



Silesian University of Technology

**Faculty of Automatic Control, Electronics
and Computer Science**

Master's Thesis

**Interactive business cards based
on Augmented Reality idea**

Author: Filip Sitko

Supervisor: dr inż. Michał Kawulok

Gliwice, September 2011

Contents

1. Introduction.....	3
1.1. Problem Definition.....	3
1.2. History.....	6
1.3. Applications.....	9
1.4. Requirements.....	12
2. Augmented Reality problem analysis	13
2.1. Display.....	13
2.2. Video capture	21
2.3. Marker detection	24
3. Project Design	30
3.1. Library choice.....	30
3.2. Development Process	32
3.3. Algorithm Design	33
3.4. GUI	44
4. Internal Specification	45
4.1. Main program functions	45
4.2. Parameters	49
4.3. Algorithm block diagram	50

5. External Specification	52
5.1. 'How to' instruction	52
5.2. Errors	54
6. Testing and results analysis	55
6.1. Marker choice analysis.....	55
6.2. Environment dependencies	70
6.3. Camera parameters	72
7. Summary.....	73
8. Bibliography	74
9. Contents of the CD.....	76

Chapter 1

Introduction

Perception can be certainly considered as an essential factor of human life. Every information about the environment we are living in is received by our senses. Despite of human body imperfections people always tried to improve their perception skills and their inventions helped to find the new ways to explore and understand the surrounding world. Augmented Reality idea introduces a new dimension of perception and opens vast new possibilities that will aid nearly every area of human life.

1.1. Problem Definition

In fundamental terms, the Augmented Reality, often abbreviated to AR is an area of Mixed Reality that refers to the real-time view of a physical world which is augmented by elements generated or triggered by a computer input and can be considered as the connection between the real world and the virtual one. Given real subject image captured by a camera is processed and combined with virtual layers (such as graphics, sounds, data and even smells which are triggered by computer input).

Most common definition was created by Ronald Azuma who described it as follows:[1] “Augmented Reality (AR) is a variation of Virtual Environments (VE), or Virtual Reality as it is more commonly called. VE technologies completely immerse a user inside a synthetic environment. While immersed, the user cannot see the real world around him. In contrast, AR allows the user to see the real world, with virtual objects superimposed upon or composited with the real world. Therefore, AR supplements reality, rather than completely replacing it. This survey defines AR as systems that have the following three characteristics:

- Combines real and virtual
- Interactive in real time
- Registered in 3-D”



Figure 1.1.1. Real desk with a virtual lamp and two virtual chairs.

Augmented Reality is commonly mistaken with Virtual Reality, hence to provide better understanding of Augmented Reality the Paul Milgram’s Virtuality Continuum[2] graph (Figure 1.1.2.) should be introduced to show the general classification of Mixed Reality areas and their unique features.

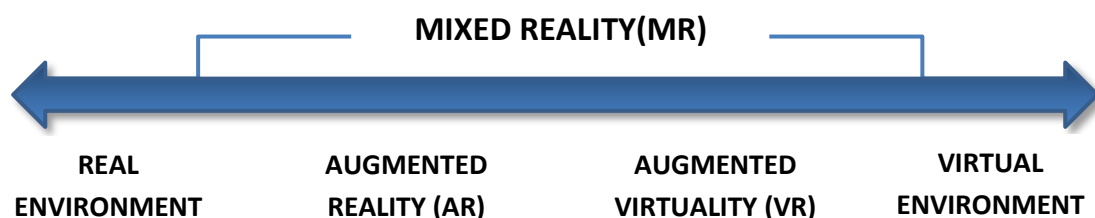


Figure 1.1.2. Paul Milgram’s Virtuality Continuum graph.

Nowadays technology based on Mixed Reality is rapidly developed and distinct boundaries of each area are impossible to define[3]. However, to remark the main differences between them each one can be described by a short definition and unique features:

- **Real environment:**
View of the real, physical world as it can be perceived directly.
- **Augmented Reality(AR):**
Real world view augmented by a computer-generated inputs which create a possibility of interaction.

- **Augmented Virtuality(AV):**
Virtual space view augmented by a real world inputs most commonly used for Human-Computer Interaction (HCI).
- **Virtual Reality(VR):**
Fully simulated world view which provides environment elements controlled by a real world input.

Reproduction Fidelity (FR) of the virtual image should be proportional to the quality of captured image of the real world. Basing on Reproduction Fidelity graph (Figure 1.1.3.) it can be noticed that to obtain the most realistic views combination the computer-generated models details should be real world image fidelity dependent (e.g. High fidelity 3D model with shadings and textures would look unnatural if projected on a low-resolution monoscopic video).

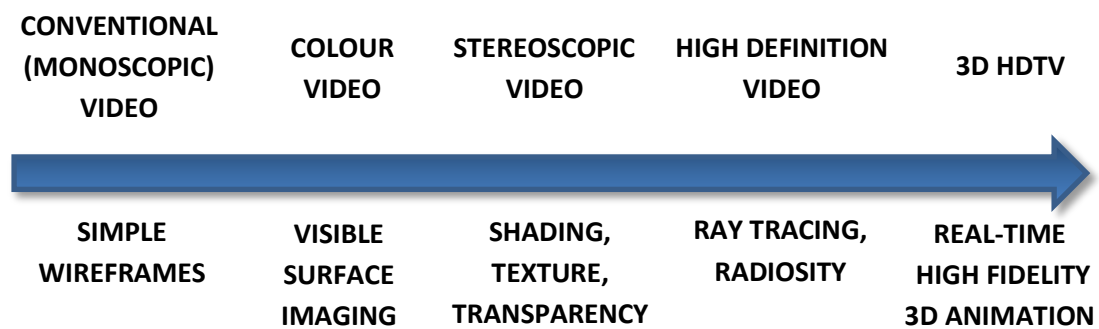


Figure 1.1.3. Paul Milgram's Reproduction Fidelity(FR) graph

1.2. History

Augmented Reality in form which is known nowadays was imagined as a technology of the future since the first computer was designed. People could observe multiple applications of AR in science-fiction movies but did not know that this concept was already researched. Rapid development of AR can be noticed within last 10 years and is commonly considered to be the one of the inventions of XXI century.

This belief is incorrect as the beginning of AR is dated for 1962 as Morton Heilig created a bicycle simulator called Sensorama (Figure 1.2.1.) based on multimodal (multi-sense) technology. The machine could provide stereoscopic 3D vision in wide-angle view, body tilt, stereo sound and even wind tracks and smells triggered as the film was displayed. As nearly all senses were involved during the simulation Sensorama gave the general idea of Augmented Reality which was developed further using the computer.

Ivan's Sutherland invention of the first head-mounted display named Sword of Damocles (Figure 1.2.2.) in 1968 was the next milestone in AR history. Allowing to see computer-generated wireframe rooms according to user head position it gave the background for user interaction with virtual world.

In 1975 Myron Krueger established artificial reality laboratory called Videoplace. Based on cameras, projectors and computer hardware it created an interactive artificial environment for the first time.

These inventions aroused interest of Augmented Reality and from this point it became a popular subject of studies and computer science research. In 1989 Jaron Lanier coined the 'Virtual Reality' phrase by leading the company that sold VR goggles and gloves. In 1990 Tom Caudell, an aircraft manufacturer, popularized 'Augmented Reality' phrase.

1994 introduced Paul Milgrim's Virtuality Continuum concept (Figure 1.1.1.) and classified Augmented Reality as an area of Mixed Reality specifying its boundaries and unique features.

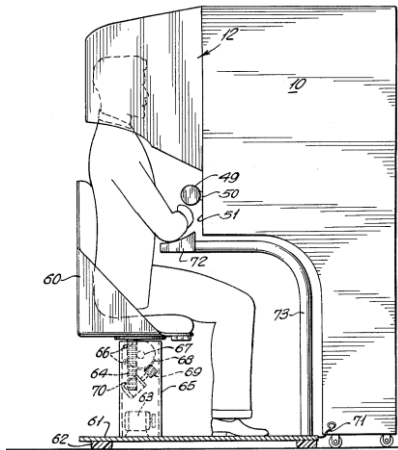


Figure 1.2.1. Morton Heilig's Sensorama. [Figure 5 of U.S. Patent #3050870]



Figure 1.2.2. Ivan's Sutherland Sword of Damocles[4]

In 1996 Jun Rekimoto presents 2D matrix markers (Figure 1.2.3.) as a square-shaped barcodes, one of the first marker systems to allow simultaneously identify real world objects and estimate their coordinate systems.

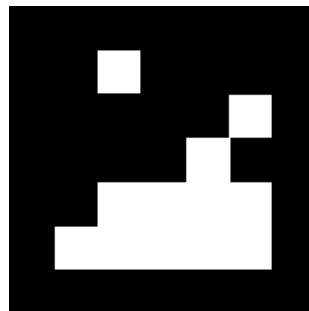


Figure 1.2.3. Jun Rekimoto's exemplary 2D matrix marker[4]

Commonly known definition of the Augmented Reality term and its field was defined by Ronald Azuma in "A Survey of Augmented Reality" in 1997. His paper survey described first AR systems from basis and it is popular material even for today's research purposes.

The release of the ARToolkit (Figure 1.2.4.) - open source computer vision tracking library developed by Hirokazu Kato at the HITlab in 1999 was the real milestone for the AR research. It began a new wave of interest among developers and opened new possibilities for AR programming.

