VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUGMENTED REALITY FOR TRAINING AND EXECUTION OF AIRPLANE MAINTENANCE

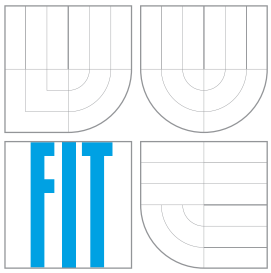
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL KOŠÍK

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POUŽITÍ ROZŠÍŘENÉ REALITY PŘI TRÉNINKU A PROVÁDĚNÍ ÚDRŽBY LETADEL

AUGMENTED REALITY FOR TRAINING AND

EXECUTION OF AIRPLANE MAINTENANCE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL KOŠÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. ADAM HEROUT, Ph.D.

BRNO 2013

Abstrakt

Tato práce pojednává o problematice rozšířené reality a její využití při tréninku servisních techniků jako i při provádění servisních procedur. Poskytuje přehled metod používaných při rozšířené realitě se zaměřením na jejich využití při trénování a vykonávání technické údržby. Rovněž je tady přehled předchozích prací zabývajících se touto problematikou. Na otestování možností, které rozšířená realita poskytuje, byl vytvořen prototyp a na základě jeho výsledků a studiu předchozích prací byla následně vytvořena hardwarová specifikace finálního systému spolu s definicí jeho návrhu. Tento systém bude v budoucnu umožňovat servisnímu technikovi, který bude mít na hlavě nasazenou malou kameru snímající jeho zorné pole, počítač a brýle s displejem, rozšířit reální svět viděný v brýlích o další informace týkající se aktuálně prováděného kroku servisní procedury. Tyto informace mohou být textové nebo grafické, jako např. zvýraznění komponenty, s kterou bude technik v rámci aktuálního kroku procedury pracovat, nebo zobrazení doplňující grafiky s např. schématem dané komponenty. Za účelem získání autentické datové sady byla kontaktována firma LET vyrábějící letoun LET L-410, která pro účely této práce poskytla letoun, prostory a své techniky k natočení videí obsahujícími servisní procedury. Na natočení těchto videí byla použita kamera Contour+2, kterou je možné jednoduše přichytit k hlavě. Natočena videa ale obsahovala řadu nedostatků – pořízena kamera nedokázala kvůli horším světelným podmínkám v hangáru, kde se procedury natáčeli, poskytnout dostatečně kvalitní videa. Kromě toho, kvůli širokému zornému poli, které tato kamera ponoukala, natočené videa obsahovali vysokou úroveň distorze. Tato distorze byla odstraněna pomocí OpenCV. Následně byla podle zkušeností získaných z prototypu a nastudované literatury vytvořena finální aplikace. Tato aplikace byla napsána v jazyce C++ a využívala kromě knihovny OpenCV i knihovnu ALVAR, která obsahuje algoritmy a detektory určeny pro rozšířenou realitu. Taky byla vytvořena digitální reprezentace servisní procedury, která v sobě zahrnuje textovou dokumentaci, cesty k dodatečným technickým schématům a taky cesty k jednotlivým detektorům. Speciálním rozšířením tu jsou tzv. Oblasti zájmu (Areas of Interest, zkratka AoIs), které určují, kde přesně se na detekovaném objektu nachází komponenta, s kterou se má v aktuálním servisním kroku manipulovat. Toto rozšíření umožňuje hledat v obraze i mnohem větší objekty, v rámci kterých se nacházejí skutečně hledané komponenty. Za účelem zvýšení úspešnosti počtu detekcí, která byla zpočátku nízká, bylo implementováno několik rozšíření – validace matic homografie, které se počítají pro každý detektor, přineslo výrazné snížení počtu chybných detekcí. Představení historie, která si pamatuje několik předchozích detekcí a na jejich základě upravuje pozici aktuálně vykreslovaných objektů, přinesla větší stabilitu při vykreslování těchto objektů, které už mezi jednotlivými snímky neměnili natolik svou pozici v scéně. Taky byla implementována možnost spustit v rámci jednoho kroku procedury vícero detektorů, ta však nebyla nakonec využita, protože byla výpočetně příliš náročná. Na konci této práce se nachází kapitola věnována testování, kde jsou jednotlivé algoritmy otestovány a jsou navrženy vylepšení, které budou implementovány v dalším pokračovaní této práce, která byla tvořena ve spolupráci s firmou Honeywell. I přestože se kvůli nedostatkům v datové sadě a detektoru nepodařilo přinést úplně funkční řešení, znalosti získané tvorbou této diplomové práce budou využity na jejím pokračovaní v rámci firmy Honeywell.

Abstract

This paper provides an overview of methods used with augmented reality applications with emphasis on their use in training and execution during maintenance procedures. In

addition, a description of the previous work done in this field is provided and analyzed. First, a prototype application testing the possibilities of the augmented reality is created. Subsequently, based on the experience gained from this prototype and the previous work analyses, a final application is designed, implemented and tested on a set of video recordings with captured aircraft maintenance procedures. Finally, this application's performance is evaluated and several improvements to its functionality are suggested. At the end, the results are summarized and the future work's direction is set.

Klíčová slova

rozšířená realita, údržba, trénování, letadlo, počítačové vidění, ALVAR, Fern, head-mounted displej, area of interest.

Keywords

augmented reality, maintenance, training, airplane, computer vision, ALVAR, Fern, head-mounted display, area of interest.

Citace

Michal Košík: Augmented Reality for Training and
Execution of Airplane Maintenance, diplomová práce, Brno, FIT VUT v Brně, 2013

Augmented Reality for Training and Execution of Airplane Maintenance

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením pana Adama Herouta

.....
Michal Košík
May 22, 2013

Poděkování

Týmto by som sa rád poďakoval svojmu vedúcemu Doc. Ing. Adamovi Heroutovi, Ph.D. a Ing. Matějovi Dusíkovi zo spoločnosti Honeywell za ich profesionálny prístup, cenné rady a ochotu pomôct pri riešení problémov, ktoré sa pri tvorbe tejto práce vyskytli. Takisto by som sa chcel podakovať pánovi Hynkovi Fojtíkovi za jeho spoluprácu pri natáčaní leteckých servisných procedúr. Veľká vďaka rovnako patrí mojim rodičom a Róbertovi Kalmárovi, Lenke Sedláčkovej, Helene Vondrovej, Helene Duríkovej a Márii Tarbajovej, ktorí mi boli oporou v momentoch, keď som to najviac potreboval.

Contents

Chapter 1

Introduction

Augmented reality (AR) — as a part of mixed reality described by [25] — can be defined as a system that enhances the real world with some computer-generated virtual objects (either 2D or 3D) which appear to coexist in the same space as the real world and thus enhances user's experience of the world as well.

Such system has the following list of properties [2]:

- Combines real and virtual objects in a real environment.
- Runs interactively, and in the real time.
- Aligns real and virtual objects with each other.

This definition puts no restrictions on which human senses the augmented reality should be applied to. Nevertheless, in this work only the augmentation of visual information received from the real world will be presented.

Augmented reality has many applications starting with showing tourist information based on what the tourist is looking at, through playing interactive games using the real world as a playground, up to helping with design by placing virtual models to the real environment [2].

On the other hand, maintenance is a process which goal is to ensure a maintained object is functional and to minimize the probability it fails.

The augmented reality for maintenance is a special case of augmented reality; usually, a technician equipped with a head-mounted display and a camera is navigated step-by-step through a maintenance procedure. Information connected with the current step of procedure is shown to the technician in order to increase efficiency and reduce the time the maintenance requires.

In this work, the following objectives were set:

- Research possibilities provided by augmented reality for training and execution of airplane maintenance.
- Explore existing solutions, together with toolkits and algorithms for augmented reality suitable for this application.
- Propose a hardware configuration of a final output system.
- Create a prototype and test its possibilities.

- Propose a future work and tests.

Cooperation with Honeywell's Advanced Technology division (located in Brno, Czech Republic) was established; therefore output of this work can be tested in real-world situations. However, it is important to mention this project has a goal to explore the potential benefits of using the AR for supporting maintenance procedures, not to bring a production-ready implementation. Therefore, the software and hardware choices made for this project may not reflect the needs of a commercial application.

This work consists of nine chapters that cover the introduction to AR; training and airplane maintenance; related work; datasets acquisition and analyses; design, application and testing of created prototypes; and a summary of reached results together with suggestions of the future continuation of the project.

The second chapter provides introduction to augmented reality with emphasis on its use in maintenance applications. Different types of displays and algorithms are explained. In addition, the most suitable AR toolkits are presented and previous applications of the AR in maintenance (including such companies as BMW of Airbus) are examined.

In the third chapter, a basic insight into airplane maintenance is provided including an overview of maintenance intervals, the most important types of documentation for performing maintenance procedures, and a description of L-410 aircraft which maintenance procedures were used for demonstration of the final application.

The goal of the fourth chapter is to summarize previous work done on this project during Term Project. This work is also clearly distinguished from the work done afterwards. Additionally, a first prototype created for this project is described and analyzed here.

The fifth chapter specifies requirements for hardware in terms of its usage with maintenance and AR. Consequently, specific hardware and software components are chosen to be used in this work and are briefly described as well.

In the sixth chapter, the main dataset acquisition is described along with selected procedures' description and a list of requests these procedures had to meet. Furthermore, obtained datasets are analyzed.

The seventh chapter offers an overview of created algorithms and explains reasons for their creation and usage.

Subsequently, the eight chapter tests algorithms implemented or used in the previous chapter and provides and explains their results.

Finally, the ninth chapter concludes all the work already done and presents suggested plans for continuation of this thesis.

Chapter 2

Augmented reality in maintenance applications

Current maintenance systems require a maintenance engineer[1] to continually consult his or her progress within a maintenance procedure with blueprints, manuals, computer screens, or other references which are often out of his or her field of view. This results in switching technician's focus back and forth between the references and the area where the maintenance takes place. In addition, the technician has to filter references for relevant information and match and localize parts from the references in the repairing area what can be time-consuming, considering for example aircraft or vehicle systems that are rather complex.

A possible approach to this problem is to use augmented reality – by wearing a head-mounted display and a point-of-view camera, technician's natural view can be augmented with instructions, labels, descriptions of individual parts, various animations and so on. An effective assistance in these instances can also save technician's time while reducing mental workload [12].

The goal is to reduce the amount of time necessary for the technician to perform a maintenance procedure along with reduction of human errors. By incorporating the AR, this can be done by delivering information on current procedure's step directly to the technician with a minimum amount of communication between the technician and the source of information. In one of its studies, Boeing compares time efficiency and error robustness between information delivered by augmented reality and web-based interface with voice recognition and text-to-speech technology [17]. The time necessary to complete a maintenance procedure was shorter when augmented reality was used.

Another comparison between the augmented reality and the web assisted maintenance used for maintenance inside armored vehicle turret also resulted in a shorter time needed when augmented reality was used, showing a promising way of handling maintenance and repairs [13].

In this chapter, basic requirements for the AR are explained starting with types of displays used, describing the methods and algorithm applied, and with an overview of AR toolkits that offer marker-less tracking suitable for this work and ending with examples of use of AR in maintenance by universities and commercial companies.

[1] In this work, maintenance engineer is referred to as a technician or a mechanic

2.1 Display types

Because of its nature, it is logical to use the AR[2] with some sort of display. In general, three types of display are being used with the AR: See-through head-mounted displays (HMD), Projection-based displays and handheld displays [36].

Handheld displays are small portable displays; commonly, they are displays on smartphones, PDAs and tablets.

Projection-based displays display graphical information directly on real objects, therefore users do not need to wear any special equipment.

2.1.1 See-through head-mounted displays (HMD)

HMDs are the most often used displays with maintenance; therefore they will be described in more detail here. There are two types of see-through HMDs – optical see-through (OST) and video-see through (VST) displays. They serve the same goal – to insert computer-generated objects directly in the user's field of view and hence enhance his or her experience of the real world. What differs is the way how this goal is achieved. Figure 2.1 shows the difference between OST and VST HMD.

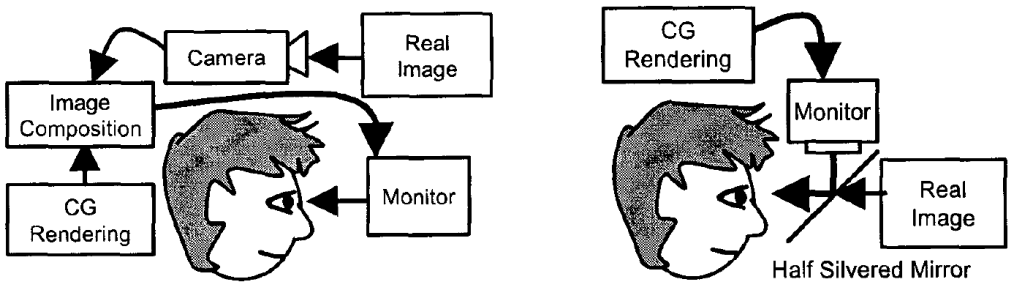


Figure 2.1: Difference between VST HMD (left) and OST HMD (right) [20]. Depending on application OST HMD may include a camera as well.

OST HMDs usually involve a semitransparent eye-glass allowing their user to see the real world with his or her own eyes and show computer-generated content on it at the same time. Therefore, OST HMDs offer an instantaneous, natural view of a real scene and usually also wider field of view. The main challenge with this displays is a correct positioning of virtual objects to the real world [20].

On the other hand, VST HMDs offer a video of the real world enhanced with virtual objects, so the positioning problem present at OST HMDs doesn't occur here. In addition, the real-world video can be processed before showing on the display, so a variety of image-processing techniques (e.g. intensity correction) can be applied [20].

2.1.2 Available OST HMDs overview

Recently, several different OST HMDs were introduced. However, only Google Glass and Lumus OE-32 will be described in this section because of their availability for developers and good support from manufacturers.

[2]Only visually augmented reality is described in this work.

Figure 2.2: Google Glass – available on the market [11].

Google Glass developed by Google X Lab is a lens-less glass with only a frame that incorporates a camera, a display and a microphone as can be seen on Figure 2.2. In addition, it offers a dual core processor, 1 GB of RAM and is running on Android 4.0.4. It offers wired or wireless connection, speakers, a microphone and high-level voice commands recognition [10]. The main properties of this HMD can be found in Table 2.1.

At the time of this work being written, Google Glass was available only for developers with an expected release date for general public in Q2 2014.

Display format	640 x 360 pixels
Built-in camera	Yes (videos in 720p, photos in 5 MP)
Field of View	40°
Virtual screen size	Equivalent of 25" screen at approx. 2.7 m away
Connectivity	802.11b/g, Bluetooth, Micro USB
Storage	12 GB
Processor	OMAP 4430 SoC (dual-core)
RAM	1 GB

Table 2.1: Google Glass properties

Another OST HMD solution is model OE-32 from Lumus, which prototype is available in Honeywell. This display is inserted into glasses and it offers high resolution and see-through experience enabling the overlay of information over one's view of the real world. Also, the small dimensions of these glasses ensure a good wearing comfort.

The whole glasses can be found on Figure 2.3. The main properties of the glasses and near-to-eye display are available in Table 2.2. Since it is only a prototype no detailed specifications will be provided here. More information on this display can be found at [22].

Both solutions bear a different set of properties. In comparison to Google Glass, Lumus OE-32 offers higher display resolution with two displays that cover greater area of user's field of view and are ready for 3D mode; on the other hand, Google Glass comes up with smaller dimensions, integrated camera, microphone, and speakers connected with Google services and advanced voice recognition.

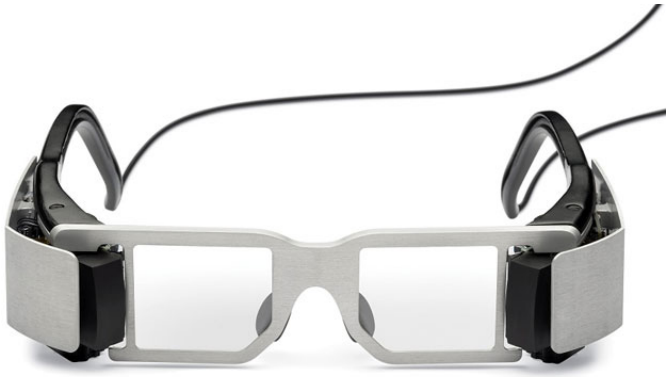


Figure 2.3: Lumus OE-32 – a near-to-eye display used in this thesis [22].

Display format	1280 x 720 pixels
Built-in camera	No
Field of View	40°
Virtual screen size Equivalent	87„ screen at approx. 3.3 m away
Connectivity	Micro USB
Transparency	$> 78\,\%$
Display thickness	1.6 mm
Virtual screen size	75 %

Table 2.2: Lumus OE-32 properties

2.2 Methods and algorithms used with augmented reality

2.2.1 Vision-based tracking

AR applications require an accurate knowledge of relative position and rotation of the camera and the scene (or an object in the scene). Therefore, real-time information of all six degrees of freedom defining the relation between the scene and the camera is required. To recover this information is the main goal of tracking, making the tracking one of the core techniques used in the AR [36].

