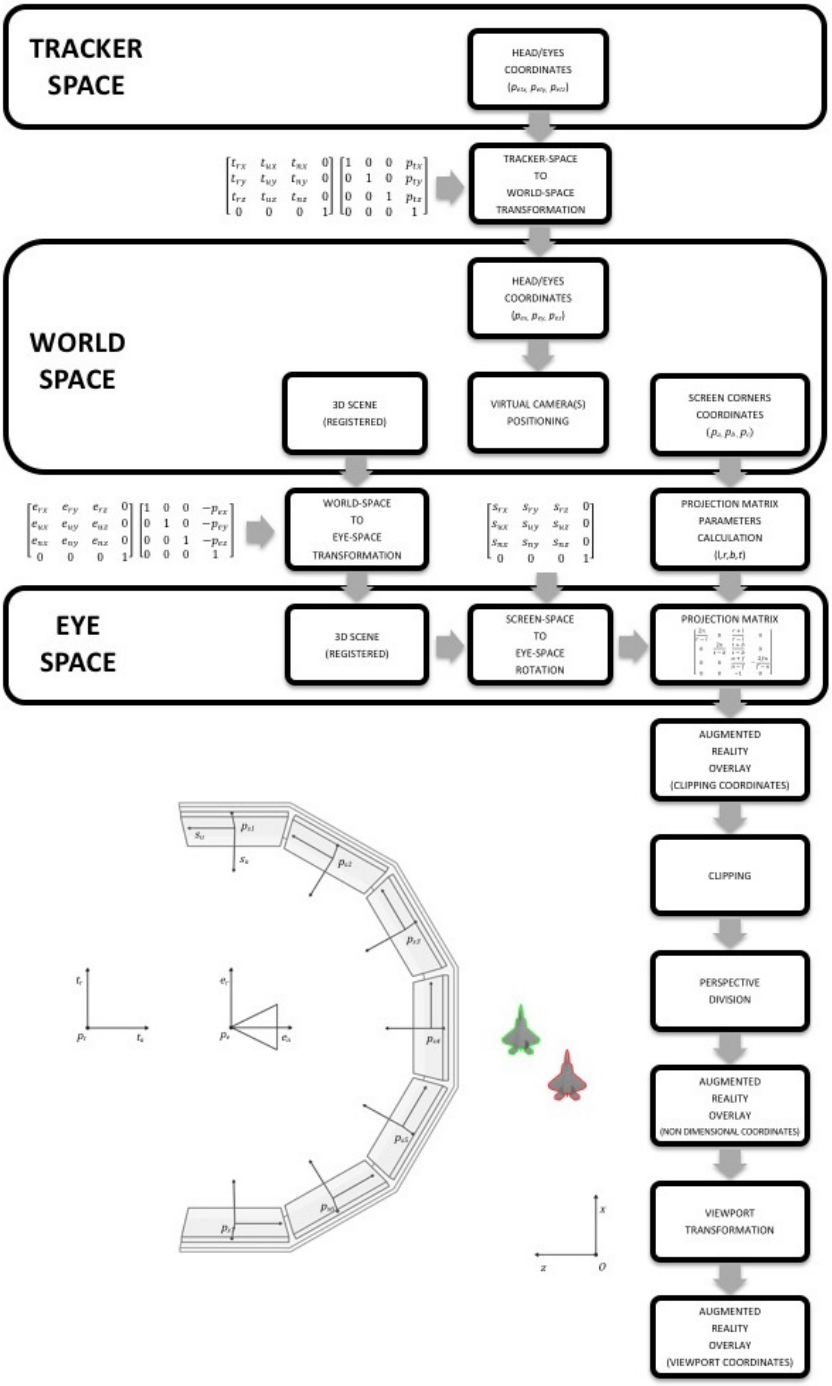

Fig. 7. An overall schematics of the modified rendering pipeline

References

1. Office for Official Publications of the European Communities: Single European Sky - Results from the transport research programme, http://www.transport-research.info/sites/default/files/brochure/20040617_111440_74437_singlesky.pdf.
2. SESAR JU: Enhanced Ground Controller Situational Awareness in all Weather Conditions, https://www.atmmasterplan.eu/oisteps/AO-0201.
3. Clean Sky JU: Clean Sky 2 - Joint Technical Programme, http://ec.europa.eu/research/participants/data/ref/h2020/other/guide-appl/jti/h2020-guide-techprog-cleansky-ju_en.pdf.
4. SESAR JU: Honeywell in SESAR: Advancing ATM in the heart of Europe, http://www.sesarju.eu/sites/default/files/documents/reports/SESAR_magazine_issue_7.pdf, (2012).
5. Reisman, R., Brown, D.: Design of Augmented Reality Tools for Air Traffic Control Towers. In: Proceedings of 6th AIAA Aviation Technology, Integration and Operations Conference (ATIO). American Institute of Aeronautics and Astronautics, Wichita, KS (2006).
6. Monica Tavanti: Control Tower Operations: A Literature Review of Task Analysis Studies. EEC Technical/Scientific Report No. 2006-013 (2006).
7. Ella Pinska: An Investigation of the Head-up Time at Tower and Ground Control Positions. In: Proceeding of the 5th Eurocontrol Innovative Research Workshop. pp. 81–86 (2006).
8. Schmidt, M., Rudolph, M., Werther, B., Möhlenbrink, C., Fürstenau, N.: Development of an Augmented Vision Video Panorama Human-Machine Interface for Remote Airport Tower Operation. In: Smith, M.J. and Salvendy, G. (eds.) HCI (9). pp. 1119–1128. Springer (2007).