The first game based on Augmented Reality concept “ARQuake” (Figure 1.2.5.) was developed in 2002 and started by its inventor Bruce H. Thomas. It provided outdoor first-person shooter based on virtual environment generated upon real world captured images.

ARToolkit was redesigned and ported to Adobe Flash (FLARToolkit) by Tomohiko Koyama (Saqoosha) in 2009 bringing Augmented Reality to web browsers and starting a new trend wave of web-based applications.

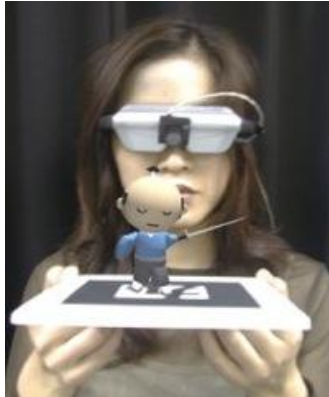


Figure 1.2.4. ARToolkit based 3D model projection[4]



Figure 1.2.5. ARQuake outdoor combined views image[4]

Augmented Reality technology is rapidly developed nowadays for variety of platforms starting with personal computers through mobile devices and ending with touch screens and technical aparatures. According to Rick Oller:[5] “The future of augmented reality holds tremendous promise. With display technology getting better, smaller, lighter and requiring less power every year, it is only a matter of time before augmented reality displays can be fitted to an ordinary pair of glasses, or even contact lenses.”

1.3. Applications

Augmented Reality idea provides variety of new perception and interaction possibilities which are used in almost every field of human life. The applications of AR are practically limited only by imagination and technical development process. Although AR is rather new technology the rapid thrive in development brought it to daily life often staying unnoticed. Focusing only on common applications Augmented Reality can be used for:

- **Navigation:**

AR can be used as a navigation tool which helps to define object positions in challenging environmental conditions. Combined with GPS functionality outdoor objects recognition and virtual routes can provide all the necessary data to enhance navigation process. Digital maps and object/human position indication aid the exploring and tracking tasks.

- **Sightseeing:**

Stored data with history and object descriptions, interactive animations and audio guides and even visual routes to places of interest are really attractive additions for sightseeing tours. AR technology can provide it almost everywhere, so usually head-mounted or mobile displays are designed for this purpose as their size and mobility are essential.

- **Military:**

Detailed mission briefings, GPS-based navigation routes and location data, digital maps, enemy locations or even enhanced firing systems are introduced to the military with AR technology usage. Head-mounted display and goggles with addition of other devices (GPS receivers, orientation trackers, computers and handheld control devices) are the most suitable solutions for this propose as it requires to be mobile since it is used by troops.

- **Medicine:**

Medical students use the technology to practice surgery in a controlled environment. Visualizations aid in explaining complex medical conditions to patients. Surgery risk ratio is significantly reduced as surgeons get improved sensory perception during

complicated operations. AR combined with MRI or X-ray systems can be an invaluable tool bringing all of them into one view.

- **Maintenance and task support:**

Using head worn display a mechanic can be aided by additional data, instructions and label for specified object's parts. Repairing procedures can be provided in adverse environmental conditions and specialist training expenses are reduced by using simulations.

- **Advertising and promotion:**

Promotion through interactive animations, 3d models and games is becoming popular. New web services, company products and even movies are advertised with simple AR gadgets enticing to play and interact.

- **Entertainment:**

AR technology influenced the gaming market starting new type of interactive entertainment applications and games to emerge. Bringing 3D virtual world into reality created fertile ground for developing new mobile and outdoor games and social such as sporting events and concerts.

- **Education:**

Since beginnings, AR technology stayed in association with educational institutions. Many AR research breakthroughs have been accomplished by college and universities teams, as prototypes and still developed devices were available as multipurpose educational tools. Providing possibility of real-time processing it can be used for presentations, training simulations and development research.

- **Industry:**

Augmented reality can provide hands-free visual overlays of dynamic manufacturing information targeted to specific, highly controllable automated and semi-automated assembly environments. Computer generated virtual project prototypes can replace the real ones reducing the final product expenses.

- **Architecture:**

Virtual models mockups and simulations could be projected on one platform aiding the design and planning process. As it provides possibility of collaboration on shared models it can be

used as a powerful tool improving planning and communication process.

- **Translation:**

Real-time dynamic subtitles display and text translation can really enhance communication process. Font and text recognition and even simple mathematical problems solution can be achieved using this unique feature.

These are only the main applications of Augmented Reality narrowed to the ones we are using nowadays. [5,6] Taking into account technical development progress we can expect more of them in near future as every unique area of life can be simplified, each object can be augmented to be more usable and each action can give extraordinary experience which cannot be obtained in real life.



Figure 1.3.1. Virtual fetus inside womb of a pregnant woman.
(Courtesy UNC Chapel Hill Dept. of Computer Science)[1]



Figure 1.3.2. Head-up display for a fighter plane[3]

1.4. Requirements

Final project's design should provide simple AR elements to the final business cards labeled by 2D black and white markers. Created application should meet the following general requirements:

- **Main detection algorithm written in C++:**
C++ usage should enhance the algorithm speed and influence on overall performance level.
- **High accuracy marker recognition:**
Business card markers must be detected and matched with selected template to display a virtual graphical element. High spectrum of marker angle and environment lightness level acceptance would improve the interaction process.
- **Real-time image processing (min 12 FPS):**
To provide high quality interaction acceptable frames per second ratio has to be achieved. Lower FPS processing would create an illusion of delay.
- **Image and video display:**
Simple computer-generated graphical inputs should be combined with real, physical world image capture. Usage of AR technology in this case will project photos and videos that will aid the personal identification of the business card holder.
- **Graphical User Interface:**
Simple and intuitive GUI that would make the whole application more user-friendly and allow to use every program feature in a convenient way.

Chapter 2

Augmented Reality problem analysis

There are several approaches to obtain the desired effect for Augmented Reality implementation. However, each of them has the unique features designed for specific type of devices and environments in which they are used in. Some of them should not be used for this project purposes, hence strengths-weaknesses analysis of each approach property will reveal most suitable solution to achieve the project goals.

2.1. Display

Augmented Reality based technology is classified in regard of displays used for combined computer- generated input and real physical world captured images visualization. According to Oliver Bimber and Ramesh Raskal:

“Augmented reality displays are image-forming systems that use a set of optical, electronic, and mechanical components to generate images somewhere on the optical path in between the observer’s eyes and the physical object to be augmented”[10].

There are 3 major display techniques used for design of Augmented Reality based devices and applications (Figure 2.1.1). The division into head-attached, handheld and spatial displays helps to identify the advantages and disadvantages of each group. This way the final project design can be created according to its application and the display choice.

To familiarize with these solutions the main display groups should be described:

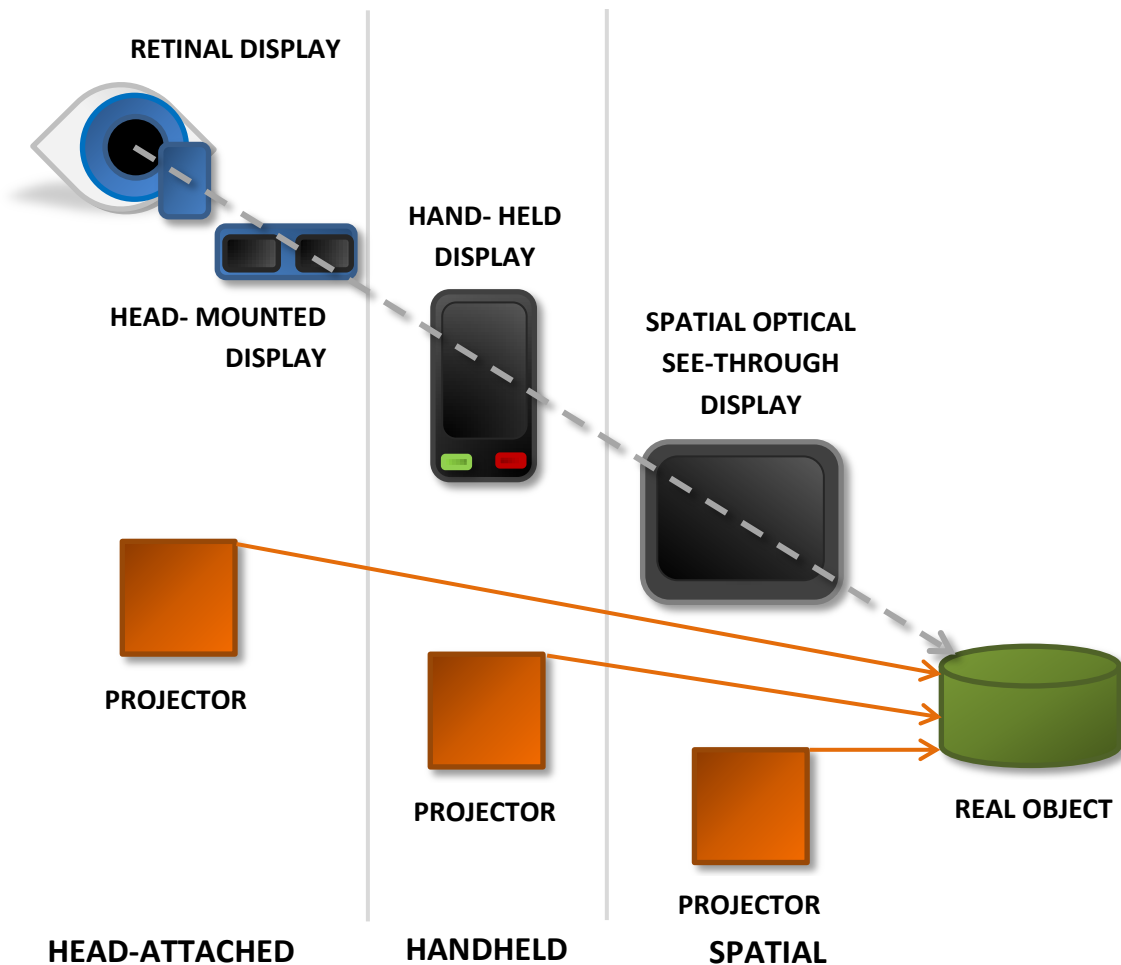


Figure 2.1.1. Image generation graph for augmented reality displays inspired by Oliver Bimber and Ramesh Raskar[10]