Generally, there are sensor-based and vision-based tracking techniques. Although there were some experiments with AR maintenance focused on sensor-based tracking [35], in this work, video-based tracking techniques for marker-less tracking will be discussed. More on sensor-based tracking techniques can be found in [31].

Vision-based tracking consists of two phases:

- Image processing which extracts some information from the images
- Pose estimation

There are three most used approaches to vision-based tracking which description follows [9].

Markers

If possible, artificial markers are inserted into the real-world scene. These markers are easy to extract and provide reliable measurements for pose estimation [9]. However, in the airplane maintenance positioning such markers is rather inconvenient; therefore it will not be further explored.

Natural features

Another approach uses natural features [9] – these features can be either edge-based or texture-based.

Edge-based features are more resistant to changes in illumination and can be computed effectively. Usually, a strong gradient in the image is being searched for to find these features.

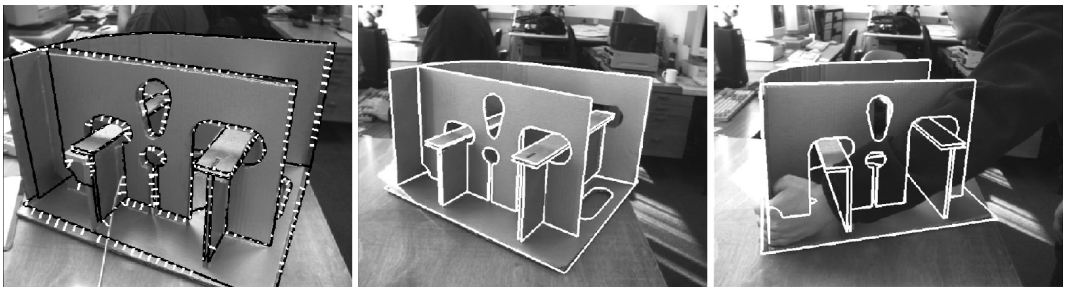


Figure 2.4: Example of an edge-based features algorithm (RAPiD algorithm) [9].

Texture-based features can be derived from optical flow, template matching or interest-point correspondences. However, the object has to be sufficiently textured.

Feature points extraction

These methods usually include an offline training stage, where a database of interest points lying on the object — and whose position on the object surface can be computed — is built. For this purpose, one to several images with the object manually registered in are required.

At runtime, feature points are extracted from the video images and matched against the database. Next, the camera position can be estimated from the correspondences of feature points.

Because the database images and the video-images may be acquired from different viewpoint, it is desired for these feature points to be highly distinctive and also invariant (or at least more or less robust) to scale, viewpoint, illumination changes, and partial occlusions. This may be done by characterizing each extracted feature point in terms of its image neighborhood and then to compare this characterization to the feature points stored in the database; such characterization is called local descriptor and can be found e.g. in SURF [9].

2.2.2 User interfaces and interaction with system

Considering the AR's purpose and possibilities it offers, a typical user interface (such as a hardware keyboard used for interaction) may not seem very appropriate. Hence, there are several different techniques that allow the user to interact with virtual content [36].

An interesting approach are tangible user interfaces (TUI) where digital information is manipulated with physical objects [16]. For example, Kato et. al. [18] uses a real paddle to select and move a virtual furniture projected to a surface of a real table; markers are used in this prototype. Another variants of TUI include e.g. mapping motion of a real object to a command and also combining it with gesture (e.g. a finger tracker that requires a special glove with retro-reflective markers [7]) and voice interaction what leads to multimodal interfaces [36].

Many other techniques that are used in AR — including marker-based tracking and collaborative user interfaces — were not described in this part, since they are not directly connected to the goals of this work. For a comprehensive overview of these methods and techniques used in AR field, the reader is advised to refer to Zhou et. al. [36].

2.2.3 Interpolation methods in two-dimensional discrete images

Interpolation in image is used for calculating a formerly unknown value v of point P — further expressed as $v(P)$ — by using its adjacent points. The unknown value may be a result of various image transformations (for example perspective transformations, resizing, or remapping image points to new coordinates when removing distortion) that create new points with an unknown value or remove redundant points (e.g. when downscaling an image) while trying to keep as much of original image information as possible.

Input for these method are points $\Pi = A, B, C, D, \ldots$; their values $v(A), (B), v(C), (D), \ldots$; and point P which value $v(P)$ will be computed. Output is an estimation of $v(P)$. Only basic interpolation methods and methods used in this paper will be described here. For all these methods, a point with unknown value is P and a set of its adjacent points is $\Pi, P \notin \Pi$. More information about these methods together with other interpolation methods can be found in [30].

Nearest neighbour interpolation

Nearest neighbour is the simplest interpolation method; it basically returns value of the nearest point to P, mathematically:

$$v(P) = v(A|\forall A, B \in \Pi : d(A, P) \leq d(B, P)) \tag{2.1}$$

Area interpolation [28]

Another simple interpolation method that calculates $v(P)$ as an average value of its adjacent points. Mathematically:

$$v(P) = \frac{\sum_{p \in \Pi} v(p)}{|\Pi|} \tag{2.2}$$

This method can be used with image downscaling since it is fast and gives moire'-free results [28].

Linear interpolation

Linear interpolation is a method of calculating $v(P)$ by fitting a straight line between two points A and B and determining its value at P, as equation 2.3 shows.

$$v(P) = v(A) + (v(B) - v(A))\frac{|P - A|}{|B - A|}, A, B \in \Pi \tag{2.3}$$

Bilinear interpolation

Bilinear interpolation uses — as its name states — two linear interpolation to determine a value of P laying in an area created by four adjacent points; in this case $\Pi = A, B, C, D$. First, linear interpolations P_{AB}, in between points A and B, and P_{CD} between points C and D are calculated. Then, linear interpolation of P_{AB} and P_{CD} results in final value of P. These equations are shown in 2.4.

$$\begin{aligned} v(P_{AB}) &= v(A) + (v(B) - v(A))\frac{|P_{AB} - A|}{|B - A|}, A, B \in \Pi \\ v(P_{CD}) &= v(C) + (v(D) - v(C))\frac{|P_{CD} - C|}{|D - C|}, C, D \in \Pi \\ v(P) &= v(P_{AB}) + (v(P_{CD}) - v(P_{AB}))\frac{|P - P_{AB}|}{|P_{CD} - P_{AB}|} \end{aligned} \tag{2.4}$$

2.2.4 Camera distortion and calibration

Since camera distortion was a serious problem that needed to be solved in this thesis, a closer explanation of distortion and camera calibration, together with a pinhole camera model, is provided here.

Pinhole camera

A very basic camera model is a *pinhole camera* - it is an imaginary wall with a pinhole (a small aperture) in its center allowing to pass through only a single ray of light from any particular point in a scene taken by this camera. This ray is then projected on an *image plane* (also called *projective plane*) and is always focused. The size of projected object relative to the size of the real object located in the scene is determined by a single camera parameter – its *focal length* (a distance between image and pinhole plane). Figure 2.5 demonstrates a mathematical representation of a pinhole camera where image plane is before the pinhole. In this representation a pinhole is reinterpreted as a center of projection.

The pinhole camera has no distortion. On the other hand, it does not gather enough light for allowing a rapid exposure. By adding lenses (that allow to gather more light than a pinhole camera) exposure time may be reduced. However, the lenses bring a more complex model than a pinhole camera model and they also add *distortions* [5].

In addition to distortions, another problem arises when using lens – *principle point* (a point at the intersection of the image plane and the optical axis) is usually not equivalent to the center of lens camera's imager. To model this possible displacement two parameters representing the shift from its expected position — c_x and c_y — are used [5]. Then, a projection of a point q from the real world into the image plane can be calculated as:

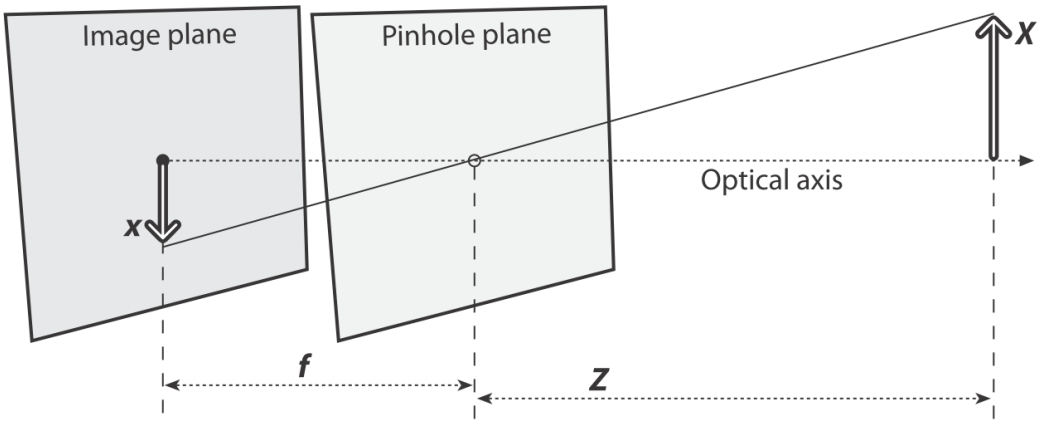


Figure 2.5: Mathematical definition of pinhole camera [5].

$$q = MQ, q = \begin{bmatrix} x \\ y \\ w \end{bmatrix}, M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, Q = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.5}$$

Here, q holds the homogeneous coordinates, M is *camera intrinsic matrix* containing *camera intrinsic parameters* that remain fixed for a specific camera, and Q is a position of point in 3D space. Elements f_x and f_y of matrix M represent focal length in x and y axis respectively. This is caused by individual pixels on the imager that may be rather rectangular than square.

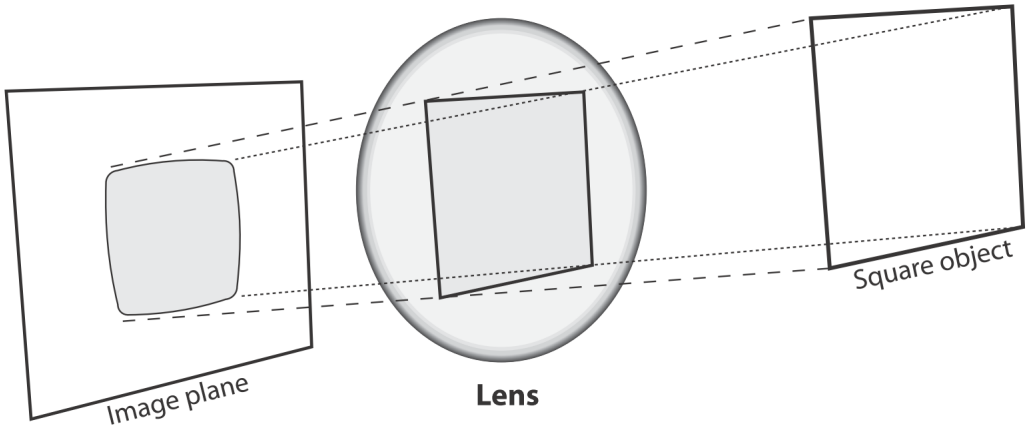


Figure 2.6: Example of radial distortion – rays farther from the center of lens are bent more than rays closer to the center [5].

Distortion

Although there are many types of distortion, *radial* and *tangential* distortions have the most significant effect (in comparison to the other types) [5]; hence, only these two types

of distortion will be described here.

Radial distortion is a result of a shape of lens (more commonly used spherical lens provides a higher level of radial distortion than more mathematically ideal parabolic lens; on the other hand, parabolic lens is more difficult to produce [5]). This distortion causes that lines become curves – this is because the rays farther from the center of the lens are bent more than the rays closer to it, as is shown on Figure 2.6. At the optical center of imager, distortion is zero and increases toward the periphery; therefore, radial distortion is most visible at the edges of the imager [9].

Removing radial distortion is rather simple – a radial location (x, y) of a point on the imager is rescaled to location $(x_{corrected}, y_{corrected})$, using following equation:

$$\begin{aligned} x_{corrected} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{corrected} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{2.6}$$

Here, r is distance from the optical center of imager and k_1, k_2, k_3 are the terms of a Taylor series expansion around $r = 0$. For majority of normal cameras it is sufficient to use only first two terms of this series expansion; for "fish-eye„ cameras it is recommended to use all three [5].

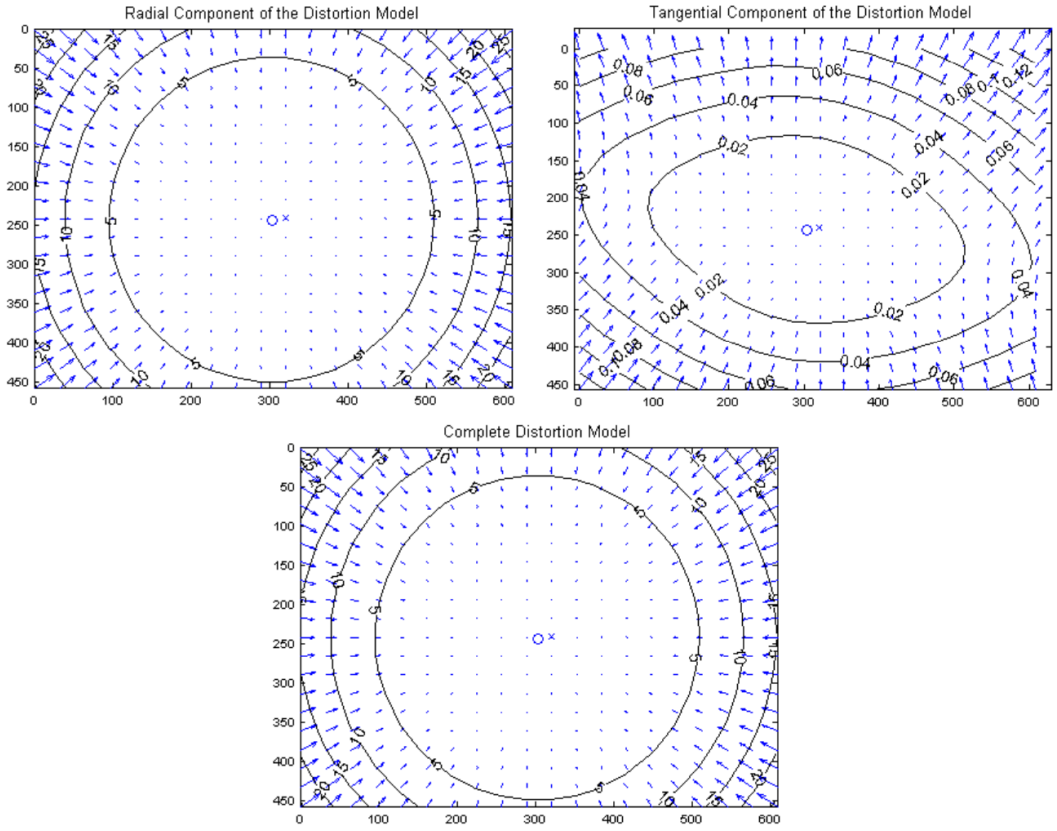


Figure 2.7: Displacements of points on a rectangular grid of camera's imager caused by radial (top left), tangential (top right) distortion, and their combination (bottom). [4].

Tangential distortion is caused by assembly process of the camera when the lens is

not parallel to the image plane. This distortion type is characterized by minimally two additional parameters, p_1 and p_2, that help to correct for the distortion as Equation 2.7 shows.

$$\begin{aligned} x_{corrected} &= x + (2p_1y + p_2(r^2 + 2x^2)) \\ y_{corrected} &= y + (p_1(r^2 + 2y^2) + 2p_2x) \end{aligned} \tag{2.7}$$

Comparison of radial and tangential distortion together with an example of their joint effect can be found on Figure 2.7. To remove tangential and radial distortion, all five coefficients — namely k_1, k_2, k_3, p_1, p_2 — must be known.

To mathematically correct from all these deviations from the simple pinhole model (in other words, deviations between the image taken by the camera and the real world), *camera calibration* is being used.

Camera calibration

Camera calibration is an *estimation* of camera intrinsic matrix, distortion coefficients and extrinsic matrix [21]. Extrinsic matrix is a 4×4 matrix containing camera's translation and rotation from its origin to observed object's origin.

Basically, there are two ways of calibrating camera: either one picture of an object with many known 3D points can be taken or several images from different viewpoints of an object with fewer 3D points are taken.

The second approach requires computing of the camera *extrinsic parameters* in each view; however it is usually used because of availability of algorithms for camera position computing and because its datasets are easier to obtain.

As a dataset, several images with chessboard or other easily recognizable object are often used [5]. Using a *chessboard*[3] has several advantages:

- It is a planar object.
- Its squares are easily identifiable and extractable.
- These squares lie in straight lines in real world.