9. Ellis, S.R.: Towards Determination of Visual Requirements for Augmented Reality Displays and Virtual Environments for the Airport Tower. In: Virtual Media for Military Applications RTO-MP-HFM-136. pp. 31–1 – 31–10. RTO, Neuilly-sur-Seine, France (2006).
10. Shackelford, R, Karpe, P.: Low/Zero Visibility Tower Tools (L/ZVTT) Benefits Assessment. , NASA Ames Research Center Technical Reports, Moffett Field, CA 94035 (1998).
11. Neveu, C., Blackmon, T., Stark, L.: Evaluation of the Effects of a Head-mounted Display on Ocular Accommodation. Presence. 7, 278–289 (1998).
12. Stephen R. Ellis, Bernard D. Adelstein, Ronald J. Reisman, Joelle R. Schmidt-Ott, Jonathan Gips, Jimm Krozel: Augmented Reality in a Simulated Tower Environment: Effect of Field of View on Aircraft Detection. NASA Ames Research Center Technical Reports (2002).
13. Weintraub, D.J., Ensing, M.J.: Human factors issues in head-up display and design: the book of HUD. Crew System Ergonomics Information Analysis Center, Wright-Patterson AFB, Ohio (1992).
14. Herman Rediess: An augmented reality pilot display for airport operations under low and zero visibility conditions. In: Guidance, Navigation, and Control Conference. American Institute of Aeronautics and Astronautics (1997).
15. Ruffner, J.W., Deaver, D.M., Henry, D.J.: Requirements analysis for an air traffic control tower surface surveillance enhanced vision system. Presented at the SPIE-The International Society for Optical Engineering (2003).
16. N. Fürstenau: Virtual Tower. 5th ATM R&D Symposium, Braunschweig. (2005).
17. Azuma, R.: A Survey of Augmented Reality. In: Presence: Teleoperators and Virtual Environments 6. pp. 355–385 (1997).