- **Head-Attached displays:**

Head attached displays require the AR display system worn on a user's head. Depending on the image generation technology there are 3 main types distinguished:

- **Retinal Displays:**

Instead of providing screens in front of the eyes, retinal displays utilize low-power semiconductor lasers or special light emitting diodes to scan modulated light directly onto retina of the human eye. The advantage of using this type of display is a higher resolution and a potentially wider field of view in comparison to screen-based displays. Vivid contrast, brightness and low-power consumption make it perfect for mobile outdoor applications (nowadays retinal displays can be fitted into casual glasses and

goggles). However there are some disadvantages which made it not so popular. Due to low-power consumption and cheap components mainly monochrome (red) lasers utilized. Moreover due to the complete bypass of the ocular motor system by scanning directly onto the retina, the sense of ocular accommodation is not supported hence the focal length is fixed. Stereoscopic versions of retinal displays do not exist yet so the application possibilities are very limited. To present the overall retinal displays idea the simplified diagram should be provided (Figure 2.1.2.).

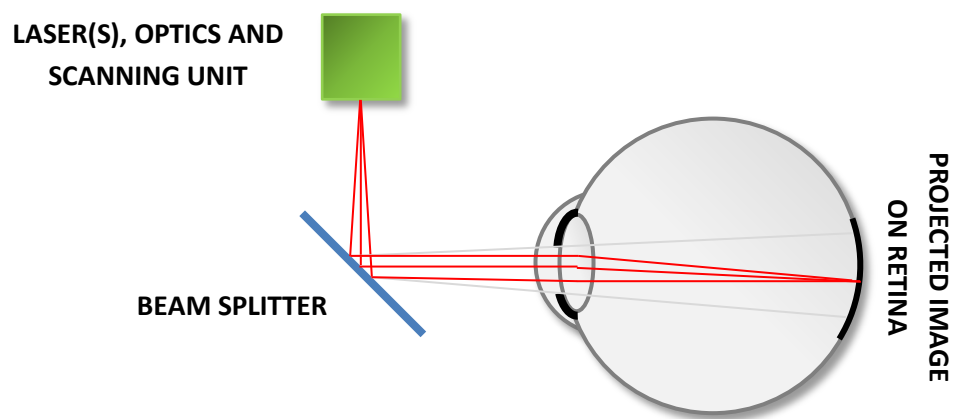


Figure 2.1.2. Simplified diagram of a retinal display inspired by Oliver Bimber and Ramesh Raskar[10]

Head Mounted Displays:

Two different head-mounted display technologies exist to superimpose graphics onto the user's view of the real world:

- Video see-through displays (Figure 2.1.3.): make use of video-mixing and display the merged images within a closed-view head-mounted display.
- Optical see-through displays (Figure 2.1.4.): make use of optical combiners (essentially half-silvered mirrors or transparent LCD displays).

Main advantages of this technology are its mobility and possibility of full color spectrum and fidelity level image generation. Several disadvantages which are inherited from general limitation of head-attached display technology should be also noticed:

- Lack in resolution of generated image (limitations of attached miniature displays)

- Limited field of view (due to limitations of applied optics)
- Visual perception issues (fixed focal length problem occurs as the eyes are constantly forced to either continuously shift focus between the different depth levels, or perceive one depth level as unsharp - mostly for see-through displays)
- Increased incidence of discomfort due to simulator sickness because of head-attached image plane (especially during fast head movements)

Optical see-through devices require difficult (user and session-dependent) calibration and precise head tracking to ensure a correct graphical overlay. Nevertheless head-mounted displays were the dominant display technology within the AR research field. They support mobile applications and multi-user applications if a large number of users need to be supported. To present the overall head-mounted displays idea the simplified diagram should be provided (Figure 2.1.3., Figure 2.1.4.).

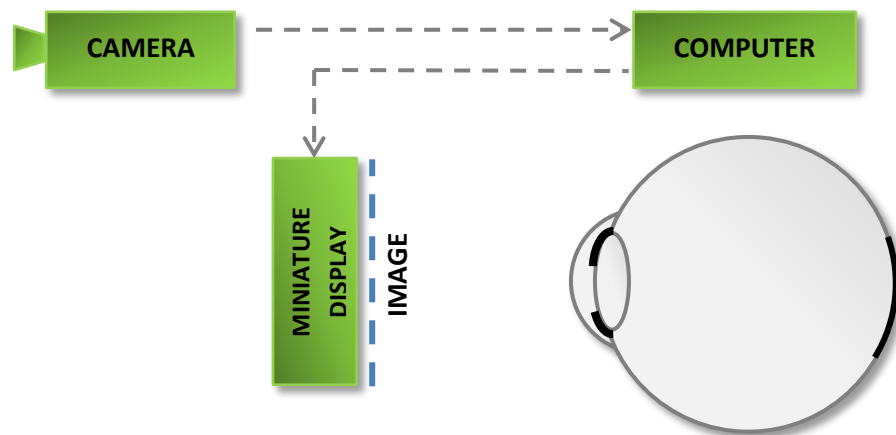


Figure 2.1.3. Simplified diagram of a video see through head-mounted display inspired by Oliver Bimber and Ramesh Raskar[10]

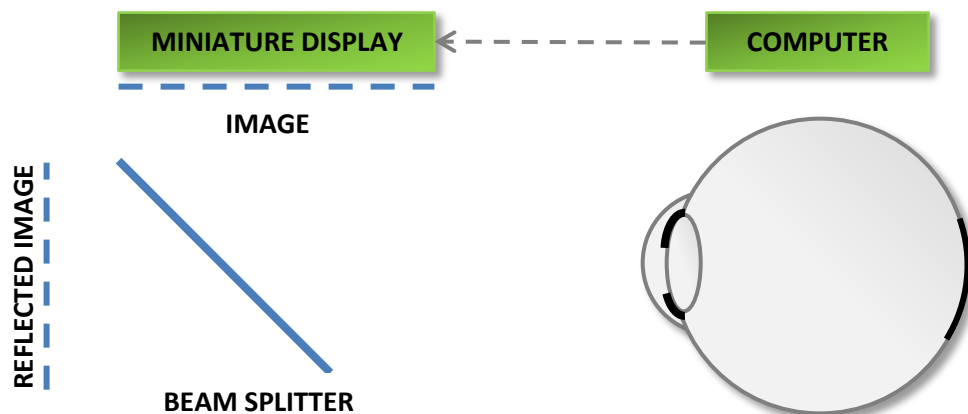


Figure 2.1.4. Simplified diagram of an optical see through head-mounted display inspired by Oliver Bimber and Ramesh Raskar[10]

Head- Mounted Projectors:

Head-mounted projective displays redirect the frustum of miniature projectors with mirror beam combiners so that the images are beamed onto specified surfaces (usually retro-reflective surfaces because of their light-reflective visual properties) that are located in front of the viewer. As whole technology is projector-based it provides wider field of view as the effect of inconsistency and convergence is decreased in comparison with other HMD types, however there are also some shortcomings like limited projector generated image resolution and brightness, necessity of special display surfaces, strong light conditions dependencies and limitations to closed space environment (as the ceiling is necessary for system installation). These factors made head-mounted projector more suitable for Virtual Reality applications. To present the overall head-mounted projectors idea the simplified diagram should be provided (Figure 2.1.5.).