Because the chessboard is a planar object, *planar homography* — a projective mapping from one plane to another — may be used map it to the surface of the camera's imager. This mapping of point Q on the planar object to the point q (both in their homogeneous coordinates) on the imager is defined as:

$$\begin{aligned} &\textit{Define } Q\textit{: } Q = [\ X \quad Y \quad Z \quad 1\ ]^T \\ &\textit{Define } q\textit{: } q = [\ x \quad y \quad 1\ ]^T \\ &q = sHQ \end{aligned} \tag{2.8}$$

Parameter s stands for a scale factor. H consists of two parts:

1. Physical transformation – uses rotation R and translation $\boldsymbol{t}$ to relate the plane we are viewing to image plane.

[3] Chessboard here represents a flat object with alternating white and black squares, not necessarily an 8×8 board for playing chess

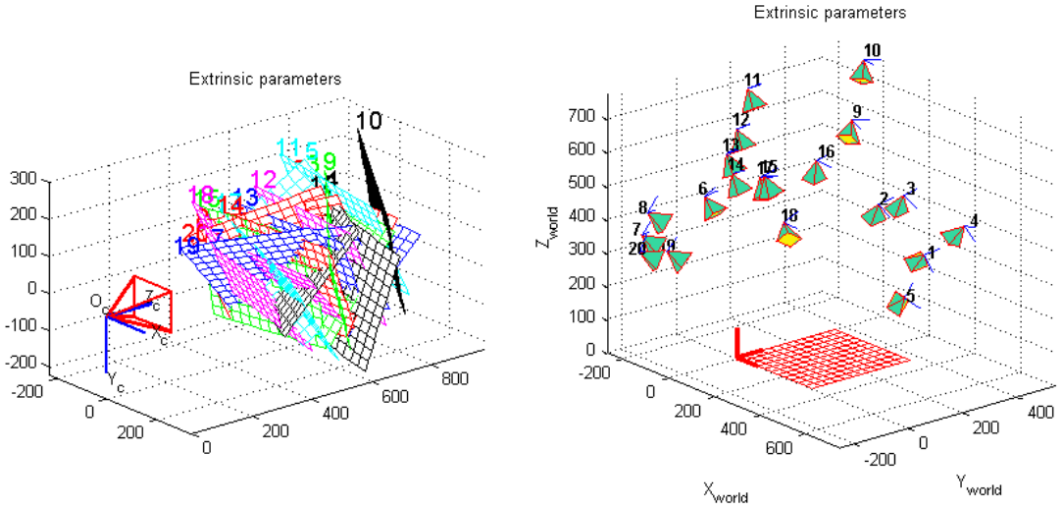


Figure 2.8: Transformation of 20 images of one object (a chessboard) from camera coordinates (left) to object coordinates (right) [4].

2. Projection – uses camera intrinsic matrix M.

Then, the equation 2.8 can be expressed as:

$$q = sMR\boldsymbol{t}Q \tag{2.9}$$

Without loss of generality, the object plane can be defined so that $Z = 0$ and thus reducing rotation matrix R, as Figure 2.8 shows.

Since rotation is defined by three angles and translation is described by three offsets, there are six unknown parameters unique for each view. Assuming that intrinsic matrix has only four parameters (as in the case of OpenCV implementation [21]) there are ten unknown parameters (and intrinsic parameters remain fixed). A planar object provides eight equations; it is a mapping of a rectangular into a quadrilateral that can be described by four (x, y) points. Thus, given sufficient amount of images any number of intrinsic parameters can be computed [5].

Intrinsic (together with extrinsic) and distortion parameters can be dealt with separately, since the former are tied to 3D geometry of where the planar object is in space and the latter are tied to the 2D geometry of how a pattern of points gets distorted. Therefore, when intrinsics and extrinsics are known, distortion parameters can be computed. By combining equations 2.6 and 2.7 an image point's location $(x_{corrected}, y_{corrected})$ equal to its location in a perfect pinhole camera can be obtained from its original distorted location (x_d, y_d) by following equation:

$$\begin{bmatrix} x_{corrected} \\ y_{corrected} \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} 2p_1 x_d y_d + p_2(r^2 + 2x_d^2) \\ p_1(r^2 + 2y_d^2) + 2p_2 x_d y_d \end{bmatrix} \tag{2.10}$$

With a list of several such equation for different image points distortion coefficients are computed.

2.3 Toolkits for augmented reality overview

In last several years, a number of toolkits for AR have appeared. Generally, they can be divided into two groups:

- Toolkits based on marker detection
- Toolkits offering a marker-less detection

Although some previous solution were using markers for detection (available for example in such toolkits as AR Toolkit; its newer version AR Toolkit Plus; or ArUco that is based on OpenCV), in this work marker-less detection is required.

There are several toolkits with marker-less detection available. A short description of several of them is to find below.

2.3.1 OpenCV

OpenCV is an open source library for computer vision and machine learning originally created by Intel. Even though it is natively written in C++, it provides C, C++, Python and Java interfaces and supports Windows, Linux, Mac OS, Android, and iOS operating systems. This library is used by such corporations as Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota and has more than 47 thousand community members. In addition, it contains more than 2500 optimized algorithms that provide a wide functionality in the field of computer vision including [5], [27]:

- Marker-less detection
- Tracking of camera movement
- Pose estimation
- Camera calibration

Also, a full-featured CUDA interface for OpenCV is currently being developed.

2.3.2 ALVAR

ALVAR (A Library for Virtual and Augmented Reality) is a set of products and libraries for creating augmented reality and virtual application. It has been developed since year 2009 by the VTT Technical Research Centre of Finland under the GPL, therefore it is suitable for commercial use as well. Moreover, a commercial version of ALVAR — with advanced computer vision algorithms — and ALVAR Mobile SDK — for Android, Symbian, Maemo, Flash and Silverlight platforms — can be purchased.

ALVAR available under GPL is available for Windows and Linux operating systems and depends on OpenCV – OpenCV libraries have to be installed in order to use ALVAR. It incorporates functions:

- Marker-based tracking, pose detection, etc.
- Template-based tracking – matching against predefined images or objects; use of SfM and Fern's algorithms

- Camera calibration

A part of installation is also a comprehensive HTML manual, installation instructions, and FAQ sheet.

Information posted in this section can be found in [1].

2.3.3 Vuforia

Vuforia is an AR toolkit for creating applications for Android and iOS devices. It was developed by Qualcomm and offers e.g. marker-less detection and tracking, virtual buttons, a comprehensive online manual and growing community what makes, in combination with modern Android or iOS based smartphones, interesting choice for future use of augmented reality in maintenance.

More information on Vuforia can be found in [34].

2.4 Related work

The idea of using the augmented reality for training and maintenance is not new and has been applied several times by several companies and institutes. This section offers a brief overview of several of such applications.

2.4.1 BMW

BMW experimented with the AR in repair procedures – the goal was to guide a mechanic through an entire repair procedure as Figure 2.9 shows.

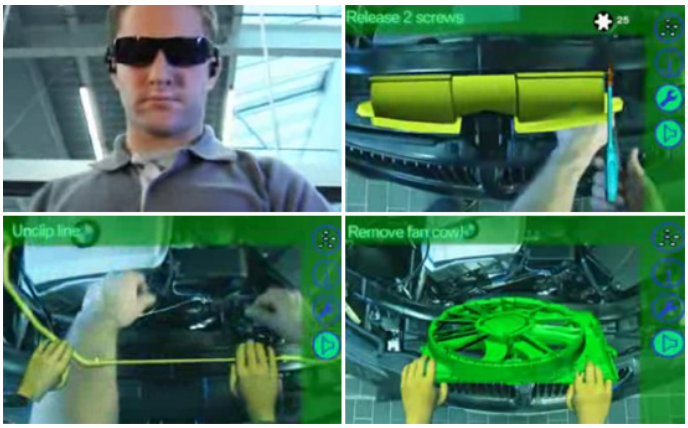


Figure 2.9: Augmented reality for maintenance by BMW [3].

Using the AR, the mechanic can see a 3D model of the part he is maintaining with additional text and audio instructions. Apart from that, he also sees an animation showing the tools and actions to be used in the current step of procedure. Also, several voice commands are present.

2.4.2 Boeing

Boeing has been attempting to include augmented reality in its assembling process since 1990's; the idea of its first project was, "by superimposing diagrams or text onto the surface of a work-piece and stabilizing it there on specific coordinates, an information about current step to be constantly being sent to a factory worker making his or her work more effective„ [26].

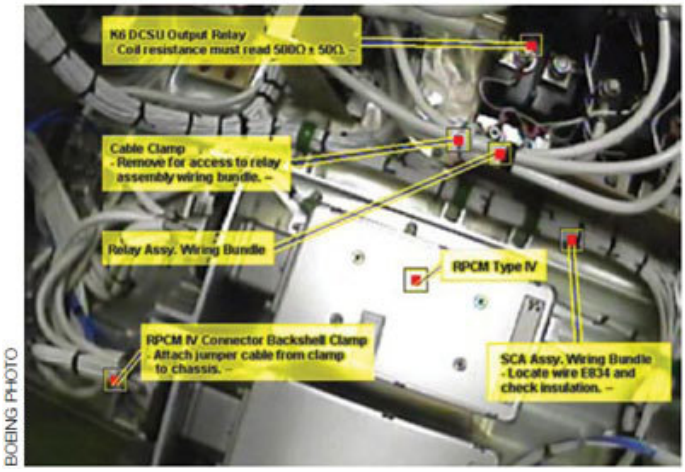


Figure 2.10: Augmented reality for maintenance in space by Boeing [24].

In its next project, Boeing wants to bring augmented reality to space training (as shown on Figure 2.10) in order to reduce the amount of time required for training the crew for maintenance. The AR set consists of a camera and a HMD [24].

2.4.3 Airbus

Airbus has experienced with the augmented reality as well. Its first prototype was using the augmented reality in installation of fresh water pipes – by using a standard laptop, a near-to-eye display (Sony Glasstron) and a web-camera, a technician was receiving step-by-step instructions and 2D drawings augmenting the real world.

The second prototype focused on a cable plugging support – this time, a small wearable computer (Xybernaut MA V) was used along with a mini camera (Toshiba IK-MX1) and a near-to-eye display (Tekgear M2). In this case, marker-based tracking was used and the real world was augmented by SVG[4].

However, Airbus has not deployed any of this prototypes in productive settings, mainly because of [35]:

- Missing user acceptance
- Maturity of near-to-eye displays was not high enough several years ago
- Difficult tracking conditions inside an aircraft – state of construction is always changing, geometry data is not complete (e.g. screws and isolation), bad lighting conditions (movable lamps and hard reflections) and many identical looking parts (e.g. stringers)

[4]Scalable Vector Graphics

2.4.4 Columbia University

At Columbia University, a prototype augmented reality application to support United States Marine Corps (USMC) mechanics conducting routine maintenance tasks inside a turret of an LAV-25A1 armored personnel carrier was designed, implemented and user-tested [13].

The entire turret volume is approximately 1 cubic meter, but since a mechanic has to fit into this space, the resulting work area is reduced to only approximately 0.34 cubic meters. In this space, a large amount of electrical, pneumatic, hydraulic, and mechanical infrastructure is located.

The prototype uses a VST HMD with two cameras (Point Grey Firefly MV) capturing the scene from the mechanic's point of view and a PC running Windows XP Pro. In addition, the mechanic uses a wrist-worn controller to navigate through tasks in a repair sequence. The mechanic's head position and movement is constantly being tracked by 10 tracking cameras placed inside the turret.

(a) (b) (c) (d)

Figure 2.11: Augmented reality for maintenance and repair by Columbia University [13].

Figure 2.11 shows the outcome of the prototype: when a maintenance or repair procedure is selected, first a screen-fixed arrow — that indicates the shortest rotation distance to target area — is shown (a); as the user orients on the target, a semi-transparent 3D arrow points to the target (b); the arrow is getting more transparent the closer the user orients on the target (c); finally, when the user locks on the target, the arrow fades out completely and a precise target location is highlighted [13].

This prototype was tested on several USMC mechanics that were given to complete 18 common tasks under field conditions. The results showed the augmented reality to be intuitive and satisfying for the tested sequence of tasks leading to reducing time required to complete a maintenance procedure. Also, when mechanics began physically manipulating objects in a task, they tended to not require information provided by the display. More about this prototype and results can be found in [13] and [12].

All results from the first prototype were processed and the prototype was further improved. A comprehensive description of these improvements coupled with a comparison between the prototype and a laptop-based documentation currently employed in practice with USMC mechanics can be found in [14].

Chapter 3

Introduction to airplane maintenance

Maintenance can be defined as a set of actions necessary to sustain or restore the integrity and performance of an airplane (in the case of this work), that includes inspection, overhaul, repair, preservation, and replacement efforts. In other words, maintenance is the process of ensuring that a system continually performs its intended function at its designed-in level of reliability and safety [19].

Basically, there are two types of maintenance – preventive and unscheduled maintenance.

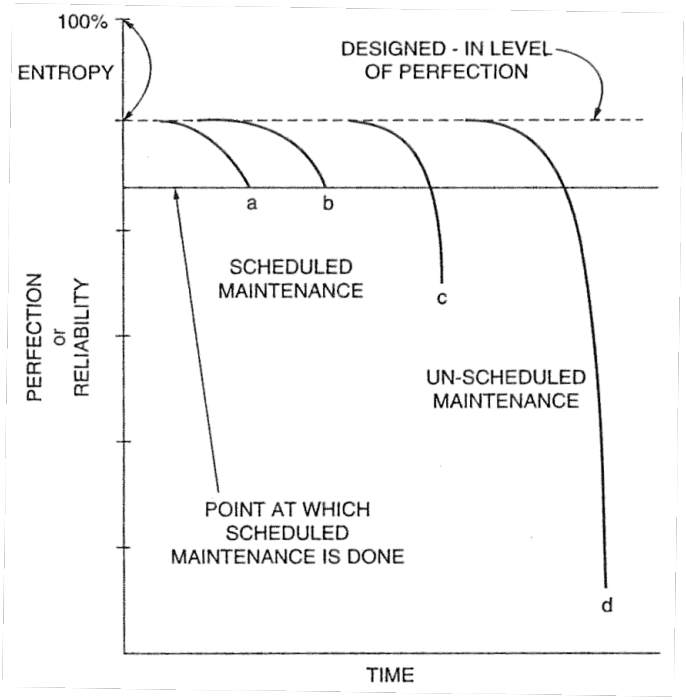


Figure 3.1: Scheduled and unscheduled maintenance [19].

- Preventive maintenance – a scheduled maintenance usually performed at regular intervals that is supposed to restore the system to its normal level and prevent its deterioration to an unusable level.

- Unscheduled maintenance – occurs, when the system breaks down completely; in such cases, the maintenance actions necessary to restore the system to its normal level are often more complicated; a troubleshooting, adjusting and also replacement of components may be necessary as well.

Figure 3.1 shows the effect of maintenance to reliability of the maintained unit or system; the entropy here represents the difference between an ideal and the real-world system.

Not all systems can be checked regularly to prevent them from failing – for example, a high-frequency radio used for communication has to be operated till a failure appears before maintenance can be done. Also because of this inability to maintain all airplane's systems, the aviation industry has developed three different concepts:

- Redundancy – is quite common in engineering design; if one component (called primary unit) fails, the other (called backup unit) is available to take over the function; because of that, the maintenance is sometimes performed only on a backup unit, since the primary unit is constantly in use and therefore it is known whether it is functional or not.
- Line replaceable unit (LRU) – a system that has been designed in a way that its parts that tend to fail the most quickly are easily replaceable; thus, no immediate maintenance is needed.
- Minimum requirement list – allows an airplane with some inoperative systems to stay in an active service and to legally defer the maintenance, if all items on this list are functional; this list is created by a manufacturer.

Although the scheduled maintenance can significantly prolong a service life of a component, at some point in time the component will deteriorate beyond an acceptable level or will fail completely. This results in an otherwise chaotic unscheduled maintenance. Because of this, various maintenance programs were developed. These programs specify the frequency and type of maintenance of various components, their testing, how a malfunction shall be handled, etc. A decision process that chooses the best action (e.g. repair, replacement, or report to further maintenance) for a given input is also included.

Sections 3.1 and 3.2 draw from [19], section 3.3 draws from [15].

3.1 Maintenance intervals

Not every system of the aircraft has to be checked with the same frequency. A set of standard maintenance intervals is defined:

- Transit checks – are performed after landing and before next takeoff, while the airplane is in transit at the airport; it consists of oil level check and fill actions and a general visual inspection that includes checking for any fluid leaks, opening or loosing panels and damage to the flight control surfaces or antennae. Any problem found results in an unscheduled maintenance.
- 48 - hour checks – usually replace 24 - hours (daily) checks. They include more detailed and complex tasks than transit checks, e.g. checking brakes and wheels, oil level for the auxiliary power unit and aircraft hydraulic fluid level.