18. Reisman, R.J., Feiner, S.K., Brown, D.M.: Augmented Reality Tower Technology Flight Test. In: International Conference on Human-Computer Interaction in Aerospace (HCI-Aero). , Santa Clara, CA (2014).
19. Ronald J. Reisman, David M. Brown: Augmented Reality Tower Technology Assessment. NASA Ames Research Center Technical Reports (2010).
20. De Crescenzio F., Bagassi S., Fantini M., Lucchi F.: Virtual Reality based HUD (Head Up Display) to simulate 3D conformal symbols in the design of future cockpits. Presented at the Council of European Aerospace Societies, Venice, Italy (24-28 October, 2).
21. Sara Bagassi, Francesca De Crescenzio, Francesca Lucchi, Franco Persiani: Innovation in Man Machine Interfaces: Use of 3D Conformal Symbols in the Design of future HUDs (Head-Up Displays). Presented at the 28th International Congress of The Aeronautical Sciences, Brisbane, Australia September 23 (2012).
22. SESAR JU: System Wide Information Management (SWIM), http://www.sesarju.eu/sesar-solutions/swim.
23. SESAR JU: SWIM Concept of Operations, https://www.eurocontrol.int/sites/default/files/publication/files/del08.01.01-d41-swim_conops.pdf.
24. SESAR JU: Time Based Separation, http://www.sesarju.eu/sesar-solutions/airport-integration-and-throughput/time-based-separation.
25. SESAR: 06.09.03 Remote & virtual TWR, https://www.atmmasterplan.eu/projects/16484.
26. N. Fürstenau, M. Schmidt, M. Rudolph, C. Möhlenbrink, A. Papenfuß, S. Kaltenhäuser: Steps Towards the Virtual Tower: Remote Airport Traffic Control Center (RAiCe). Presented at the ENRI International Workshop on ATM/CNS, Tokyo, Japan (2009).
27. D. Schulz-Rückert: Future of Aerodrome Traffic Control. Presented at the Aachen Aviation Convention, Aachen, Germany (2007).
28. Bimber, O., Raskar, R.: Spatial Augmented Reality: Merging Real and Virtual Worlds. A. K. Peters, Ltd., Natick, MA, USA (2005).
29. Peterson, S.: Very large format stereoscopic head-up display for the airport tower. In: In Proceedings of the Virtual Images Seminar, number 16. CNRS/Renault (2007).
30. Peterson, Stephen, Ella Pinska: Human Performance with simulated Collimation in Transparent Projection Screens. In: Proceedings of the Second International Conference on Research in Air Transportation (ICRAT) (2006).
31. Rolland, J.P., Baillot, Y., Goon, A.A.: A Survey of Tracking Technology for Virtual Environments. Fundamentals of Wearable Computers and Augmented Reality. 67–112 (2001).
32. Zhou, F., Duh, H.B.-L., Billinghurst, M.: Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. In: Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on. pp. 193–202 (2008).
33. Nicola Masotti, Franco Persiani: Gaze-Coupled Perspective for Enhanced Human-Machine Interfaces in Aeronautics. Presented at the Conferences in Air Transport & Operations, Delft University of Technology, Delft, the Netherlands. July 20 (2015).
34. OpenGL: gluPerspective, https://www.opengl.org/sdk/docs/man2/xhtml/gluPerspective.xml.
35. Song Ho Ahn: OpenGL Projection Matrix, http://www.songho.ca/opengl/gl_projectionmatrix.html.
36. Woo, M., Neider, J., Davis, T., Shreiner, D.: OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1999).

37. Microsoft: DirectX Graphics and Gaming, https://msdn.microsoft.com/en-us/library/windows/desktop/ee663274(v=vs.85).aspx.
38. glFrustum - OpenGL, https://www.opengl.org/sdk/docs/man2/xhtml/glFrustum.xml.
39. Kooima, R.: Generalized Perspective Projection. J. Sch. Electron. Eng. Comput. Sci. (2009).
40. Liverani, A., Persiani, F., De Crescenzio, F.: An Immersive Reconfigurable Room (I.R.R.) for Virtual Reality Simulation. In: Electronic Proceedings of 12th International Conference on Design Tools and Methods in Industrial Engineering. pp. E1–9 – E1–16, Rimini, Italy (2001).