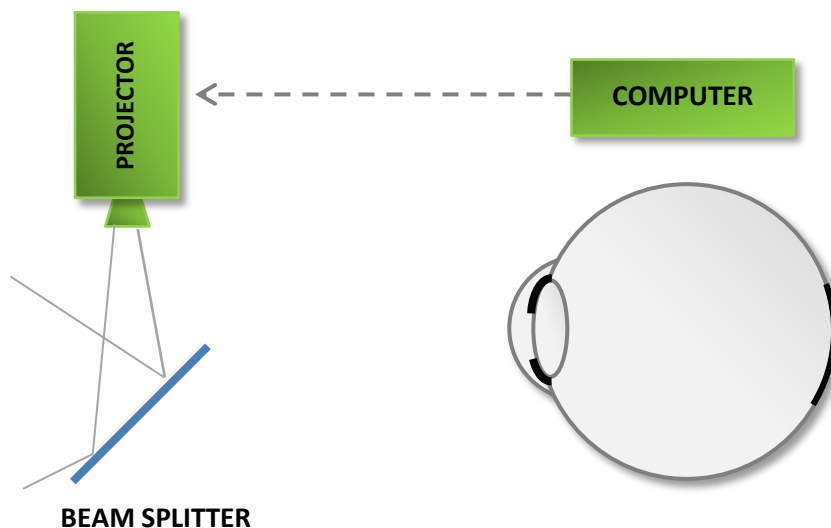


Figure 2.1.5. Simplified diagram of projector head-mounted display inspired by Oliver Bimber and Ramesh Raskar[10]

- **Handheld displays:**

Hand-held display concept provides generating images within arm's reach within single device which combines processor, memory, display, and interaction technology. Whole idea aims at supporting a wireless and unconstrained mobile handling, hence this technology can be utilized using a Tablet PCs, PDAs, personal digital assistants and mobile phones which brought the AR technology to mass market (nowadays AR based entertainment applications are very popular among smartphone users). Mobile optical see-through and hand-held mirror beam combiners and other mobile technology combinations also exist, however classic concept is the most common one due to its advantages like mobility, simplicity and cheap, available device components. There are also some disadvantages that should be introduced. The image analysis and rendering units are processor and memory intensive and require floating point operations, what is critical for low-end devices (mobile phones, PDAs). It also causes slow frame rates and decreases the precision of generated image. Mobile device screen size restricts the covered field of view. However since it can be moved and navigate through an information space (which is definitely larger than a screen size) this technology supports a visual perception phenomenon called Parks effect. Thus far mobile devices cameras were not adapted for AR tasks due to resolution or focus planes limitation, however recent rapid development of mobile devices technologies made them perfect for simple AR tasks. Unfortunately in comparison with other types of displays hand-held devices do not allow to complete any tasks in completely hand-free working environment.



Figure 2.1.6. A first prototype on a conventional consumer cell phone[10]

- **Spatial displays (SAR):**

Spatial displays refer to those display techniques that detach most of the technology from the user and integrate it into the environment. There are 3 unique approaches which differ in environment augmentation process:

Screen-Based Video See-Through Displays:

These systems make use of video mixing and display merged images directly onto monitor's screen. As the output signal video can be provided for regular monitors or projectors the main advantage of this method is the hand-free usage convenience and wide range of observers for a single AR application instance. As to adapt this technique only off-the-shelf hardware components and standard PC equipment is required- this method is considered to be the most cost-efficient. Naturally it also has disadvantages connected with general modern screens issues. Limited screen resolution and size decrease the image quality and user's field of view, however modern development in displays and projectors area makes these factors negligible. The key disadvantage is a remote view rather than a see-through concept which decrease the level of view interactivity.

Spatial Optical See-Through Displays:

The main idea of this technique is to generate images that are aligned within the physical environment using spatial optical combiners, such as planar or curved mirror beam combiners, transparent screens, or optical holograms (Figure 2.1.7.). This method provides more realistic image marked out with easy eye accommodation property, high and scalable resolution, wider field of view and more controllable environment (by tracking, illumination level etc.). Spatially aligned optics generate some shortcoming as lack of support for mobile devices, observers and environment interaction limitations. Due to a screen size limitation the virtual objects outside the display area are unnaturally cropped which is called a window violation effect. Since the generated image is realistic and superimposed to natural environment this technique is commonly used to create holograms and optical illusions.



Figure 2.1.7. Transparent projection screen[10]

Projection-Based Spatial Displays:

Projector-based spatial displays use front-projection to seamlessly project images directly on a physical object surface instead of displaying them on an image plane (or surface) somewhere within the viewer's visual field. To enhance the image quality and interaction level the whole system is a combination of single or multiple static or steerable projectors. Main disadvantage of this technique is shadow-casting of physical objects and interacting users due to front-projection. Display area constrained to the size and shape of the physical object and increased complexity of consistent geometric alignment and color calibration as the number of applied projectors increases makes the whole system hard to calibrate especially for 3D objects. This technique is commonly used for presentations and outdoor structures visualizations.

As the purpose of this project is based on the business card image augmentation it is aimed to a bulk consumer. Taking it into consideration the display technique should mark out with availability and mobility to provide fast and simple augmentation method. These requirements can be satisfied by hand-held displays and spatial screen-based video displays. Hence the PC application can be applied for a wider spectrum of consumers and cost-efficient approach needed the screen-based video display technique was chosen for this project purposes. This solution applied to notebooks combines advantages of both techniques bringing all the disadvantages to the minimum.

2.2. Video capture

Choice of Augmented Reality display technique depends on the video capture method used for designed application. As AR environment should present the area as 3D plane there are two main methods that provide this desired effect. To introduce them both it is essential to understand the main properties of human perception. An interesting definition and main principles of 3D viewing was formulated by Hong Hua: [11]

“Human eyes rely on many visual cues to perceive and interpret depth in the real world. Such depth cues can be monocular or binocular. Monocular depth cues are observed only with one eye and common examples include perspective, occlusion, texture gradients, distribution of light and shadows, and motion parallax. Binocular depth perception is based on displacements (i.e. binocular disparity) between the projections of a scene object onto the left and right retina due to eye separation. The binocular disparity is processed by the brain, giving the impression of relief in an effect known as stereopsis. Stereoscopic displays enable depth sensation by exploring the binocular disparity.”

Single camera video capture:

This image capture method is commonly used in a web-based AR applications focused on the bulk consumer due to its simplicity. As only single camera is required the webcams already installed in mobile devices or cheap standalone cameras are sufficient. The main idea of AR environment creation then is to generate a proper algorithm to evaluate the 3D space basing on the objects position and perspective on the captured image. Then the virtual objects are superimposed onto the image according to evaluated object location properties. This method however generates 2D images only with an illusion of 3 dimensional space. The modern development of image processing introduced the stereoscopic 3D image generation out of 2D one, however it requires application of complex algorithms which would decrease the FPS (frames per second) rate and certainly would not be as accurate as stereoscopic video capture (but is perfect for already captured video files which do not need to be processed in real-time).

Stereoscopic video capture:

Stereoscopic video capture method is more complex and requires the usage of two cameras, however it provides more detailed, virtual 3D environment. As two cameras capture the image simultaneously, the main issue is the proper synchronization of both videos. Having this problem

solved it is possible to specify each object's position in 3D space and generate more detailed virtual environment. It provides a possibility to superimpose 3D virtual model onto the obtained image very accurately in two different perspectives. In conclusion with the usage of simple 3D glasses the viewer gets very realistic and precise image which can be generated in real-time.

Camera calibration:

Both approaches are affected by shortcoming of the camera usage. In theory there is a possibility to define a perfect lens that introduces no distortions. However, every lens used in camera generates two types of distortion. The first one is the radial distortion (Figure 2.2.1.) caused by the shape of lens. It occurs as an image bulging phenomenon known as a barrel effect-rays farther from the center of the lens are bent more than those closer in. For radial distortions, the distortion is equal to 0 at the (optical) center of the imager and increases each step toward the periphery.

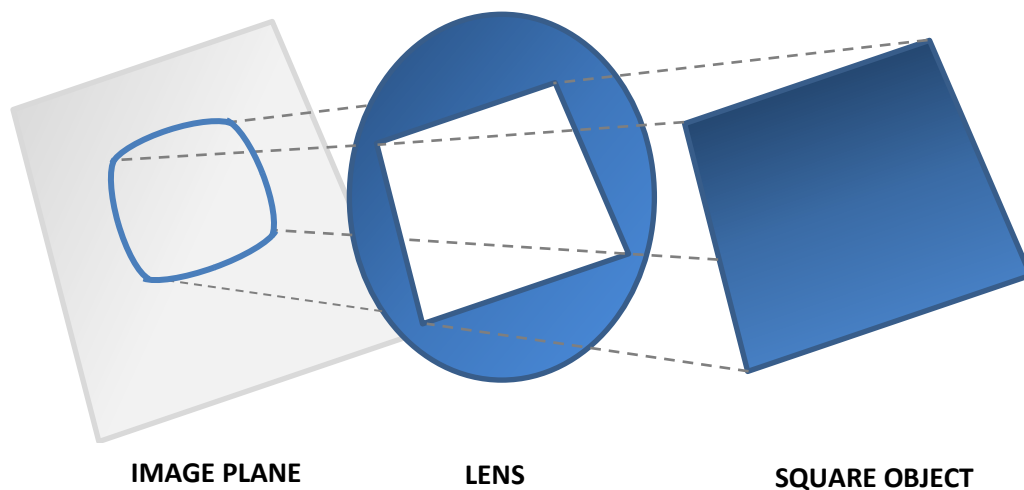


Figure 2.2.1. Radial distortion: rays farther from the center of a simple lens are bent too much compared to rays that pass closer to the center; thus, the sides of a square appear to bow out on the image plane (this is also known as barrel distortion)[7]

Tangential distortion results when the lens is not fully parallel to the image plane. This effect can be usually observed in cheap cameras in which the imager is glued to the back of the camera. It occurs as an tangential image perspective deformation. There exist other kinds of lens distortion but typically they have lesser effect on obtained image. To obtain the corrected

image the camera calibration should be applied. In practice, this distortion is small and can be characterized by the first few terms of a Taylor series expansion around $r = 0$. For cheap web cameras, we generally use the first two such terms: the first of which is conventionally called k_1 and the second k_2 . For highly distorted cameras such as fish-eye lenses we can use a third radial distortion term k_3 . In general, the radial location of a point on the imager will be rescaled according to the following equations:

$$\begin{aligned} x_{corrected} &= x(1 + k_1r^2 + k_2r^4 + k_3r^6) \\ y_{corrected} &= y(1 + k_1r^2 + k_2r^4 + k_3r^6) \end{aligned} \quad (1)$$



Figure 2.2.2. Camera image before undistortion (left) and after undistortion (right)[7]

essential (as the see-through technique is unobtainable and not suitable for this project). Single camera video capture was chosen. This method is commonly used within PC entertainment applications as most of mobile computers are equipped with a single webcam and a vast majority of consumers would use them to run the application. This solution also eliminates the issues connected with videos synchronization so it was found to be the most suitable choice.