- Hourly limit checks - are performed after a certain number of hours the unit or the system has been operating since the last check or maintenance; cover checks of such parts as engines, airplane flight controls and so on.
- Operating cycle limit checks – are similar to hourly limits checks, but instead of hours, operating cycles (number of flights) of an airplane are taken in account. Tires, brakes and landing gears are checked on operating cycle basis, since they are used only during takeoffs and landings.

3.2 Documentation

There are several types of documentation. Since their great number, only the most important are enlisted below. Most of these documents have standard revision cycles.

- Airplane maintenance manual (AMM) – contains all the basic information on operation and maintenance of the airplane and its on-board equipment. It describes components and functionality of each system and subsystem along with a description of basic maintenance and servicing actions such as removal and installation of LRUs, various operational tests, etc.
- Component and vendor manual – similar to AMM; it describes all components built by the airplane manufacturer or by outside vendors who supply electronics, computers and other systems that are installed on the airplane.
- Fault reporting manual – it is a set of flow diagrams that are designed to locate and isolate problems within aircraft's various systems.
- Structural repair manual – this manual provides its reader with information how to perform certain simple repairs of airplane structure.
- Task cards – these cards are produced by airplane manufacturer and are usually for one action only, e.g. step-by-step explanation: "open panel, turn certain equipment on and close the panel„.

3.2.1 Example of a service procedure

Because of the aim of this project, an example of a service procedure describing an installation of an engine mount [23] is shown here.

1. Move the engine mount into position on the firewall.
2. Install the 5 bolts which attach the engine mount to the engine nacelle firewall:
   - Push the bolts through the firewall and mounting pads.
   - Install the washers and self-locking nuts on the bolts.
3. Install the engine coolant radiator.
4. Install the inter-cooler mounting bracket.
5. Install the turbo charger inter-cooler.

6. Install the engine oil cooler.
7. Install the inter-cooler / oil-cooler shroud.
8. Install the engine air filter housing and alternate air valve.
9. Install the cabin heating heat-exchanger and shroud.
10. Install the propeller un-feathering accumulator.
11. Install all the clamps, clips and ties that hold the electrical harness and hose to the engine mount.

This is a typical example of a service procedure. It is a step-by-step manual with references to other procedures or manuals (not shown in the example above) that can be easily followed.

3.3 Basic description of L-410

For demonstration of the AR in maintenance, the Aircraft Industries' L-410 airplane (shown on Figure 3.2) was chosen. L-410 is a 19 seat twin-turboprop commuter aircraft that is mostly used in passenger transport. It is delivered with a complete set of technical and operational documentation.

Figure 3.2: Airplane Let L410 [15].

L-410's general characteristics are listed in Table 3.1:
The manufacturer of L-410 airplane is located in Kunovice, Czech Republic.

Crew	1 or 2 pilots
Capacity	17/19 passengers
Wing Span	19.98 m
Length	14.42 m
Height	5.83 m
Max. Take-off Weight	6600 kg
Max. Landing Weight	6400 kg
Max. Payload	1800 kg
Max. Cruise Speed	386 km/h
Maximum Range	1400 km

Table 3.1: L410 - general characteristics

Chapter 4

Previous work

This chapter describes the work done on this thesis within *Term Project* and the findings, suggestions, and solutions created during this phase of the project.

During this work, the theoretical background from previous chapters was used to get a first insight into the AR topic and to find out possible benefits and limitations of this technology in order to be able to determine a hardware and software configuration [1]. Also, Chapter 3 and majority of Chapter 2 of this thesis were created to process and categorize gained knowledge. Furthermore, based on this knowledge, a first simple prototype was built. Experience gained from this prototype was then used in an solution design in Section 7.1.

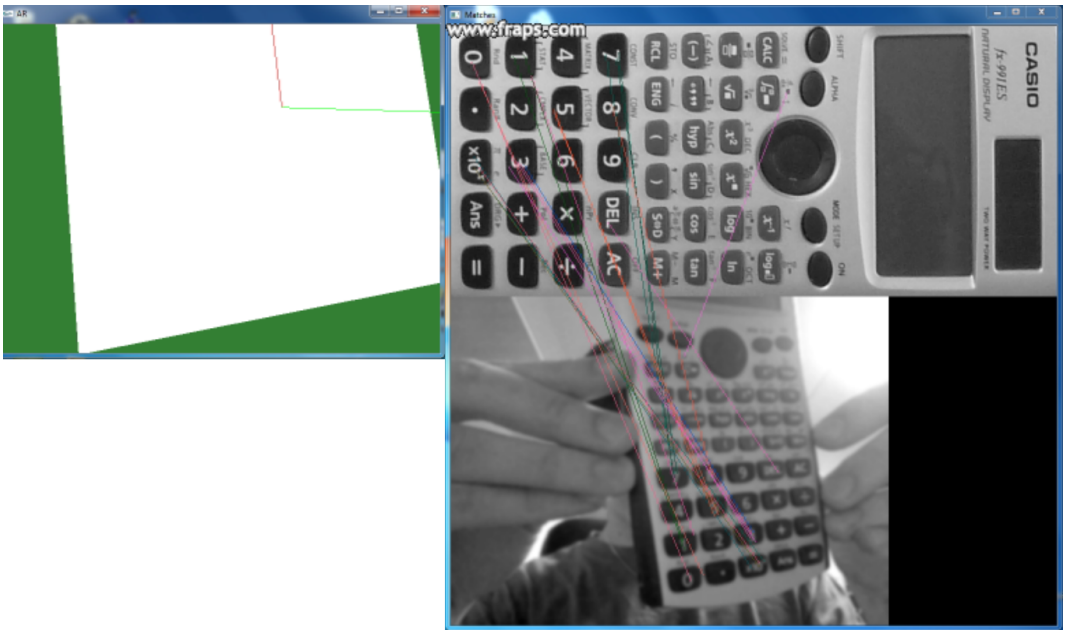


Figure 4.1: Matching the feature points on original object's image (upper right) with the same object found in the video (lower right) and showing its position rotation and translation according to the original image (left). Matched feature points are connected with lines.

[1] For more pertinent thesis structure, hardware and software configuration was moved to Chapter 5.

4.1 Prototype

To explore the options of object detection and tracking, a prototype has been constructed. This prototype uses a laptop's web-camera[2] to get and process video frames in real-time. Fern algorithm with feature points extraction had been used for tracking and pose estimation [1]. The prototype's algorithm has two phases:

1. *Training phase* – in this phase, one image of the object that will be tracked is required. The resolution of this image is recommended to be between 200×200 to 500×500 pixels and the training takes approximately 1 to 4 minutes. The output of the training is a datafile with information on found feature points.

2. *Tracking phase* – here, feature points are extracted from video frames and matched to the dataset. From the matches, a camera position and orientation according to the tracked object is calculated.

For a more user-friendly overview of tracking performance, a GLUT library was used for showing a plane located in a 3D space representing the position and rotation of the real-world object in comparison to its learned model (stored in the dataset). Additionally, matched points were connected by lines[3]. Figure 4.1 shows tracking of a testing object - a Casio calculator. The calculator was chosen due to unavailability of maintenance videos.

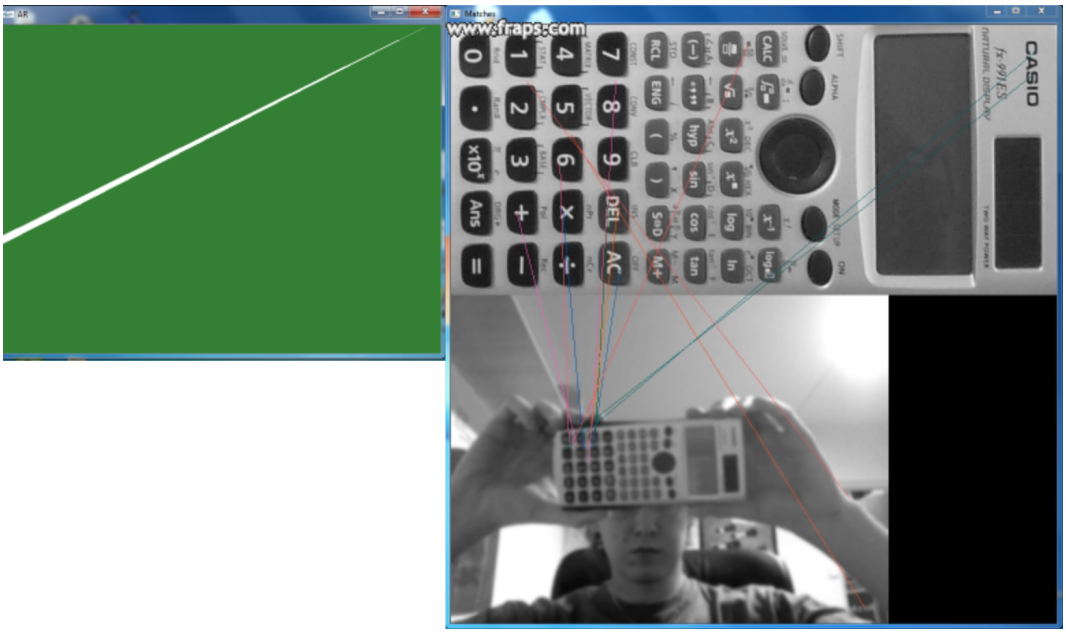


Figure 4.2: Tracking fails if the object is too distant.

However, during repetitive experiments, a problem occurred in this prototype; if the object was more distant, tracking failed, as Figure 4.2 demonstrates. This may have been caused due to the low quality of the used web-camera.

[2]It was the only available camera at that time.
[3]This option is provided by ALVAR.

Chapter 5

Hardware and software configuration

As presented in section 2.4, augmented reality in maintenance applications usually operates with a camera that captures a scene from a mechanic's point of view; a computer that detects and recognizes the scene and contains information about procedure; and a head-mounted display that imprints additional information of the current step of maintenance procedure on the video captured from the camera. This concept was chosen in this work as well.

A description of first two components — i.e. the camera and the computer — that were used follows below. However, it is important to mention that this project has a goal to explore the potential benefits of using the AR for supporting maintenance procedures, not to bring a production-ready implementation. Therefore, the software and hardware choices made for this project may not necessary reflect the needs of a commercial application.

5.1 Camera

Since camera is an essential component in AR for maintenance applications, the choosing of the camera, that was used during dataset acquisition, had received considerable attention. For this purpose, several point-of-view cameras were compared and evaluated before visiting LET. The most important properties (besides camera's dimensions and mounting capabilities) that were examined were:

- Video resolution and quality – critical for computer vision algorithms.
- Live stream capability – for enabling the connection between the camera and an OST HMD in the future[1].
- Wide field of view – the wider portion of the scene can be captured, the better orientation in it.

Finally, Contour+2 camera was chosen [6]. This camera comes with a headband, so it can be easily attached to technician's head. Because of a built-in microphone, voice commands can be transmitted along with video. Furthermore, it offers (via a 3rd party hardware) a wireless live video stream to a computer.

The full specifications can be found in Table 5.1.

[1] Honeywell has access to Lumus OE-32; therefore, it will probably be the first testing device.

Figure 5.1: Contour+2 – a camera chosen for this thesis.

Resolution	1920×1080, 1280×960, 1280×720, 854 x 480 pixels
Field of View	170°
Battery life	2 – 2.5 hours
Dimensions	$96 \times 34 \times 60$ mm
Internal microphone	Yes
Live stream	via HDMI or wirelessly via 3rd party hardware

Table 5.1: Camera properties

5.2 Computation

For this thesis, a Lenovo T500 laptop with Windows 7 SP1 and Visual Studio 2010 was used. The source code was developed in C++ programming language. ALVAR toolkit [1] had been used for the core functionality implementation because of high-level functions that allow to build a prototype rather easily. Also, OpenCV 2.4.0 [27] was used for handling the other functionality in the implemented solution because of its software maturity, comprehensive manual pages, and its active and vast community. Finally, TinyXML2 was used for XML documents parsing and Doxygen for generating documentation from the source code. Installation steps for installation of these libraries can be found in Appendix B.

Chapter 6

Data acquisition and analysis

Since this thesis deals with AR in the field of airplane maintenance, a suitable dataset — a set of video recordings of service procedures — had to be obtained. To accomplish this objective, a contact with LET — an aircraft manufacturer of aircraft L-410, located in Kunovice, Czech Republic — was established. After a mutual agreement, LET chose several procedures that were supposed to fulfill the requirements sent to them prior to the appointment that was held on 14 March 2013. In this chapter, a rough draft of these procedures is provided along with hardware used to obtain the dataset and a first dataset analysis.

Figure 6.1: Recorded procedure from the technician's point of view with Contour+2 camera.

6.1 Hardware equipment

For dataset acquisition, several devices were brought to LET:

- *Contour+2 camera with a headband mount* that was mounted on technician's head

with the intention of capturing video from his point of view.

- *Samsung Galaxy Note II smartphone* which was used for controlling the camera, changing its settings and capture modes, and receiving live video stream from the camera via Bluetooth.
- *Sony HDR FX1 HDV Camcorder* that was capturing service procedures from a different perspective for documentary purposes.
- *Nikon D5000 camera* for photo documentation.
- *Laptop* for backing-up recorded data directly after their capture and for problem solving.

In addition, LET provided lighting equipment for specific procedures performed in dimmer environment; its technicians; an L-410 aircraft; and service manuals with description of all steps within performed procedures.

Direct Bluetooth stream from the camera to the smartphone enabled to check the video output in real time[1]; therefore, it was possible to correct for some undesired faults (such as pointing the camera out of the actual maintenance area) and thus improve quality of captured video recordings. However, the chosen camera and headband mount put certain limitations on the video recording that were not fully eliminated. These limitations are defined in Section 6.3.

Figure 6.2: Contour+2 camera fixed on head using the headband mount.

[1] Although this stream was in low quality it still allowed for a check of camera's line of view

6.2 Procedures description

Preceding the visit of LET, a set of qualities to be met by the selected procedures was sent to LET. Among the requested qualities were:

- The procedure should consists of only several steps – the goal of this thesis is to test a prototype not to create a production-ready solution; in addition, less number of steps usually results in shorter procedures that are better for demonstration.
- During the procedure, the technician should have non-blocked view to all components he is manipulating with.

In return, LET responded with a list of four procedures that were supposed to fulfill these qualities. The selected procedures follow:

- *Lifting the aircraft*
- *Disassembling of the main landing gear and its break*
- *Disassembling of the fire alert unit DPS*
- *Assembling of the fire alert unit DPS*
- *Checking of the fire alert unit DPS*

6.3 Data analysis

All procedures were captured by Contour+2 camera mounted on the technician's head the same way as Figure 6.2 demonstrates. The video was captured in 1280×720 resolution at the rate of 30 frames per second. This resolution was chosen with the intention to keep the highest possible video quality while offering the widest field of view as close as possible to human's peripheral view. However, it has caused rather significant distortion.

Camera's headband mount allowed only for a limited and not very accurate positioning causing the camera was capturing scene slightly above the scene seen by the technician. In addition, the camera was attached from the left side of technician's head[2]. Since the technician was right-handed, the camera often captures the scene from a more acute angle as it would have done if the camera was on the technician's right side of the head. This was fixed during the execution of the last procedure — Checking of the fire alert unit DPS — so it offers a good opportunity to study the difference.

In the video recordings, the technician often turns his head to left or right, or moves through scene causing the manipulated component (or even the whole original scene) getting out of the camera's field of view even for several seconds. Due to camera's small sensor and high focal ratio (f-number) all rapid camera movements are blurry to a certain level.

Every service procedure was executed and recorded only once due to limited time.

[2]This was caused by assembling the camera mount by the author of this thesis, who is left-handed and who, at given time, was not aware of impact of such assembly to the output quality.

Chapter 7

Implementation

This chapter describes the process of final applications design and their implementation. Explanations of algorithms and approaches used during this process are to found here, including references to literature and previous chapters of this work. The output of the implementation — the source code — is sufficiently commented and Doxygen tool was used to generate a comprehensive HTML documentation[1] with cross-references; this files can be found on DVD.

For these reasons, there are no examples of the source code available in this chapter; its goal is to familiarize the reader with the ideas and reasons that led to use or creation of following algorithms and approaches, not with their specific implementation in a particular programming language.

7.1 Application design

Based on the results from the prototype and information from [12], [3], [14], and [35] an application design was created and implemented. In Section 6.3, many imperfections found in video recording were described; only a robust solution may be able to handle such imperfections and eliminate false or erroneous detections – or at least to decrease their occurrences. Such solution is presented below.

This solution has to be able to undistort and resize distorted video inputs, train detectors from given components' images, guide through individual procedures' steps, highlight a component which will be manipulated within the current step, identify an erroneous detection — which would result either into deformation of highlighting in the current step, or would cause a small undesirable position deviations from previously highlighted position[2] — and also to be as effective as possible.

It was discovered that some algorithms described later in this chapter will be required to run only a limited number of times. To reduce the number of parameters that it would be necessary to implement for an application which would work with all these algorithms, functionality was divided and a total of four applications were designed. Their brief description follows:

[1] All useful algorithms' parameters that cannot be set via applications' command arguments, can be set in header file `Constants.h`; these parameters are separated into several categories here in order to make the orientation in the file easier. It is recommended to study this file.

[2] These deviations had also occurred in the prototype causing the drawn plane was not stable but tended to slightly change its position and orientation.

- *Camera Calibrator* – calculates camera's intrinsic and distortion parameters from a set of calibration images.
- *Markerless Creator* – trains ALVAR's detector from an input image and stores the result on disk.
- *Frames Extractor* – a supportive application for the first two applications; it plays video and is able to export individual video frames as images and save them on disk; these images can be used for instance for camera calibration. The application is also able to pause video, if further investigation of the current video frame is needed before saving.
- *ARfTaEoAM* – an abbreviation of *Augmented Reality for Training and Execution of Airplane Maintenance*, the name of this thesis. It is the main application that implements the detection, validation and drawing algorithms described further in this chapter.

First three applications are only supportive and their execution may be required only once per camera and procedure[3]. Application parameters for each of these applications are located in Appendix C.

7.1.1 Procedure

One run of the main application works solely with one *procedure*. Due to the digital implementation of this procedure — which works with AR, detection and its validation, etc. — it has to differ from the typical definition of a procedure as presented in Section 3.2.1. Besides its name and textual (and possible image) description of individual steps, it should contain detector's data for each step of procedure as well.

Based on the lower quality of video dataset and experience with the prototype, it should be also possible to use two or more detectors (with different detection goals) simultaneously to reduce erroneous detections. Consequently, an arbiter unit would be required to validate the output of each detector and choose the best one.

To compensate for errors in detection even more, a history, that would keep information about a certain number of previous detection, should be implemented. From looking into the history, it would be possible to detect an anomaly from previous detections and correct it.

Sometimes the component, that is manipulated within the current step of procedure, is either too small or very difficult to detect. If it is firmly attached to another component or its position in relation to this component does not change, it would be easier if the detector detected this latter component and then calculated an offset to the former one. That's why an *Area of interest* (*AoI*, plural *AoIs*) was introduced which realizes this idea[4]. As one step may describe manipulation with more than one component (e.g. loosing several screws), it should be possible to work with more AoIs within one detector.

All this information should be collected in the electronic version of procedure that should be stored on a disk. Taking into consideration that a complex procedure may include many steps with various sets of detectors, all of them containing several AoIs for

[3]There are already created all necessary files for execution of the main application on the DVD attached to this thesis; therefore these application may not be executed at all.

[4]Basically, it just contains an offset between detected object and the searched object.

its each step, XML format was chosen for storing rather than plain text, as it offers good editing, processing and validation options. A sample of a procedure XML file is available in Appendix D.

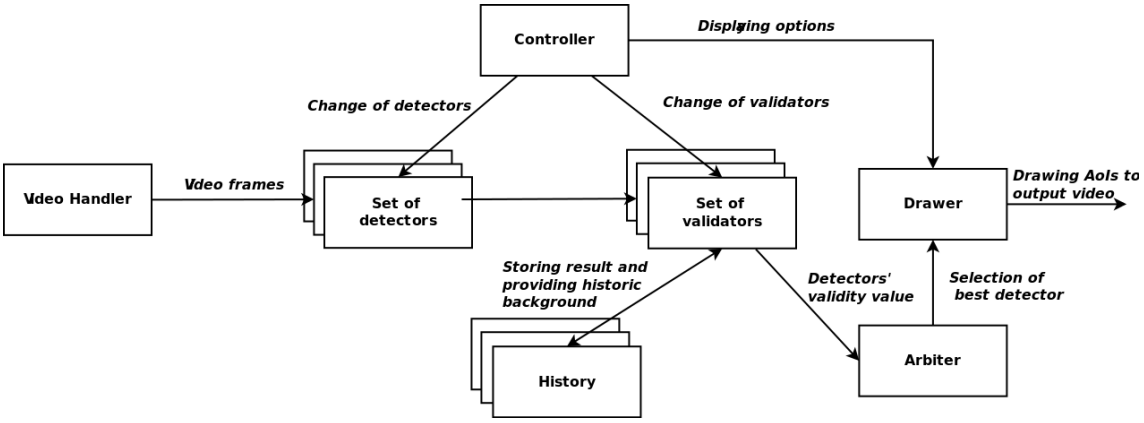


Figure 7.1: Main application design.

7.1.2 Final design

The ideas presented in this section were put together resulting in the design of the main application as illustrated on Figure 7.1. Here, the video input goes to a set of detectors, where each detects a different scene, and their output is validated with the use of history of previous detections. Arbiter then chooses the best detection which will be drawn. Controller switches among steps of procedure and detectors and validators associated with them.

Only a simple user interface was implemented in this work. This interface is based on a manual selection of procedure and also a manual switching among its steps. Even though a gesture recognition may seem as a very natural way of interacting with an AR environment, in conjunction with maintenance, where mechanic's hands are often in camera's field of view, their movements during a procedure can be misunderstood as a hand gesture. Nevertheless, other possible solutions do exist (for example the solutions explained in [36]) and will be explored in the continuation of this thesis.

7.2 Preprocessing

As mentioned in section 6.3, all video recordings were captured in high definition and with a high level of distortion. Due to a computational complexity of chosen detection algorithms and their sensitivity to distorted input, the recordings were undistorted and resized, respectively. These operations were executed on the original video recordings creating a new modified dataset that had replaced the original one henceforward.

The reason for this decision was that the distortion is individual to each camera and in some cases, undistortion is not necessary. Moreover, the recordings were captured in their high quality for possible future use; the most of video cameras offer an option to change their resolution. Therefore, the undistortion and resizing should not enter the main application's loop and affect its computational time. Naturally, these operations were implemented in a way that allows their use inside the main loop; however, they were not used there.

This section offers an insight into algorithms for camera calibration, undistortion and video resizing.

7.2.1 Distortion removal

Two versions of camera calibration and distortion removal — one using ALVAR and one using OpenCV library — were implemented due to several ALVAR's properties that did not meet former expectations:

- ALVAR creates calibration files that are used directly with object detection and their separate use is rather complicated.
- It offers only limited algorithm settings.
- For camera calibration, it uses old OpenCV structures[5] instead of classes available in the newer versions, while the rest of ALVAR's classes work with these newer OpenCV's classes, causing an inconsistency in source code.

The main calibration algorithm works with chessboards and remains the same for both version:

1. *Take an image and look for chessboard corners* – a number of vertical and horizontal inner corners of the chessboard has to be specified before executing this step.
2. *If specified amount of corners was not found, go to step 1.*
3. *Get subpixel accuracy on the corners* – this improves the calibration result.
4. *Add chessboard corners' coordinates to a list of detected corners.*
5. *If not enough chessboards were found and there are still images left unchecked, go to step 1.*
6. *Calculate camera's intrinsic matrix and distortion coefficients.*
7. *Save camera's intrinsic matrix and distortion coefficients to XML file(s)*[6].

The main difference is, the ALVAR implementation works with a video input where it looks for chessboards and if one is detected, it waits at least 1.5 seconds before attempting to detect the chessboard again – this allows the camera to change its position and orientation, expectantly increasing the diversity of chessboard views. It is also possible to set a maximum detection number, so the algorithm stops after detecting this number of chessboards and starts calibrating; because of this, even a long video record may be passed as input and its processing may not take too much time. In addition, every video frame with detected chessboard is saved on disk.

On the other hand, the OpenCV implementation does not detect chessboards from videos but rather from a list of images – this reduces the time required for the first two steps of the algorithm since a list of images with a good view of chessboard can be prepared in advance. When calibration is done, intrinsic matrix and distortion coefficients are used to calculate two maps (one for the x coordinate and one for the y coordinate) that map

[5] It uses C structure `IplImage` instead of C++ class `cv::Mat`.
[6] ALVAR uses only one XML file; OpenCV solution was implemented using two separate files.

each pixel in the distorted image to its new location. In the standard version, bilinear interpolation is used to calculate missing pixels and the output of the undistortion is cropped (as illustrated on Figure 7.2), so there are no blank areas with undefined[7] pixels. It is also possible to turn this cropping off[8].

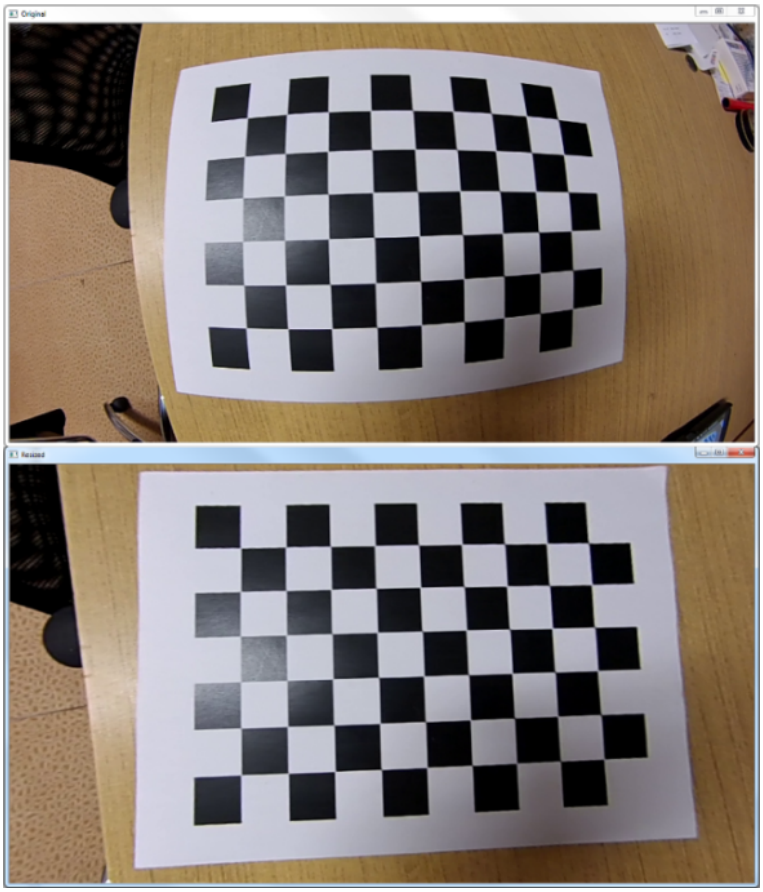


Figure 7.2: Application of distortion removal – original video frame (top image) is undistorted and area with undefined pixel values is cropped (bottom image)

.

It is recommended to provide or capture at least 10-20 chessboard images from different viewpoints at different depths [21]. Further in this work, only OpenCV implementation had been used. In addition, a method that undistorts a single frame or the whole video and saves it on disk was implemented as well.

7.2.2 Resizing

Resizing was implemented to work either with a single image or with the whole video recording. It uses interpolation for calculating new pixel values. Because this algorithm is used only for image shrinking, interpolation method was firmly set to area interpolation. For more information on this and the other interpolation methods, please see Section 2.2.3.

[7] in the case of OpenCV, set to black color.

[8] This can be done by changing value of constants `AR_CAMERACALIBRATION_CALCULATE_NEW_CAMERA_MATRIX` and `AR_CAMERACALIBRATION_ALPHA` in file *Constants.h*.

7.3 Detector training

ALVAR's FERN detector was used for detector training. ALVAR itself blurs the input grayscale image, so only strong feature points are extracted and random noise is reduced. Moreover, ALVAR also displays the input image with all found features during the training phase.

The algorithm for detector training consists of only three steps:

1. *Open image that will be used for detector training and convert it to grayscale.*
2. *Blur the image and extract its all feature points.*
3. *Save the result of learning in a* `.dat` *data file.*

The resolution of input images is recommended to be between 200×200 to 500×500 pixels and the training takes approximately 1 to 4 minutes [1]. The output of the training is a datafile with size of approximately 80 MB with stored information on found feature points.

It is possible to train more detectors in a sequence and choose their names and location. Appendices C.2 and F describe this functionality in more detail.

7.4 Procedure

Procedure implementation follows its design introduced in Section 7.1.1 and its XML representation can be found in Appendix D. For better understanding of the algorithms and approaches described in this chapter, it is important to get familiar with the structure of procedure. A brief insight into its organization follows.

7.4.1 Structure

Each procedure consists of its name, a set of steps and a set of detectors that are used with the procedure. Detectors were separated from steps due to a possibility of the same detector being used in several procedure steps. Since each detector stores approximately 80 MB of data, loading the same detector for every step again would not be effective; therefore, the detectors are stored separately and each step works only with their references.

Except of detector references, every step holds additional information on its unique ID, which defines its position in procedure[9]; a textual description of current step; a reference image, if available, describing a component in more detail; and additional information on all detectors associated with the step.

The additional information involves detector's ID[10] and a set of AoIs created for this detector. These AoIs are rectangles — each defined by their top left corner and their width and height — that envelope the area where a component is located. The use of rectangular offers many advantages which are further discussed in Section 7.6.

[9] After loading an XML successfully, all steps are sorted according to their ID.
[10] Detector's ID has the same purpose as step's ID.

7.4.2 Loading and Validation

A third party library — *TinyXML2* in version *1.0.11* [33] — was used for parsing an XML file. After the XML document is fully loaded and parsed, a check is performed. This check shows warning or error messages if some inconsistency or missing information is found in the procedure. Error messages cause the whole application to exit, warning messages allow it to run, however its functionality may be restricted. For a comprehensive information on this functionality, the reader is recommended to look into source code documentation generated by Doxygen.

7.5 Features detection and homography calculation

For features detection and homography (matrix) calculation, a combination of ALVAR and OpenCV is used. For features detection, ALVAR detector is used. This detector has to contain a `.dat` file[11] with a *model* — the object learned by the detector — which will be detected and works only with grayscale input images. Before the detection process begins, the input image is blurred – this behaviour is identical with the process of model's datafile creation described in Section 7.3.

The detector then extracts feature points from the input image and matches them to model feature points. The result of these operations are two feature points sets — set of model feature points M and set of image feature points I — for which the following applies:

$$\begin{aligned} &m_x \in M, i_x \in I, |M| = |I| \\ &\forall x \in \langle 1, |M| \rangle : m_x \to i_x \wedge i_x \to m_x \end{aligned} \tag{7.1}$$

Put in other words, a bijection exists between sets M and I, and each model feature point at position x is matched to image feature point at the same position. From these matches, homography matrix can be calculated.

However, this matches contain a number of mismatches or *outliers*. ALVAR uses its own internal algorithms for detection of such mismatches and computation of homography matrix.

This process can be summarized in several steps (assuming the model feature points were already successfully loaded):

1. *Extract image feature points.*
2. *Match these points to model points.*
3. *Remove outliers.*
4. *Calculate homography matrix.*

Another approach to the detection of outliers and calculation of homography matrix — based on solution from [21] and using OpenCV — was implemented in this thesis. Feature points detection remains the same as in the previous case, but a new outlier removal and homography matrix calculation were implemented.

In this implementation, image and model points are gained from the detector at the beginning. Then, OpenCV's implementation of RANSAC (RANdom SAmple Consensus)

[11] This file can be created by Markerless Creator application.

method is used for detection of outliers which are removed and finally, the homography matrix is computed using OpenCV's function implementing the technique described in Section 2.2.4. RANSAC is an iterative method that estimates parameters of a mathematical model (in this case homography calculation) by randomly selecting from a dataset that may contain a number of outliers. It means that the result given by this algorithm is correct only with a certain probability. More about this algorithm can be found in [21].

The homography calculation algorithm can be summed-up into these four steps:

1. *Get image and model feature points from detector.*
2. *Use RANSAC method for outliers detection.*
3. *Remove outliers.*
4. *Calculate homography matrix from reduced sets of image and model feature points.*

Unrelated to the chosen homography matrix calculation algorithm, the feature points used for its computing may be shown in output video. In the case of OpenCV implementation, also all detected feature points are shown.

7.6 Homography matrix validation

As mentioned in sections 4.1 and 6.3, there are segments in video recordings where the model is either not present or is indistinguishable from its background. These cause incorrect detections, since the homography matrix is computed every time at least eight feature points between model and image are matched[12]. Because of the nature of chosen features and video recordings rich on features, it can be assumed that more than 8 points are detected in each video frame even if the object does not occur there, resulting in an erroneous homography matrix.

The purpose of the algorithms (or tests) introduced in this section is to detect these erroneous matrices. Several different checks that use simple geometric objects — only AoIs and squares are used — were implemented. Because of use of simple two-dimensional objects and analytic geometry, these algorithms can classify the input matrix only with several basic arithmetic and trigonometric operations.

The idea behind these algorithms is simple – if homography matrix is not valid due to previously listed reasons, its application to an object will deform it noticeably. Such deformations can be divided into several categories with certain characteristics, which were discovered by observation of a number of deformations that occurred in the video dataset.

Each check searches for one typical characteristic of invalid matrix. If any test fails, the homography matrix is rejected as invalid.

Effect of these algorithms on erroneous matrices detection — both individual and combined — is tested and evaluated in Section 8.3.