2.3. Marker detection

The marker detection process is the essential step in designing AR 2D marker based application. Accuracy of this algorithm influences deeply on further steps as if no marker is found, the virtual models should not be superimposed onto the captured image. There are several detection methods which should be introduced to fully understand the detection problem and choose the most suitable solution. Only overall view on detection methods can be found here as the detailed algorithm process is fully described in Chapter 3.3. Whole marker detection analysis would be provided as the full most common algorithm steps with OpenCV[7] library methods applied as this library was chosen for final project implementation (Chapter 3.1.):

2.3.1. Blob detection

To obtain a maximum accuracy for finding markers the captured image should be processed to distinguish and labelize every object that can be the marker.[12] There are two main approaches to obtain it, however only the first steps differ and the rest of the algorithm stays unchanged:

The first one assumes color extraction process as the marker can be designed as a monochromatic (usually red, green or blue) shape or only its corners are colored with different colors to evaluate its rotation value. In this case captured RGB image should be split into 3 channels and each of them should be thresholded to get the final 3 binary images.

The second one is based on the black and white 2D markers marked out by the unique shape. As the colors are not relevant in this approach, whole image should be converted into one channel grayscale and then the threshold filter should be applied to measure only the single pixels luminosity values. This solution is faster, but would not work for color detection.

The rest of the algorithm is the same for both approaches. To enhance the detection process the binary image(s) should be sharpened by the morphology filters erode and dilate. They are usually applied to eliminate noises and image distortions as the single pixels surrounded by the opposite value one are absorbed to the surrounding

regions in the resulting image. Note that this type of correction can slightly change the object's contours, so these filters should be used with relevant parameters (since the overall algorithm performance can be decreased, there is no specific requirement to use them if the benefits are too small). Having the luminosity value of each pixel, the blob detection should be performed to find all pixel groups with the similar values that create convex shapes. The most commonly used type of blob detection in OpenCV is the Laplace operator method. Every time a new pixel is assigned to a blob, the values for the bounding box, area and centroid are updated. Two general blob detection approaches diagram should be presented for better understanding (Figure 2.3.1.).

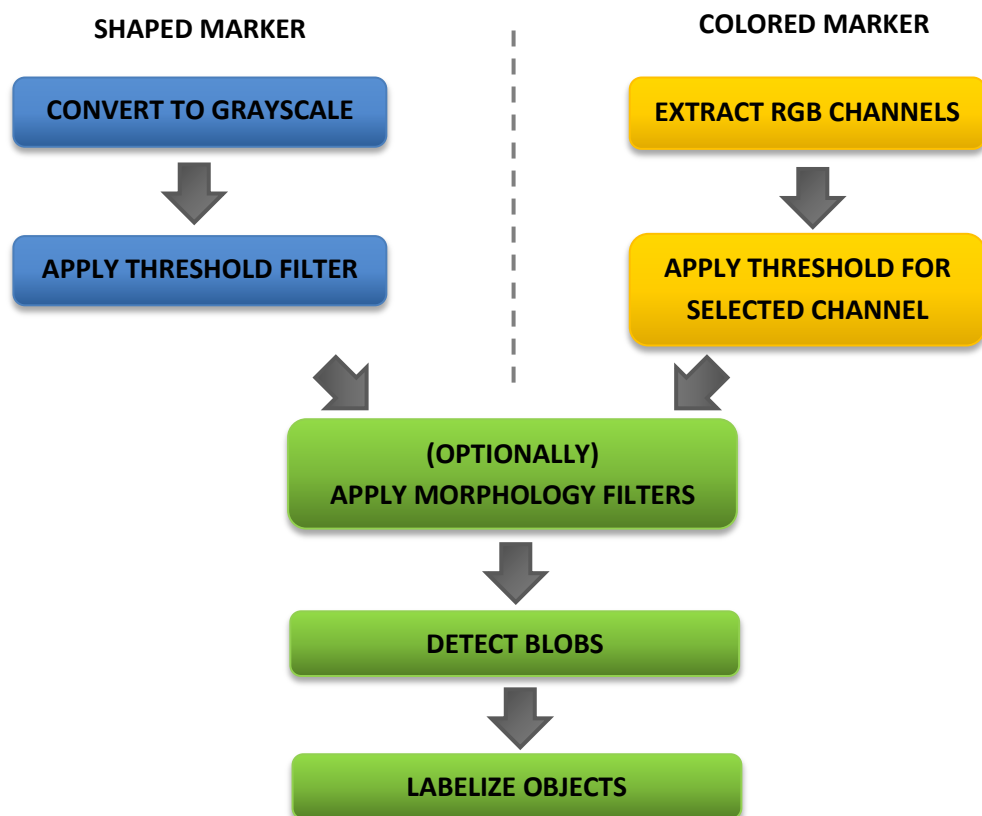


Figure 2.3.1. Block diagram of full blob detection procedure.

2.3.2. Corner finding

Having the objects labeled it is essential to evaluate their position and size parameters which would be used for perspective estimation. In fact, depending on the chosen algorithm the blob detection step can be completely omitted and corner/edge finding can be applied to a raw binary image. In order to obtain the corner values, various algorithm could be utilized, however only most common, OpenCV supported ones would be described:

First of all, the processed image should be converted into a binary image to provide better accuracy for corner detection (there is no need to binarize the image if the blob detection was performed, since the result is already binarized). It can be obtained by applying one of the threshold filters. The classic simple threshold or filters which can provide variable threshold level: adaptive threshold (Gaussian/mean value based) or histogram-based hysteresis threshold should be chosen depending on environmental conditions in which the application would be used. Instead of simple thresholding the Canny edge detection combining threshold and contours finding main features can be used. It is based on computation of the first derivatives in x and y and then combined into four directional derivatives. The points where these directional derivatives are local maxima are then candidates for assembling into edges. The most significant new dimension advantage is assembling the individual edge candidate pixels into contours. They are formed by applying a hysteresis threshold to the pixels. This means that there are two thresholds, an upper and a lower. If a pixel has a gradient larger than the upper threshold, then it is accepted as an edge pixel - if a pixel is below the lower threshold, it is rejected. If the pixel's gradient is between the thresholds, then it will be accepted only if it is connected to a pixel that is above the high threshold.

Depending on the shape of the marker to be found there are several choices to be made in order to extract shape features. If the desired shape is a polygon the best choice is to use the subpixel corner finding algorithm. The actual computation of the subpixel location uses a system of dot-product expressions that all equal 0, where each equation arises from considering a single pixel in the region around the iterated pixel.

OpenCV library provides also the features finding algorithms as the Shi and Tomasi corner finding[8] implementation (based on the second derivatives evaluation (using the Sobel operators) that are needed to compute eigenvalues). This method however is not so

accurate for extracting geometrical measurements so it is not popular for marker detection procedures. Another supported method is the Hough transform algorithm. Even though for polygon-shaped markers Hough line transform is not the best choice, it is very good solution to use Hough circular transform to extract circular-shaped markers. However the most common 2D markers are square-shaped so this is not popular method of marker extraction.

Although algorithms like the Canny edge detector[7] can be used to find the edge pixels that separate different segments in an image, they do not provide any information about those edges as entities in themselves. Having contours evaluated the polygon can be approximated by matching the found contour lines and basing on their intersection points obtain the corners coordinates and create polygon sides. Additionally, convexity check should be performed and if the result is positive the full marker shape is ready for extraction using the corner position coordinates. Both side count and convexity check are supported by OpenCV function `cvApproxPoly` and was used in the final algorithm. General corner finding block diagram should be presented for better understanding (Figure 2.3.2.).