7.6.1 Sides ratio check

A very simple test that uses a reference square. A perspective transformation using homography is applied to this square. Subsequently, the ratio of the longest and the shortest side of the transformed square is compared with a predefined constant.

[12] Although seven matches would be enough, OpenCV implementation uses eight [21].

This check eliminates matrices that would stretch model over a certain threshold where it can be assumed no valid detection should occur.

7.6.2 Complex quadrilateral check

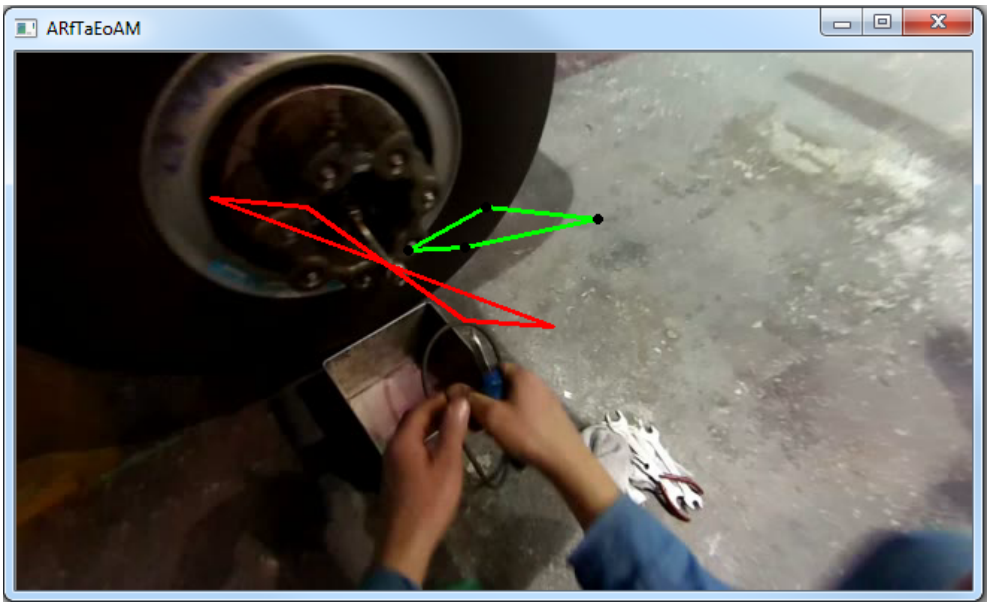


Figure 7.3: Dependency on AoI dimensions in quadrilateral check – green AoI reports that the current homography matrix passes the check while an red AoI is deformed and reports invalid homography.

In a number of invalid homography matrices cases, the transformed AoI became complex quadrilaterals [32], as two of their opposite sides crossed. This test analyzes if such case occurred.

Although AoI's dimensions affect this check's success rate, as Figure 7.3 illustrates, during tests, its performance was overcome by other checks, no matter what dimension were set to either AoIs or reference objects.

7.6.3 Angles check

In video dataset, a number of invalid homography matrices appeared that passed the first two checks; however their inner angles were too acute turning them basically into a line (or a line-like object) on the screen. To prevent this from happening, angles check was implemented. This test simply calculates all inner angles of a transformed AoIs using the law of cosines (a, b and c are lengths of triangle's sides and α, β and γ are their corresponding angles) [32]:

$$\begin{aligned} a^2 &= b^2 + c^2 - 2bc\cos\alpha \\ b^2 &= c^2 + c^2 - 2ac\cos\beta \\ c^2 &= a^2 + b^2 - 2ab\cos\gamma \end{aligned} \tag{7.2}$$

If any of calculated angles is more acute than a predefined constant, the homography matrix is rejected.

When all inner angles are known, another quick test that investigates, if transformed AoI remained convex or became a concave quadrilateral [32] is performed. An easy method to test it is to sum its inner angles – if the result is 360°, there is a good probability that the AoI remained convex.

7.7 History and AoIs' acceptation process

Even when detection succeeded and the detected object was enveloped by a transformed AoI properly, this envelop used to move or resize between two frames, what was rather disturbing. In order to reduce this resizing and movement, *history* was introduced.

History was implemented for each detector. It is a cyclic list of a predefined size, where records of previously calculated homography matrices, their validity, and transformed AoIs are stored. Its goal is to adjust AoI's coordinates or replace them completely (in case an invalid homography is detected).

This adjustment or replacement of a single AoI is done according to the following algorithm:

1. *If calculated homography was not valid, go to step* 4.
2. *Calculate weighted sum of distances between AoI's coordinate and all history entries and subtract it from AoI's original coordinate* (as equation 7.5 shows) *for each of AoI's eight coordinates*[13]. In this equation, c_{new} stands for new coordinate; c_{old} represents AoI's original coordinate; h defines index to the history, where $h = 1$ represents the most distant history; and w expresses weight. The more distant entry in history, the lower weight it carries.
3. *Calculate error according to equation* 7.6. *Go to step* 6.
4. *(Homography matrix was not valid) Replace AoI coordinate's value with* $center_{history}$ *calculated according to equation* 7.4.
5. *Set error to* 0.
6. *Return new AoI coordinates and calculated error.*

$$w = \begin{cases} 0, & homography\ for\ entry\ h\ is\ invalid \\ h, & homography\ for\ entry\ h\ is\ valid \end{cases} \tag{7.3}$$

$$center_{history} = \frac{\sum_{h=1}^{|H|}(c_h)w}{sum_{h=1}^{|H|}w} \tag{7.4}$$

$$c_{new} = c_{old} - \frac{\sum_{h=1}^{|H|}(c_{old} - c_h)w}{sum_{h=1}^{|H|}w} \tag{7.5}$$

$$error = \sqrt{\sum_{i=1}^{8}(c_{old_i} - center_{history_i})^2} \tag{7.6}$$

Then, the history algorithm can be described as:

[13] AoI has four corners, each corner is characterized by its x and y coordinate.

1. *Validate homography using checks from Section 7.6.*

2. *Use validity information from previous step to calculate new coordinates of each AoI as described in previous algorithm. Accumulate AoIs' errors.*

3. *Store all AoIs along with their homography matrix and validity information to the history.*

4. *Return an altered set of AoIs and accumulated error.*

The new set of AoIs is then used instead of the original one, when showed to the technician.

The accumulated error is calculated from a comparison to only those history entries, which were marked as valid. If the object was detected properly there will not be a significant difference from other valid detections; therefore, the accumulated error will be low. Hence, it can be assumed this error represents a certainty, that detected object corresponds to the object learned by the detector. Zero error means, no valid homography was calculated and AoIs' coordinated represents weighted and averaged history's coordinates. Otherwise, the higher the accumulated error, the higher the probability of a wrong detection.

7.8 Use of multiple detectors

As can be seen in Section 7.4 and Appendix D, it is possible to enter two or more detectors to one procedure's step. Because of this, the same object can be learned from multiple views[14] and these views can be searched for in each video frame during its processing. After all detectors are finished, result from each of them can be evaluated.

The evaluation is directly connected to the history; as mentioned in previous section, the history algorithm returns — except of an altered set of AoIs — also detector's accumulated error. This error is used when deciding which detector can provide the best results. If there are two or more detectors whose accumulated errors are equal and the lowest at the same time, either the one used in previous frame is chosen again or (if such detector is not among these detectors) the first detector in row is chosen. This detector's AoIs are then displayed.

The whole algorithm can be summarized in following steps:

1. *For each detector in the current step of procedure, try to detect the object it was trained for in the scene and calculate homography.*

2. *Validate each detector's homography and calculate detector's accumulated error.*

3. *Compare the error of each detector to error of the detector chosen in previous frame.*

4. *If this error is lower and its detector has valid homography, select it instead of the previous one.*

5. *Draw the selected detector's AoIs.*

Only a sequential version of this algorithm was implemented in order to explore its possibilities.

[14] This approach also requires a creation a different set of AoIs for each detector.

7.9 Displaying results

Detecting components, calculating their orientation and fixing detection errors would not be very meaningful, if no information was given to the technician. The easiest way how to pass this information is by augmenting the input video with a helpful content. At the same time, the technician should not be overwhelmed by amount of graphical information, as it might result in decreased productivity [12].

Based on tests and conclusions in [12] and [14], three different kinds of information passing were implemented. All information can be turned on/off, so it will not distract the technician when it is not required. Because it is not always possible to determine technician's position in relation to the searched component (for example when the technician turns or walks away from the aircraft or workbench to bring necessary equipment), a guiding line as shown on Figure 2.11 used in [13] was not possible to implement.

The following information passing techniques were implemented.

7.9.1 Highlighting the component

A very simple algorithm that uses OpenCV to connect four given points — that represent AoI's corners after application of homography matrix to its former corners' positions — with lines, creating a quadrilateral. Because of previous tests and validations, this quadrilateral is convex and should envelope the searched component.

7.9.2 Drawing text

This algorithm takes care about displaying a text bonded to the current step of procedure in the video. The text is horizontally centered and located at the bottom of the display where it does not block the technician's field of view.

7.9.3 Overlapping images

Sometimes the text and highlighting are not sufficient and an additional information — e.g. a technical image of a whole block of components — is needed as well. For this reason, it is possible to add an image with such data to every step of procedure.

This image overlaps the input video frame with its upper right corner fixed to frame's upper right corner. In order to prevent this image to cover a significant portion of video frame, maximum dimensions of overlapping images can be set; if the image exceeds any of them, it will be resized to fit into them while keeping its side ratio.

To further reduce the disturbance caused by displaying an overlapping image — because, when displayed, it blocks the a part of original video frame, no matter its dimensions — it is possible to set transparency α of this image. Then, output image o consisting of an image i overlapping a video frame f is calculated as:

$$o = i * \alpha + f * (1 - \alpha) \tag{7.7}$$

7.10 Main execution loop

The main loop follows its designed shown in Figure 7.1. Its arguments and bound control keys can be found in Appendix C.1. In addition, loading of video file, detectors, and parsing of XML file was implemented here.

Chapter 8

Reached results

The goal of this chapter is to discuss the results (in terms of efficiency, computational complexity, and contribution in general) gained by application of implemented algorithms to the video dataset, and to suggest possible future improvements that can be considered in the continuation of this work.

For testing purposes, four video recordings with four maintenance procedures were chosen:

1. *Video 1* – Disassembling of the main landing gear and its break.
2. *Video 2* – Disassembling of the fire alert unit DPS.
3. *Video 3* – Disassembling of the fire alert unit DPS and Checking of the fire alert unit DPS – second part.
4. *Video 4* – Assembling of the fire alert unit DPS.

These videos remained uncut (including technician's approach toward maintenance area) in order to provide the most authentic simulation of possible future use of AR in maintenance.

8.1 Preprocessing

This section examines gained results from undistortion and resize algorithms described in section 7.2.

The distortion level of captured video recordings was relatively high — as Figure 7.2 illustrates — and required undistortion.

Because of the ALVAR's disadvantages mentioned in section 7.2.1 (and due to incompatibility of its and OpenCV's intrinsic matrices and distortion coefficients format), camera calibration using ALVAR algorithms was not used for final calibration. However, feature in ALVAR's camera calibration implementation, which waits at least 1.5 seconds after the last chessboard detection, was used to create chessboard dataset – all images in the dataset contained chessboard detected by ALVAR, so there was a high probability that they will be detected by OpenCV as well.

This dataset was then passed to OpenCV which calculated and stored intrinsic matrix and distortion coefficients. Figure 7.2 shows the successful result of undistortion. It also shows the result was cropped.

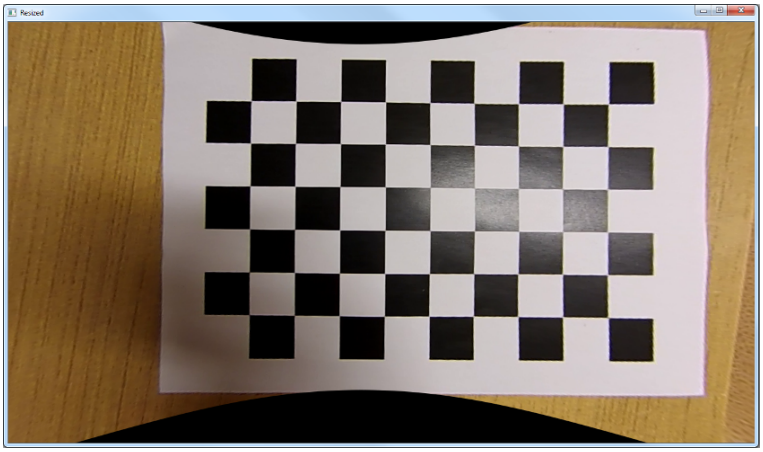


Figure 8.1: Distortion removal with calculating new camera intrinsic matrix with parameter $\alpha = 0.3$; areas with undefined pixel values appear in the output image.

The OpenCV implementation[1] offers a way of reducing or eliminating this cropping by calculating a new camera matrix based on a free scaling parameter $\alpha, \alpha \in \langle 0, 1 \rangle$, where $\alpha = 0$ means that all the pixels in undistorted image are valid (no pixel with an undefined value is present); $\alpha = 1$ means, that all the original image pixels are kept in the undistorted image. Result of crop reduction (by using $\alpha = 0.3$) is shown on Figure 8.1.

However, after output analysis, it was decided to keep the original cropped result without any undefined pixels. The main reason was that the curve between undefined (black) and valid pixels would create a new set of feature points which would decrease probability of a successful detection. Furthermore, although the camera was not aligned perfectly with the technician's line of sight, manipulated objects usually remain in the center of video recordings; therefore, it is more beneficial to keep cropped result with only valid pixels.

First, camera calibration and video undistortion were performed, and then undistorted video was resized to resolution 640×360 what was a compromise between quality and time required to process it. The results of this step were used further in this thesis.

8.2 Basic detection

Various undistorted and resized video inputs were passed to a Fern detector. These inputs contained both scenes with the object the detector was trained to detect, and scenes without this object. Both ALVAR's and OpenCV's implementations were tested and times required for processing of a single frame were recorded.

To get more accurate results, the main application was launched several times for all four videos in order to reduce deviations caused by other processes running on the operating system. Table 8.1 shows measured times.

Finding feature points and pairing them with the feature points in detector's model takes approximately 24 ms. Every detection and pairing returned in average 102 matched feature points[2]. Since there is no description of this functionality in ALVAR's documentation,

[1] OpenCV and ALVAR implementation in this text means the author's implementation that uses OpenCV or ALVAR, respectively.

[2] This was tested using several different detector's models.

Operation	Processing time (ms)
Feature points detection and matching	24.13 ms
Detection, matching and homography calculation (ALVAR)	122.44 ms
Detection, matching and homography calculation (OpenCV)	110.93 ms
Processing one video frame (ALVAR)	142.37 ms
Processing one video frame (OpenCV)	133.26 ms

Table 8.1: Average times.

and the number of matched feature points differed slightly between two frames (even when the same detector's model was used), it was only assumed that the detection and pairing returns only the strongest matches.

As soon as the homography calculation was added, processing time changed to 122 ms for ALVAR implementation and 111 ms for OpenCV implementation that used RANSAC algorithm for homography calculations. This resulted in frame's average processing time[3] of 142 ms for ALVAR implementation and 133 ms for OpenCV implementation.

Although the OpenCV implementation was slightly faster while providing similar results, both versions provided processing speed of only about 7 to 8 frames per second.

8.3 Homography matrix validation

Even though the processing speed of detection and homography calculation was noticeably low, the results provided by it were often invalid; this was caused either by the object (represented by detector's learned model) was not present in a frame, the frame was blurred due to camera's movement, or object's orientation (mostly rotation) was too different from the model's to allow a successful detection.

A possible solution of detecting these invalid results was implemented in Section 7.6. Effectiveness of the algorithms introduced there was tested and results are shown and discussed below.

The tests were based on detecting an object which either does not occur in input video or is undetectable (e.g. due to its small dimensions, orientation or blending with environment). In this situation, calculated homography usually deforms given objects; therefore, the algorithms should mark these homographies as invalid.

For this test, a simple procedure[4] was created, which contained a learned model of wheel from disassembling of the main landing gear and its break procedure (see Figure 8.2 – wheel image is located in the overlapping window during all three tests). This wheel was not detectable in any of test videos (which were already undistorted and resized) and contained two AoIs:

1. `<aoi x="0" y="0" width="246" height="211"/>` with dimensions of the learned wheel image; and
2. `<aoi x="50" y="50" width="100" height="100"/>` located inside the first AoI.

[3] This value also contains loading the frame, converting it to grayscale, drawing information and displaying it, and handling keyboard inputs.

[4] With one step and one detector.

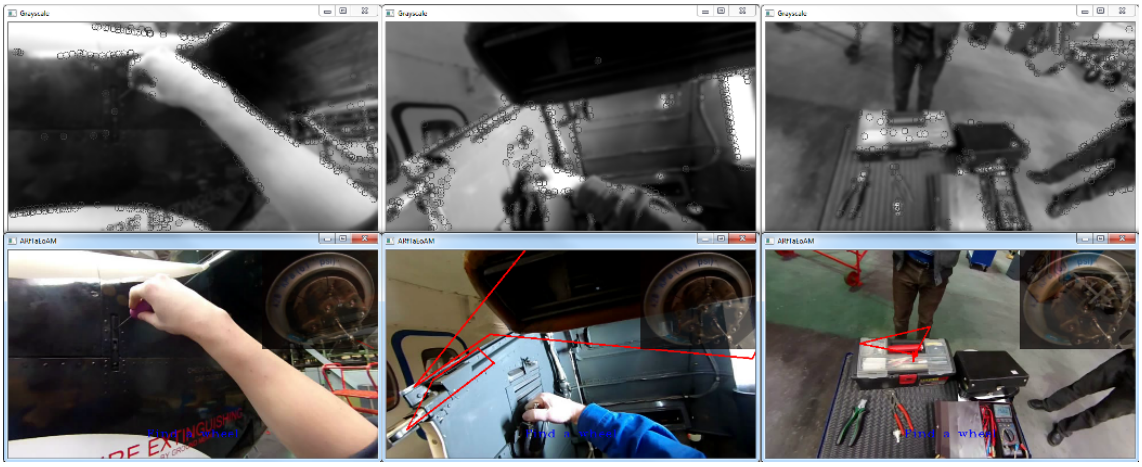


Figure 8.2: Detecting an object (wheel from Video 1) that does not occur in any scene (from left to right – Video 2, Video 3 and Video 4).

No other algorithms that would affect results of detection were used. The tests were executed with following parameters:

- Sides ratio check – if the ratio of the longest and shortest side of transformed reference square is greater than 5, the homography matrix is marked as invalid.
- Complex quadrilateral check – no parameters are required for this test.
- Angles check – rejects all matrices causing that transformed AoIs have one or more angles more acute that 5°, or whose sum of inner angles is not equal $360° \pm 0.01°$[5].

These parameters were set to such values that any AoI or reference square meeting these parameters can be considered invalid under most circumstances; this results in considering the homography matrix, that created it, invalid as well.

Algorithm name	**Abbreviation**
Sides ratio check 7.6.1	S
Complex quadrilateral check 7.6.2	C
Angles check 7.6.3	A

Table 8.2: Abbreviations of individual homography matrix validation checks.

Three tests (each with one video recording)were executed. Table 8.3 shows results of each test. V_x in this table stands for a number of matrices marked as valid in video x, I_x represents a number of matrices marked as invalid and % is check's success rate.

The tests results in this table show, that despite their simplicity, these checks offer good results by filtering over 99 % of invalid matrices when combined. It also shows that the Angles check is able to correctly classify homography in most of situations. On the other hand, Complex quadrilateral check does not contribute to success rate when combined with

[5] A rather vast space for rounding errors was given.

Video name	V_1	I_1	**%**	V_2	I_2	**%**	$\mathbf{V}_3$	$\mathbf{I}_3$	**%**
S	925	1635	63.9 %	918	1702	65.0 %	2425	4122	63.0 %
C	904	1656	64.7 %	969	1651	63.0 %	2332	4215	64.4 %
A	24	2536	99.1 %	14	2606	99.5 %	43	6504	99.3 %
$S + C$	316	2244	87.7 %	327	2293	87.5 %	794	5753	87.9 %
$S + A$	23	2537	99.1 %	13	2607	99.5 %	30	6517	99.5 %
$C + A$	24	2536	99.1 %	14	2606	99.5 %	43	6504	99.3 %
$S + C + A$	23	2537	99.1 %	13	2607	99.5 %	30	6517	99.5 %

Table 8.3: Number of homography matrices classified as Valid (incorrect classification, as all matrices should be classified as Invalid), Invalid (correct classification), and each check's (or combination of checks') success rate.

other checks[6]. However, it was not removed from the checks since there still exist theoretical situations where it can detect an invalid homography even if the other two checks fail.

Although some erroneous classifications were made, applying history further reduced this number.

8.4 Application of history

The application of the history was a continuation of the effort to further reduce the number of invalid detections and stabilize the appearance of the valid ones.

To test the effectiveness of the history, Video 1 was chosen and history size was set to 25 detections, allowing to remember 25 homography matrices along with their validity statuses and sets of AoIs.

The larger the history, the greater the resistance to dynamic changes. Setting history size to 25 detections was a result of performing several experiments. Since the input video has the frame rate of 30 fps, less than a second of detections is being remembered. This history size should be short enough to prevent from distractions caused by a dynamic change of scene where the AoI still remains drawn, and sufficient for reducing small deviance in detections as Figure 8.3 illustrates.

In this figure, AoI's coordinates are calculated from a combination of its current and history's average coordinates and are drawn in blue color. The order of frames is from left to right, from top to bottom. Between every pair of frames shown in this figure, three another frames passed. In order to show a wider fragment of video, these frames were skipped.

It can be seen, although AoI's original coordinates (green quadrilaterals) show detected object under different positions, because of the history the AoI's calculated coordinates change only a little. When an invalid homography matrix appears (red quadrilateral), AoI's coordinates remain the same as in the previous detection.

Even though the output of the main application improved significantly when history was applied, using every single valid history entry for calculating AoI's new coordinates is not ideal; a set of AoI with significantly larger or smaller dimensions which homography matrices were erroneously marked as valid, can affect the calculation of several following AoIs' coordinates noticeably. To reduce this problem, a new algorithm is suggested to be implemented in the continuation of this work.

[6]To confirm this, each test with the Complex quadrilateral check was performed twice.

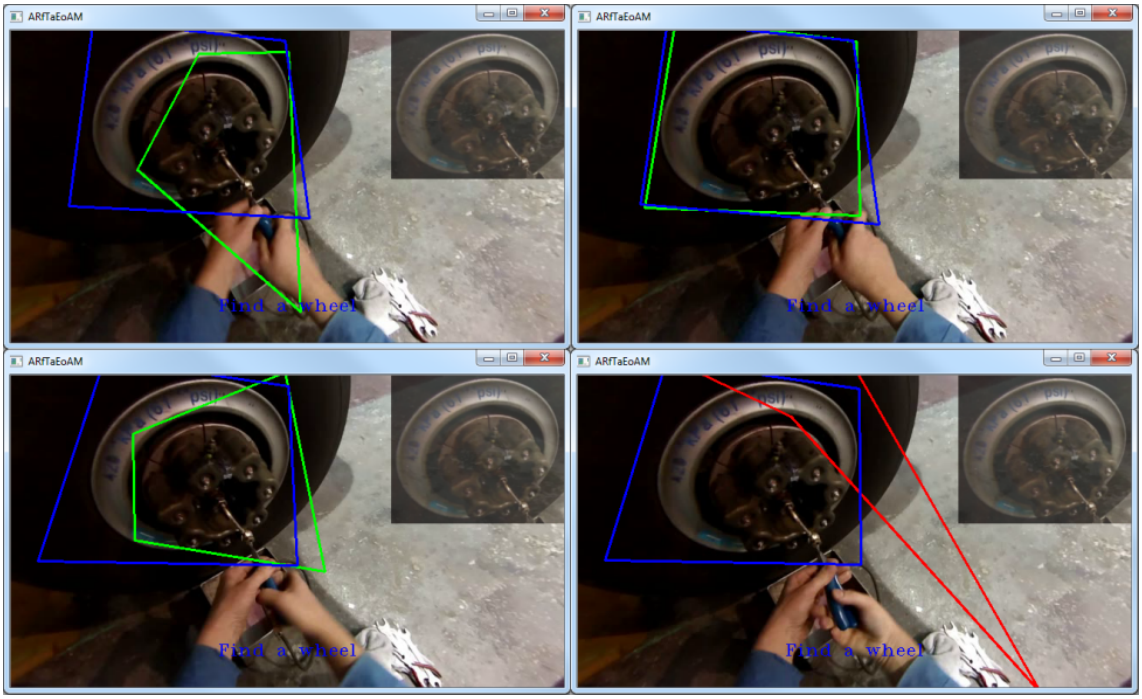


Figure 8.3: Application of history; the coordinates of AoI shown to the technician (blue) are computed from the history rather than using original AoI's coordinates (green and red).

This algorithm uses a clustering method based on density — e.g. *DBSCAN – Density Based Spatial Clustering of Applications with Noise* [8] — to find clusters of AoIs (each AoI is represented here as a point in 8-dimensional space[7]) and use only one of these clusters to adjust or replace AoI's coordinates. The idea behind this algorithm is that, when an object is detected properly in a scene, all homographies tend to locate the AoI into the similar area (in comparison to an area consisting of the cases when the detection fails) what creates a dense cluster in 8-dimensional space. This allows to separate outliers and achieve better results than with an algorithm implemented in this work. The proposed algorithm's description follows:

1. *Validate AoI's homography matrix.*
2. *If matrix is not valid, find cluster with the highest density and calculate its center using modified equation 7.4, where only entries within this cluster enter the equation. Return AoI with replaced coordinates.*
3. *Else insert AoI to history.*
4. *Find the cluster with highest density, calculate its center using modified equation 7.5, where only entries within this cluster enter the equation. Return AoI with adjusted coordinates.*

In this algorithm, AoI with homography marked as valid may affect which cluster will be chosen, since it is added before the cluster selection. The history size should be set

[7] Again, an AoI is a concave quadrilateral defined by four corners with two coordinates resulting in eight coordinates in total.

to $size + 1$ to compensate for adding the AoI from current detection before using history. Additionally, a minimal density can be set causing no AoI will be shown unless there is enough evidence the detection is correct.

8.5 Detection using multiple detectors

Because of its serial implementation, using multiple detectors resulted in processing of only 3 to 4 images per second. However, the switching between detectors and choosing the one with better evaluation worked correctly. In the continuation of this work, it is recommended to implement parallel version of this algorithm — for example by using OpenMP that can be added to this project easily [29] — and run it on a multicore architecture, what may result in significantly improved results when multiple different detectors will be running simultaneously.

8.6 Testing on video dataset

The final application was tested on dataset acquired in LET and its output was evaluated. Because of the video preprocessing and implementation of the homography validation together with the history, a majority of erroneous detections was removed and the drawing of AoIs stabilized (when the valid rear-world representation of the learned object was detected). If video segment was stable and detected object was properly visible, the detection output was similar to the one shown in Figure 8.4.

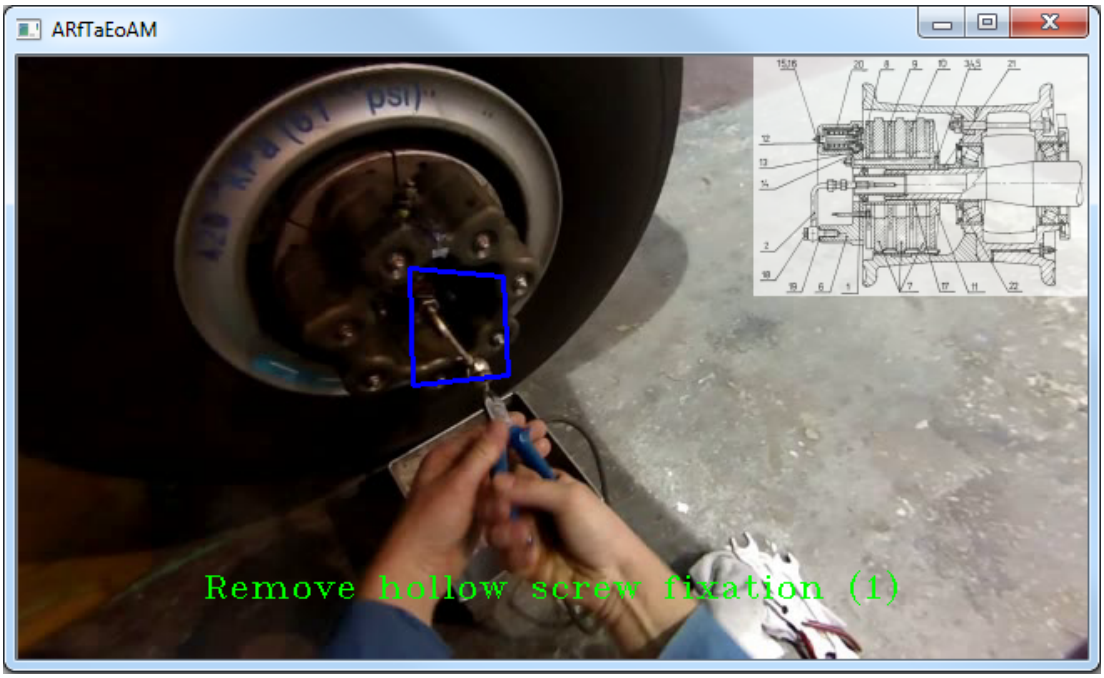


Figure 8.4: An example of a valid application output – a component that is being manipulated with during the current step of the procedure is highlighted, and procedure's text is shown to the technician with a reference to a semi-transparent schematics shown in the upper-right corner.

However, this was rather an occasional situation. The acquired video dataset contained a number of flaws caused by the acquisition process and the camera's properties, for instance:

- Because of the way the camera was mounted, it does not always capture the whole scene.
- The technician often moves his head or himself (this activity was often required by the procedure); because of the camera's larger f-number, all movements blurred the whole scene.
- The object that should be detected is often either out of camera's field of view or is being captured from too acute angle to allow a successful detection.

Because of this flaws, often no — or less often erroneous — detection appeared as shown in Figure 8.5. In addition, nor ALVAR neither OpenCV implementation offered a satisfactory processing speed to allow a smooth running of the application.

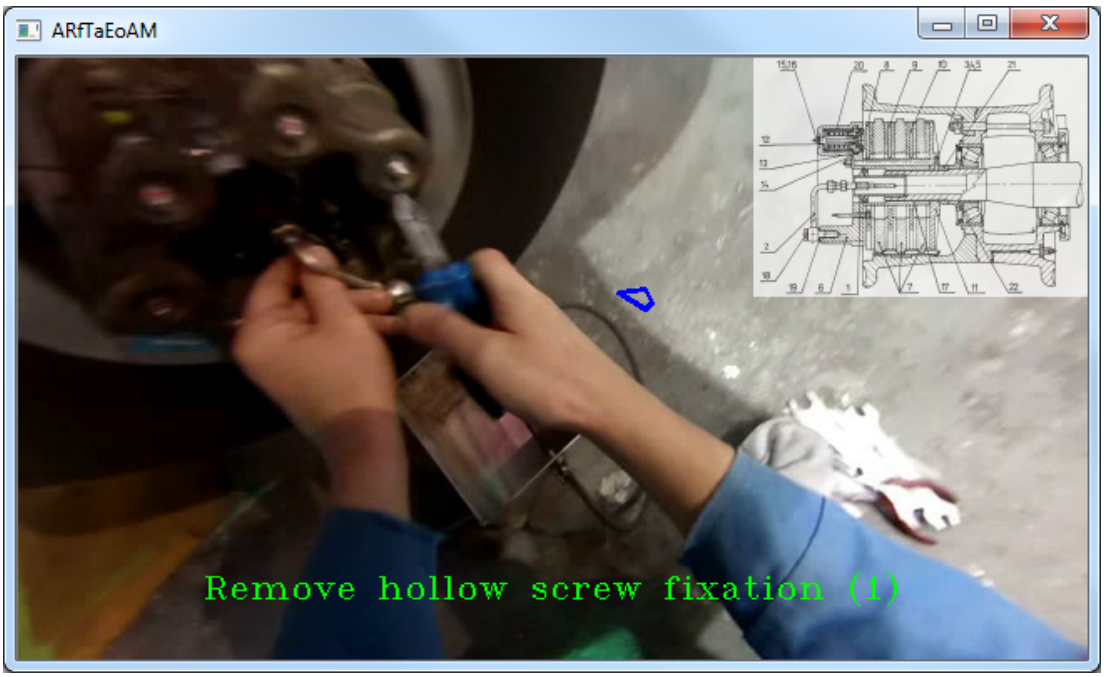


Figure 8.5: Technician's change of position resulted in the whole scene being blurry causing an erroneous detection.

Due to all these difficulties, a successful use of the AR during the whole maintenance procedure was not accomplished in this thesis. A possible solution could be reached by a use of several detectors — each with a model of the component taken from a different view — working together, as suggested in previous section.

Nevertheless, the created solution offers a good starting point to the future continuation of the project. It was being developed with the goal to continue it in the future (still in a cooperation with Honeywell). The structure of the application was designed in order to allow change of basic components (such as ALVAR's detector or individual history's algorithms) rather easily. If ALVAR's detector is switched to other solution, the suggested changes and improvements are implemented, and a new dataset (by which acquisition the

experience gained from previous dataset manipulation can be utilized) is created, the output quality of the application would increase significantly.

Chapter 9

Conclusion and future directions

In this work, a basic overview on technologies used with the augmented reality was given, providing the reader with an insight in displays, algorithms, toolkits and research done in this area. In addition, several examples of usage of augmented reality in maintenance applications were listed, including experiments of well-known companies such as BMW, Boeing or Airbus. To achieve that, a number of relevant research papers, presentations and experiments had to be studied.