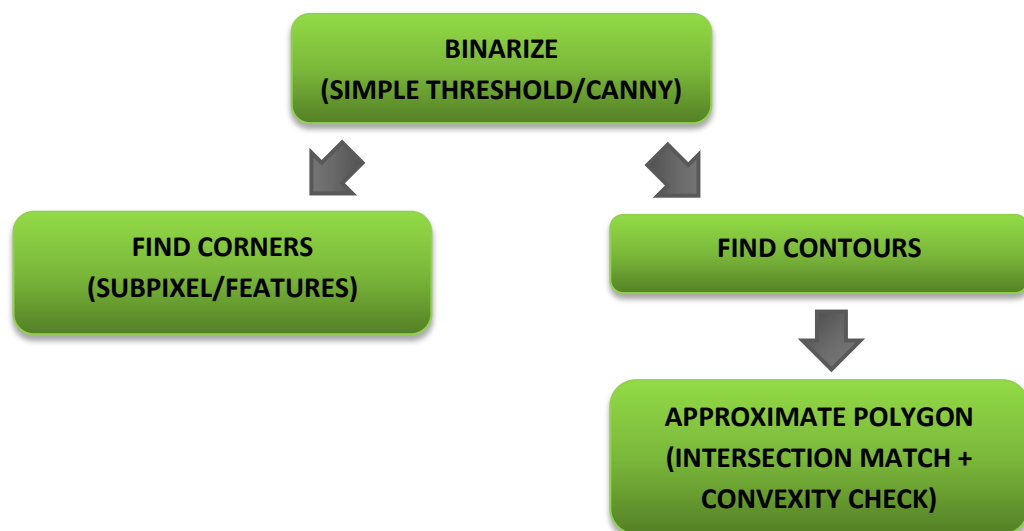


Figure 2.3.2. Block diagram of full corner finding procedure.

2.3.3. Marker tracking

To obtain better detection accuracy the marker tracking algorithm can be applied. Since the general idea assumes that if marker is found once there is no need to look for it on entire image and only neighbourhood pixel regions are scanned for the desired pattern. This way the application performance can be greatly increased as for high resolution images (HD, Full HD and higher) only fixed region would be set as Region of Interest (ROI) in each cycle. Taking into consideration the optical flow of processed image there are two kinds of marker tracking techniques can be distinguished:

Sparse optical flow based technique implemented in Lucas-Kanade[9] tracking algorithm. It requires previously specified template, which should be tracked with unique features like corners, edges or patterns so the marker detection is needed to be performed at least once for the first captured frame. Assuming that the camera view can be changed during the tracking process the marker detection have to be performed each time the specified object is out of reach as the lighting conditions may vary for each view. Sparse optical flow based algorithms are marked out by 3 assumptions: [7]

- **Brightness constancy:** A pixel from the image of an object in the scene does not change in appearance as it (possibly) moves from frame to frame. For grayscale images the brightness of a pixel does not change as it is tracked from frame to frame.
- **Temporal persistence or “small movements”:** The image motion of a surface patch changes slowly in time. In practice, this means the temporal increments are fast enough relative to the scale of motion in the image that the object does not move much from frame to frame.
- **Spatial coherence:** Neighboring points in a scene belong to the same surface, have similar motion, and project to nearby points on the image plane.

Dense optical flow based technique represented by a Horn-Schunck[9] tracking and block matching algorithms. This method is based on pixel displacement measurements so there is a possibility to track specified patterns even without previous marker detection. Some kind of velocity can be associated with each pixel in the frame or, equivalently, some displacement that represents the distance a pixel has moved between the previous frame and the current frame. Although this method is more accurate than the previous one, it is also more complex which manifests very clearly in high computational cost of algorithm. Some additional disadvantages can occur as this technique is related mainly to the motion capture process. If the tracked object is homogenous (e.g. white sheet of paper) its motion would be very hard to track as most of the white pixels from the previous frame will simply remain white in the next one. Only the edges coordinates may change, and even then only those perpendicular to the direction of motion can be accurately tracked.



Figure 2.3.3. Example of features tracking algorithm based on Jean-Yves Bouguet and Pietro Perona Visual Navigation project[13]

As both of these techniques introduce additional computation cost which would negatively influence on overall project performance none of them was used in the final algorithm. Moreover, sparse optical based one requires brightness constancy which cannot be satisfied as the environmental lighting conditions can vary. In fact tracking process is very useful for multiple or unknown object tracking, but for a single fully specified marker there is no need to use additional detection techniques.

Chapter 3

Project Design

3.1. Library choice

During last 10 years the Augmented Reality became a popular object of interest, hence multiple open source libraries were developed and aimed to a mass consumer. Nowadays practically everyone can create simple AR effects using ready-to-use applications scattered throughout the Internet. However, this work is focused on designing the AR algorithm from the beginning, so the existing libraries should only be used for analysis and requirements statement. The majority of available open source libraries (GNU General Public Licence) were developed basing on ARToolKit[15] created in 1999 by Hirokazu Kato at the HITlab. Next versions were firstly created to support different programming languages giving the opportunity to add new features and improve the existing algorithms. As Tomohiko Koyama introduced FLARToolKit[16] supporting ActionScript3 the AR based applications started to emerge throughout the internet and could be run online on websites as simple .swf files. Since these two libraries are known to be revolutionary and each of them introduces new features- both of them were analysed to create an unique algorithm for this work. Many other libraries were created basing on ARToolKit e.g. NyARToolKit (Java, C++, C#, Android), FLARToolKit Alchemy (improved performance), ARToolKit Professional (extended features) - each of them supported other open source libraries for rendering functionality (Papervision3D, OpenSceneGraph, OpenGL) as basically they were created as tracking libraries and nodekits. The key libraries development process can be presented as an evolution graph introduced by Tomohiko Koyama (Figure 3.1.1.):

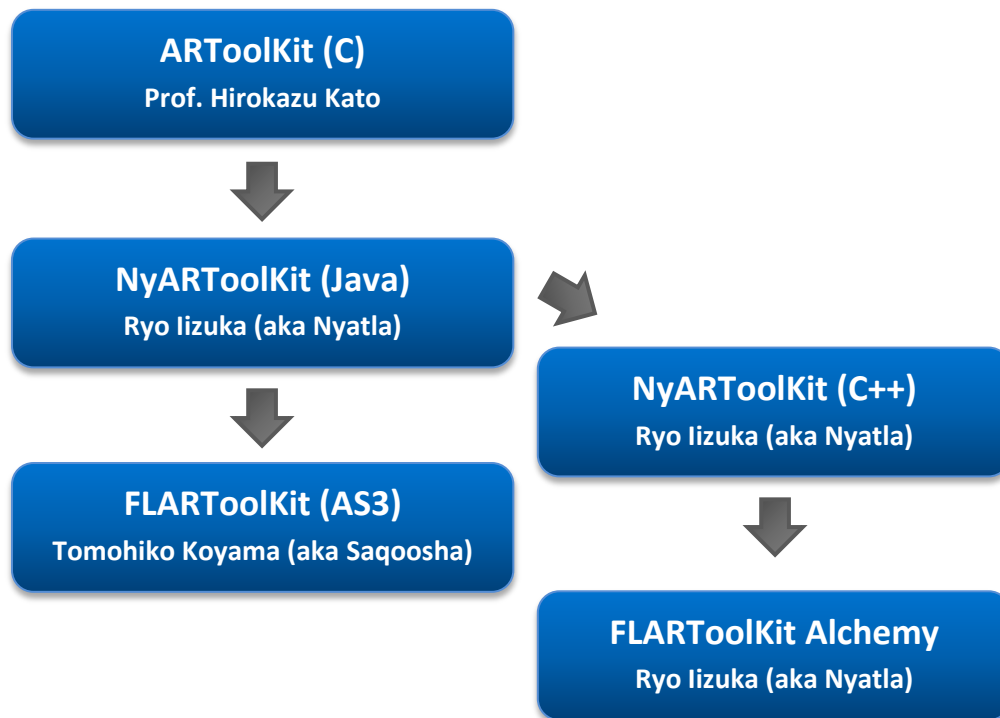


Figure 3.1.1. ARToolKit library evolution graph[15]

Despite of variety of Augmented Reality technology based libraries the Open Source Computer Vision (OpenCV) library was chosen for this project purposes. Since it supports single Computer Vision based actions like image capture, image processing, transformations and calibrations, the whole project algorithm could be created from scratch. This choice created an opportunity to develop basic AR algorithm based on 2D markers and gave the possibility to improve it and extend it with new features in future. Since the whole project was created step by step the simple debug mode could be created to monitor all application variables needed for performance analysis (full performance and results analysis and other libraries features comparison can be found in Chapter 6). OpenCV version 2.1. was chosen due to the version stability as later versions were reported to be unstable and need to be fixed.

3.2. Development Process

Tools used in development process have the direct impact on code quality, overall software delivery time and file archiving. The choice of the proper engineering model was aimed to fast delivery time and secure and reliable code and documentation archives. All these requirements are satisfied by the Continuous Integration software engineering by means of using the Apache Subversion (SVN) software versioning. It gave the opportunity to use simple revision control system hence the project could be developed in small parts giving the access to whole development process and documents creation history.

As this work is basing on Open Source libraries and tools the chosen SVN system was supported by the Google code project hosting as it turned out to be a solution for simple repository creation, control and gave the possibility to share this project under Open Source License.

The main project's page which would be extended by external documentation and 'how to' instructions can be found at:

<http://code.google.com/p/augmented-reality-cards/>

Whole project's repository which includes code, documentation and images, video, templates files can accessed by:

<https://reaveth@augmented-reality-cards.googlecode.com/svn/>

Simplified repository structure graph looks as follows:



3.3. Algorithm Design

The main project algorithm is based on standard 2D Augmented Reality implementation procedures used in the most common Toolkits, however there are many small changes which have a major influence on project performance. These algorithm alterations were focused on the marker detection accuracy improvement mainly, remembering that the frames per second ratio should meet the real-time performance requirement. Each step was designed to create unique marker based AR technique implementation supported by OpenCV library solutions. A simplified algorithm block diagram for single frame is presented below (Figure 3.3.1.):

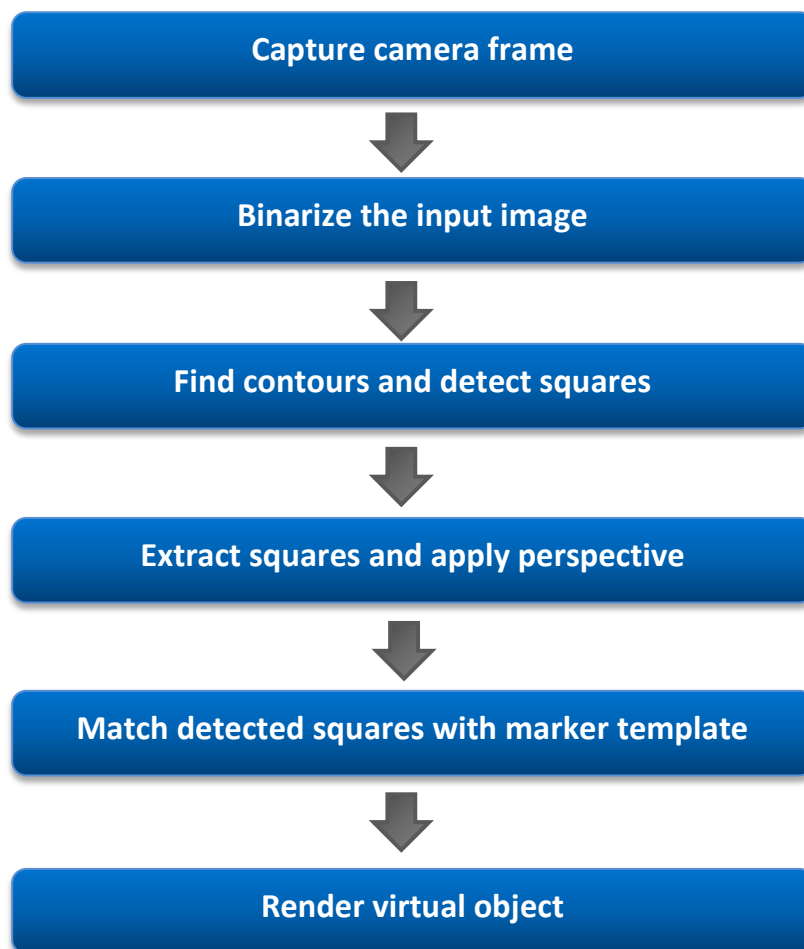


Figure 3.3.1. Simplified Augmented Reality algorithm block diagram

In order to fully understand this unique and improved 2D augmented reality implementation each step should be described in detail and enhancements should be introduced.

3.3.1. Frames acquisition.

Any type of static image can be used as an input for introduced algorithm. However, to satisfy the real-time processing requirements the analyzed image should be streamed by the camera device. In order to get the input for each cycle, the camera has to provide single frames processed and augmented by the virtual element (Naturally the video file previously saved on the disk can also be used - it has to be split into single frames and each of them is then read as input for each cycle. Such a method was used for rendered video file frames acquisition in rendering step). This task was done by means of using the `cvCreateCameraCapture()` which handles the camera image streaming. If the time of getting and displaying each frame is short enough, the generated image sequence is perceived as a video.

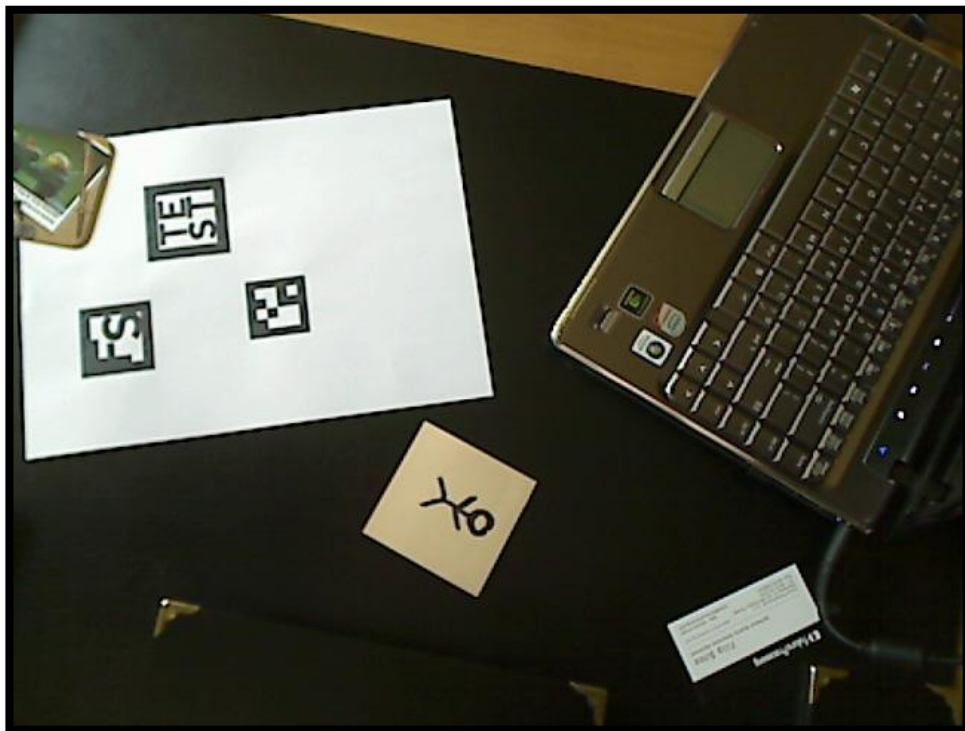


Figure 3.3.2. Original image captured from the camera device

Enhancements:

As the camera capture issue is rather simple and handled by used functions the Frames per Second ratio counter was implemented to provide the necessary information about the application performance (mostly used in debug mode). To achieve satisfactory FPS accuracy the timer has to be initialized when the streaming starts. The number of cycles (processed frames) are counted and divided by the exact time difference (time of the frame acquisition – time of the timer initialization). This way the overall performance can be monitored. It was essential to implement this improvement to check if the real-time processing requirement was satisfied.

3.3.2. Image binarization.

Method selected for this work applies to RGB images only as this is the most common and default color model used to represent and display images in electronic systems. The input frame cannot be binarized directly from the color image so it should be converted to a one-channel grayscale model (Figure 3.3.3.). This can be obtained by splitting the original image into 3 single 8-bit color channels (Red, Green, Blue). Having these obtained the grayscale 1-channel image can be computed using the perceptually weighted formula for each pixel[7]:

$$Y = (0.299)R + (0.587)G + (0.114)B \quad (2)$$

As a result 1-channel 8-bit image is obtained (each pixel represents the luminosity value in range of 0-255). This method is supported by a `cvCvtColor()` function which converts input image's color space into another. Alternatively, there is a possibility to use each 8-bit channel to generate 3 binarized images which should be processed later. This method however is used for color detection. For the simple contour detection only one image is necessary and simultaneous 3-channel processing would significantly decrease the overall project performance. As it is essential to compute the final output image in real-time the first method was selected. The obtained image should be binarized then.

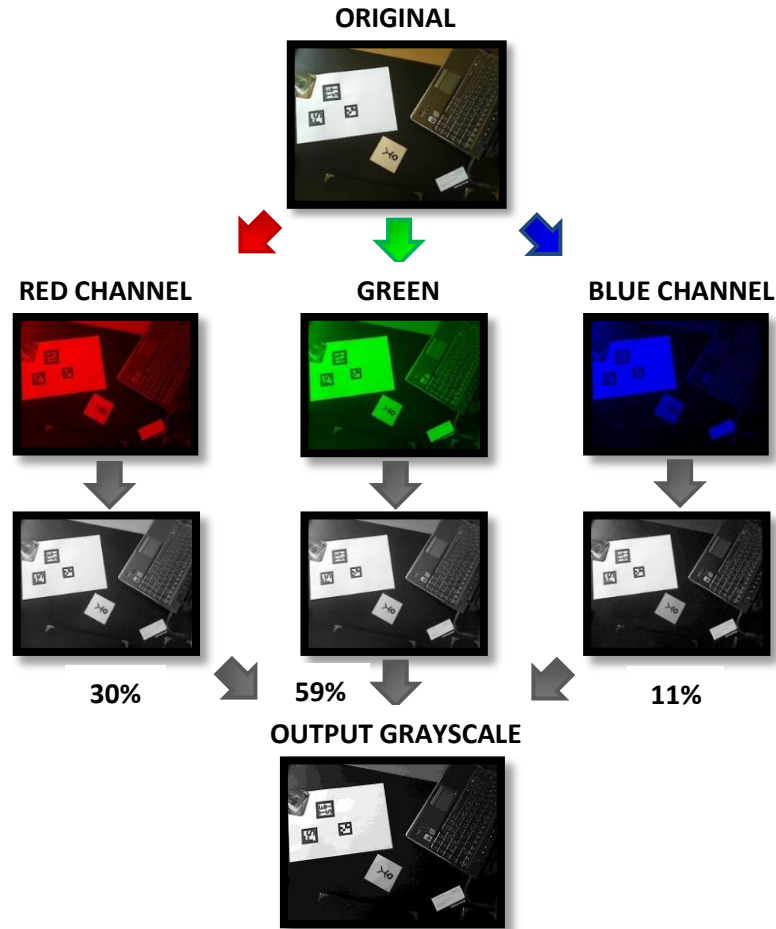


Figure 3.3.3. Grayscale model image acquisition diagram

The main idea of this technique assumes that each processed pixel of the input image should be set to max or min value (0 and 255 have been chosen for this project to provide the best contrast) depending if its luminosity level is above or below the specified threshold value. A simple thresholding algorithm was insufficient for this project purposes as the environment light sources can change decreasing the marker contour visibility as well as the emerging reflections can vastly decrease the detection accuracy. The main difference between traditional threshold and adaptive threshold method is that the latter one computes the new pixel value basing not only on a single pixel but also the neighbourhood pixels inside the specified region (with computed pixel coordinates as a center) so the threshold value is different for different regions in the image. The Gaussian method has been chosen as it gave the most accurate and clear results. It implements the idea that the pixels in the region around (x, y) are weighted according to a Gaussian function of their distance from that center point. This specific binarization process is supported by `cvAdaptiveThreshold()` function which was used in this project.