Also, because this work operates with the aircraft maintenance, this topic was widely examined and the information required for its understanding was included in here. Based on the other already existing implementations, a hardware configuration was determined for the continuation of this project.

Next, a prototype application was created in order to better understand the possibilities and limitations of the AR. This prototype pointed out several issues such as the need of a better camera and possible problems with more distant or smaller objects.

The experience gained from this prototype and studied literature led to a creation of algorithms that should remove limitations found on the prototype, and to design and to implementation of the final application.

This application was created with an intention to continue in its development after the finish of this work, since it was developed for Honeywell, which would like to bring a working solution for application of the augmented reality in maintenance within the next couple of years.

A communication with LET Industries — a manufacturer of LET-L410 aircraft — was established and a dataset was acquired in LET hangars. This dataset consisted of several maintenance procedures captured by a camera mounted on technician's head while performing these procedures.

However, this dataset proved unsatisfactory, as it contained a number of flaws – starting with a slightly different angle of view of the technician and the camera, and ending in blurry frames caused by rapid motion of technician's head.

Nevertheless, additional algorithms were introduced in order to increase detection success rate on the dataset, including distortion removal. Among the other algorithms, there were several simple tests that were able to eliminate a vast number of invalid detections. In addition, a history was established, which further improved the detection success rate and reduced small deviations in detecting the same object between two video frames. Additional improvement of history using clustering method based on density was suggested and its basic algorithm was provided.

Furthermore, an electronic implementation of the maintenance procedure was proposed

and realized. This procedure is represented by an XML file that contains all information necessary for a successful run of the application.

A usage of multiple detectors for detecting the same object in the scene from different viewpoints was proposed and implemented. However, due to the computational complexity and the sequential character of this implementation, it was not fully utilized yet.

All implemented algorithms were tested and test results were shown and discussed in this work. Although an effort to use this solution to guide a technician through a complete maintenance procedure was not successful, these tests provided new information and brought proposals of new algorithms and approaches that will be used in the continuation of this project.

The continuation of this project is again planned in cooperation with Honeywell, which has already started several initiatives that are aimed to computer-based maintenance; this represents a complex solution which includes maintenance scheduling, components monitoring, failure localization and identification, and so on; it also plans to work with the augmented reality.

Therefore, in the future continuation of this project, experience gained during working on this thesis will be put into several different areas described below.

It will be important to acquire a new high-quality data set, probably in a more controlled environment. Next, ALVAR library proved quite unsatisfactory; although it offered high-level functions for an easy creation of augmented reality applications, its options were limited and its documentation rather poor. For this reason, it is recommended to use OpenCV or other well-documented tool with a wide community. This will be quite simple since the implemented application allows to change its detector quite easily.

Parallel processing will be also an important part of future work, since it can increase the output quality of the application significantly.

Also a new user interface will be required; the UI used within this project is quite simple and aimed more to testing. For this purpose, instead of hand gestures recognition a voice commands recognition should be considered, as the hands movements during a procedure execution might be confused with hand gestures. On the other hand, voice commands — especially discrete words recognition, which can recognize several pre-trained commands also in a noisy environment, such as aircraft hangars — offer a comfortable way of controlling of an augmented reality application since the technician can use his or her hands to manipulate with other objects.

Finally, during the future work on this project, it will be important to discuss regularly the user interface and all the potential benefits, drawbacks, and possible improvements of the whole application with technicians who will test this application, so the application will fulfill their expectations and its purpose.

The augmented reality for training and maintenance holds a great potential not only in saving costs for training technicians and reducing amount of printed manuals, but also in enabling an easier way of quality control by recording and analyzing performed procedures and reducing the administration associated with requesting, planning and execution of the maintenance procedures.

The absence of the augmented reality in the maintenance in general is an indication there are still many problems that need to be solved and challenges to be overcome. I believe this work and its continuation will help to address some of them and will bring some new ideas and approaches to this not yet fully discovered field of computer vision.

Bibliography

[1] ALVAR. Alvar: Introduction [online]. http://virtual.vtt.fi/virtual/proj2/multimedia/alvar/index.html, 2012 [cit. 2013-01-02].

[2] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47, 2001.

[3] BMW. BMW Augmented Reality in practice [online]. http://www.bmw.com/com/en/owners/service/augmented_reality_workshop_1.html, 2007 [cit. 2013-01-03].

[4] J. Bouguet. Camera Calibration Toolbox for Matlab – First calibration example - Corner extraction, calibration, additional tools [online]. http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html, 2010 [cit. 2013-05-13].

[5] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 1st edition edition, 2008. ISBN 978-0-596-51613-0.

[6] Contour. Contour+ 2: Tech specs [online]. http://store.contour.com/ae/us/cameras/contour+-2/invt/1700/, 2012 [cit. 2013-01-03].

[7] K. Dorfmueller-Ulhaas and D. Schmalstieg. Finger tracking for interaction in augmented environments. *ISAR 2001*, pages 55–64, 2001.

[8] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[9] P. Fua and V Lepetit. Vision based 3d tracking and pose estimation for mixed reality. *Emerging Technologies of Augmented Reality Interfaces and Design*, pages 43–63, 2007.

[10] Google. Google Glass – Tech specs [online]. https://support.google.com/glass/answer/3064128?hl=en, 2013 [cit. 2013-05-12].

[11] Google. Google Glass – What It Does [online]. http://www.google.com/glass/start/what-it-does/, 2013 [cit. 2013-05-12].

[12] S. Henderson and S. Feiner. Augmented reality for maintenance and repair (armar). Technical Report AFRL-RH-WP-TR-2007-0112, Department of Computer Science, Columbia University, 2007.

[13] S. Henderson and S. Feiner. Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret. *Proceedings of IEEE International Symposium on Mixed and Augmented Reality*, pages 135–144, 2009.

[14] S. Henderson and S. Feiner. Exploring the benefits of augmented reality documentation for maintenance and repair. *IEEE Transactions on Visualization and Computer Graphics*, 17(10):1355–1368, 2011.

[15] Let Aircraft Industries. Aircraft l 410 uvp-e20 [online], 2012 [cit. 2012-12-30].

[16] H. Ishii and B. Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. *ACM CHI*, pages 234–241, 1997.

[17] P. Jackson, J. Ealey-Sawyer, I. Lu, and S. Jones. Testing information delivery methods using augmented reality. *IEEE and ACM International Symposium on Augmented Reality*, pages 171–172, 2001.

[18] H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto1, and K. Tachibana. Virtual object manipulation on a table-top ar environment. *ISAR 2000*, pages 111–119, 2000.

[19] H. Kinnison. *Aviation Maintenance Management*. The McGraw-Hill Companies, 1stnd edition edition, 2004. ISBN 0-07-142251-X.

[20] K. Kiyokawa, Y. Kurata, and H. Ohno. An optical see-through display for mutual occlusion of real and virtual environments. *ISAR 2000*, pages 60–67, 2000.

[21] R. Laganière. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, 1st edition edition, 2008. ISBN 978-0-596-51613-0.

[22] Lumus. Lumus OE-32 [online]. `http://www.lumus-optical.com/index.php?option=com_content&task=view&id=9&Itemid=15`, 2012 [cit. 2013-01-03].

[23] Maintenance Manual. Airplane maintenance manual, 2004.

[24] E. Memi. Boeing's working on augmented reality, which could change space training. *BOEING FRONTIERS*, 2006.

[25] P. Milgram and F. Kishino. A taxonomy of mixed reality visual display. *IEICE Trans. Information System*, (vol. E77-D, no. 12):1321–1329, 1994.

[26] D. Mizell. Virtual reality and augmented reality in aircraft design and manufacturing. *WESCON/94. Idea/Microelectronics. Conference Record*, 1994.

[27] OpenCV. About opencv [online]. `http://opencv.org/about.html`, 2012 [cit. 2013-01-02].

[28] OpenCV. Geometric Image Transformations – resize [online]. `http://docs.opencv.org/modules/imgproc/doc/geometric_transformations.html#resize`, 2013 [cit. 2013-05-12].

[29] OpenMP. Openmp: The openmp api specification for parallel programming [online]. `http://www.openpm.org`, 2013 [cit. 2013-05-03].

[30] A. Parker, V. Kenyon, and D. Troxel. Comparison of interpolating methods for image resampling. *IEEE Trans. Medicine Imaging*, (vol. 2, no. 1):31–39, 1983.

[31] J. Rolland, Y. Baillot, and A. Goon. A survey of tracking technology for virtual environments. *Fundamentals of Wearable Computers and Augmented Reality*, pages 67–112, 2001.

[32] S. Schwartzman. *The Words of Mathematics: An Etymological Dictionary of Mathematical Terms Used in English.* The Mathematical Association of America, 1st edition edition, 1996. ISBN 0883855119.

[33] L. Thomason. Tinyxml2 [online]. `https://github.com/leethomason/tinyxml2`, 2012 [cit. 2013-04-02].

[34] Vuforia. Vuforia: Developing with vuforia [online]. `https://developer.vuforia.com/resources/dev-guide/`, 2012 [cit. 2013-03-02].

[35] D Willers. Augmented reality at airbus – tracking: Experiences, restrictions & future scenarios. *International Symposium on Mixed & Augmented Reality*, 2006.

[36] F. Zhou, H. Been-Lirn Duh, and M. Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. *IEEE International Symposium on Mixed and Augmented Reality*, pages 193–202, 2008.

Appendix A

Content of DVD

This work contains a DVD with implementation of described algorithms, installation files of applications and libraries used in this work and obtained datasets. A brief description of DVD'd structure follows:

- `datasets/` – a folder with obtained or generated datasets.
- `documentation/` – a folder that contains source code documentation generated automatically by Doxygen.
- `instal/` – installation files for tools necessary for the source code to work are located in this folder.
- `latex/` – a folder containing LaTeXsource code of this thesis.
- `samples/` – a folder containing several already created procedures and their data.
- `src/` – a folder with source codes of created applications.
- `projekt.pdf` – this text in electronic form.
- `README.TXT` – a text file with a more detailed description of DVD's content.

Appendix B

Installation manual

Since installation of used libraries took a considerable amount of time, necessary steps for ALVAR library installation are provided in this Appendix. Because there were several unsuccessful attempts to install ALVAR on various Linux distributions, only a Windows installation is briefly described here, which is rather simple. Installation manual is aimed on Windows 7 SP1 with Visual Studio 2010 installed on it. All installation files can be found on DVD in folder `Installation files`.

1. Install GLUT Toolkit from subfolder `glut`.
2. Install Open Scene Graph from subfolder `openscenegraph`
3. Install OpenCV 2.4.0[1]
4. Install ALVAR library; a comprehensive installation manual for Windows can be found on [1].
5. In Visual Studio, for each project select *Project* → *Properties* and:
   - Add ALVAR's and OpenCV's include directories to *C/C++* → *General* → *Additional Include Directories*.
   - Add ALVAR's and OpenCV's library directories (containing `.bin` files) to *Linker* → *General* → *Additional Library Directories*.
   - Add following ALVAR's and OpenCV's libraries (containing `.bin` files) to *Linker* → *Input* → *Additional Dependencies*: `alvar200.lib`, `alvarplatform200.lib`, `opencv_core240.lib`, `opencv_imgproc240.lib`, `opencv_highgui240.lib`, `opencv_ml240.lib`, `opencv_video240.lib`, `opencv_features2d240.lib`, `opencv_calib3d240.lib`, `opencv_objdetect240.lib`, `opencv_contrib240.lib`, `opencv_legacy240.lib`, `opencv_flann240.lib`, `opencv_nonfree240.lib`. Add `d` at the end of each library's name for additional debugging information.
6. It should be possible to build and run the whole solution now.

[1] This version is recommended by developers of ALVAR as well; newer version of OpenCV caused malfunction of ALVAR, when attempted to install.

Appendix C

Applications' command arguments

This section lists and explains all command arguments needed to launch created applications. All arguments are mandatory; if there are multiple options how to launch an application, all of them are listed. Other parameters, that are not passed as arguments but were rather fixed, are located in header file `Constants.h`.

If application can be controlled during its execution, its controls are listed and explained here as well.

C.1 ARfTaEoAM

`ARfTaEoAM.exe procedure_xml_file videofile`

`-h` or `--help`	print application help
`procedure_xml_file`	XML file describing a procedure and its data locations in format as presented in Appendix D
`videofile`	video file containing record of procedure described by first argument

Table C.1: ARfTaEoAM – command arguments.

`<esc>`	ends application
`<spacebar>`	pauses/resumes video
`<n>`	switches to next procedure step
`<p>`	switches to previous procedure step
`<d>`	shows detected features
`<+>`	increases transparency of window with step description image
`<->`	decreases transparency of window with step description image
`<t>`	shows/hides actual step's instructions and description image
`<a>`	shows/hides original AoIs not affected by history
`<o>`	switches between ALVAR and OpenCV algorithm for homography matrix calculation

Table C.2: ARfTaEoAM – controls.

C.2 Camera Calibrator

```
CameraCalibrator.exe chessboards_filename chessboard_cols chessboard_rows
```

`-h` or `--help`	print application help
`chessboards_filename`	textfile containing paths to chessboard images; one image path per line, last line is empty; see Appendix E
`chessboard_cols`	number of horizontal inner corner points of chessboard
`chessboard_rows`	number of vertical inner corner points of chessboard

Table C.3: CameraCalibrator – command arguments.

C.3 Frames Extractor

```
FramesExtractor.exe video_filename
```

`-h` or `--help`	print application help
`video_filename`	path to video to be played and its frames to be saved

Table C.4: FramesExtractor – command arguments.

`<spacebar>`	pauses/resumes video
`<esc>`	ends application
`<s>` or `S`	saves actual video frame on disk

Table C.5: FramesExtractor – controls.

C.4 Markerless Creator

```
MarkerlessCreator.exe textfile
MarkerlessCreator.exe image_filename output_dat_file_filename
```

`-h` or `--help`	print application help
`textfile`	name of textfile containing rows with entries in format `image_filename[delimiter]output_filename`
`image_filename`	name of input image (including extension)
`output_dat_file_filename`	name of output data file (including extension)

Table C.6: MarkerlessExtractor – command arguments.

Appendix D

Procedure input XML file sample

An example of XML file, containing all data necessary for the main application to run, is provided and explained here.

```
<procedure name="Name of the procedure">
  <step id="1" image="image1.png">
    <description>
      Manual text for this step
    </description>
    <classifiers>
      <item id="1" name="detector1.dat">
        <aoi x="0" y="0" width="100" height="100"/>
        <aoi x="50" y="50" width="80" height="70"/>
      </item>
    </classifiers>
  </step>
  <step id="2" image="image_with_additional_information2.png">
    <description>
      Manual text for this step
    </description>
    <classifiers>
      <item id="1" name="detector2.dat">
        <aoi x="0" y="0" width="100" height="150"/>
      </item>
    </classifiers>
    </step>
</procedure>
```

- `image1.png` – path to image that will partially overlap the recorded video in current step.
- `detector1.dat` – path to detector trained on model that will be used for detection in current step.
- There has to be at least one step in the procedure.
- There can be 0 to n `<aoi>` and `<item>` tags.

Appendix E

Calibration input text file sample

A format of the input text file for camera calibration is explained here. On each line of this file, there is a path to an image containing a chessboard. This path can be either relative or absolute. As stated in Section 7.2.1, it is recommended to put at least 10 - 20 detectable chessboards to this text file; otherwise, the output (camera intrinsic matrix and distortion parameters) may not remove distortion properly.

```
chessboard_image_filename1.png
chessboard_image_filename2.png
chessboard_image_filename3.png
chessboard_image_filename4.png
chessboard_image_filename5.png
chessboard_image_filename6.png
⋮
chessboard_image_filenameN.png
<newline>
```

Appendix F

Markerless Creator input text file sample

Input file example for MarkerlessCreator application is showed below. Each life of this file describes an input image for detector training and a location of the output `.dat` file, respectively. These two file paths are separated by a delimiter, which can be set to any desirable string of characters; in this implementation, delimiter is set to `<tab>`.

```
object1.png[DELIMITER]learned_object1_datafile.dat
object2.png[DELIMITER]learned_object2_datafile.dat
object3.png[DELIMITER]learned_object3_datafile.dat
object4.png[DELIMITER]learned_object4_datafile.dat
object5.png[DELIMITER]learned_object5_datafile.dat
object6.png[DELIMITER]learned_object6_datafile.dat
⋮
objectN.png[DELIMITER]learned_objectN_datafile.dat
<newline>
```

Appendix G

Code metrics

Source files	23
Directories	14
LOC, Lines of code	7991
BLOC, Blank lines	1359
SLOP-P, Executable Physical	4579
SLOP-L, Executable Logical	3056
C&SLOC, Code & Comment	235
CLOC, Comment Lines	2054
CWORD, Comment Words	12470

Table G.1: Code metrics.