Enhancements:

As the simple adaptive threshold method occurred to be not sufficient and the binarized image could not be used as a final source for the later computations the major changes had to be applied to the algorithm. First of all the binarization step have proven to be the most essential part for marker contours detection so the improvements had to be focused on the square detection rather than the whole marker extraction. The image binarization process has been divided into two phases. The first one corresponds to accurate contours exposition and is applied in this step (Figure 3.3.4.). The second one focuses on the extracted and cropped square binarization to provide the most accurate results in the matching step (See 5. of this algorithm analysis). To improve the implemented adaptive thresholding method the marker tracking elements have been introduced. The main idea of this improvement assumes that the processed image dynamic threshold block size parameter fully depends on the previously computed marker bounding box size. In this way, if the previously found marker is bigger (closer to the camera device) the block size proportionally increases- the contours are thicker and easier to find. It is similar to the marker tracking methods which set the ROI area near the previously found marker to increase the algorithm performance.

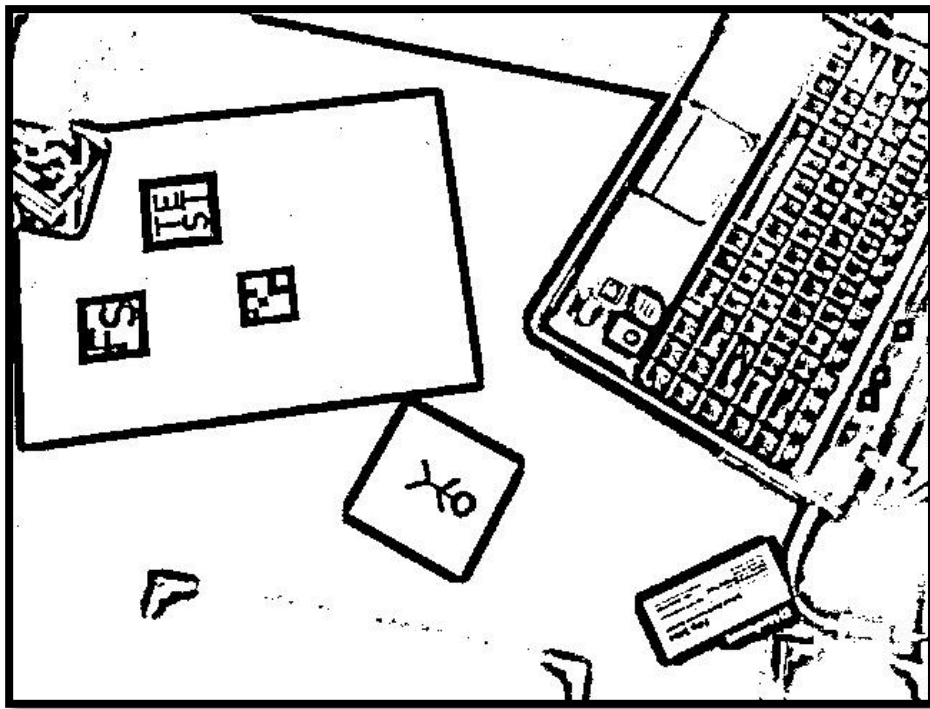


Figure 3.3.4. Binarized image example

3.3.3. Contours finding and squares detection.

Selected square detection method is based on contours finding. To extract the contours sequences from the input binary image the `cvFindContours()` function is used. Its main advantage is that it compresses horizontal, vertical, and diagonal contours segments, leaving only their ending points. Having these coordinates, the simple shapes can be computed by means of polygon approximation algorithms. Based on a simple Freeman chain approximation[14] method (Figure 3.3.5.) the shape is represented as a sequence of steps in one of eight directions designated by an integer from 0 to 7. To convert the Freeman chains to polygonal representation the `cvApproxPoly()` is utilized as the next step. Having the contours ending points evaluated earlier it returns the sequence of straight contours which create an approximated polygon. As the desired result is the quadrangle the additional requirements have to be satisfied:

- sides (contours) number equal to 4
- convex polygon found
- polygon area bigger than specified value (as the marker size should be limited to a minimal value)

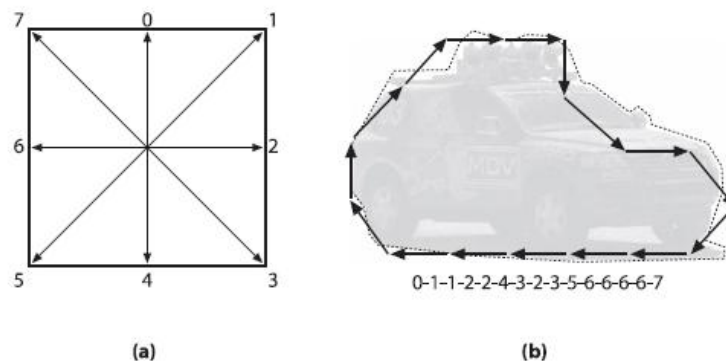


Figure 3.3.5. (a) Freeman chain moves are numbered 0–7;
(b) contour converted to a Freeman chain-code representation[7]

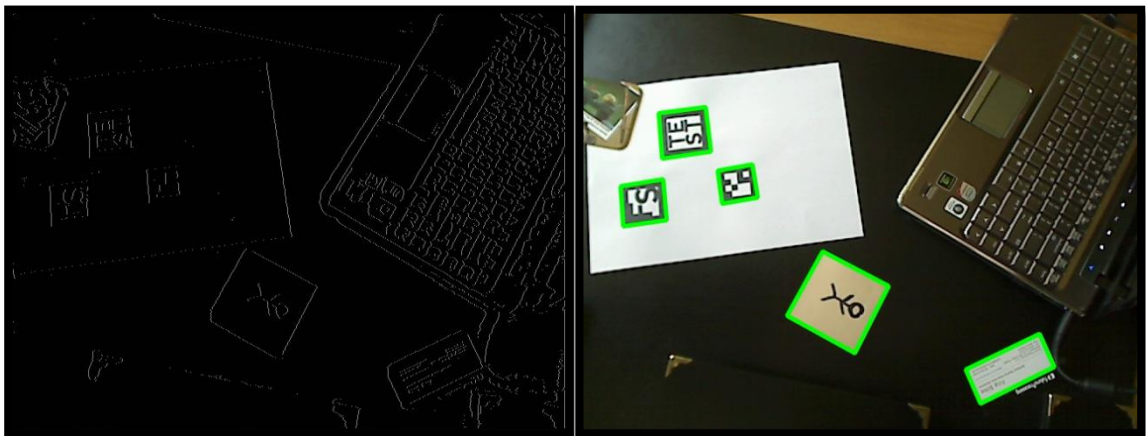


Figure 3.3.6. Obtained contours and detected squares visualization based on the binary image

Having the result quadrangle contours sequences evaluated the corners coordinates can be evaluated by means of using OpenCV sequence reader. As this step computes every quadrangle coordinates and do not affect the performance visibly - there is no need to enhance this step implementation. Each quadrangle side saved as a single contour can be colored to obtain the found objects visualization.

3.3.4. Squares extraction and perspective transformation.

Having in mind the assumption that the marker is a 2D square the obtained quadrangles should be considered as simple squares seen from different perspectives. Having this assumption in mind the evaluated quadrangles have to be transformed according to their perspective to the template-sized square (this project uses 100 x 100px square template image). First of all the set of quadrangle coordinates are read using `CV_READ_SEQ_ELEM()`. Another set of coordinates should be computed from template image as the matching reference (in this project these are coordinates of the 100 x 100px square corners). Having two complete sets of 4 corners coordinates the map matrix used for perspective transformation should be evaluated. It is done by means of `GetPerspectiveTransform()` function which calculates 3 x 3 matrix such that:

$$\begin{pmatrix} t_i \cdot x'_i & t_i \cdot y'_i & t_i \end{pmatrix}^T = map_{matrix} \cdot \begin{pmatrix} x_i & y_i & 1 \end{pmatrix}^T \quad (2)$$

where $dst(i) = (x'_i, y'_i), src(i) = (x_i, y_i), i = 0 \dots$

Then the `cvWarpPerspective()` that uses source, destination image and the map matrix as the arguments should be called to project the resulting, transformed image ready to be matched with the template.

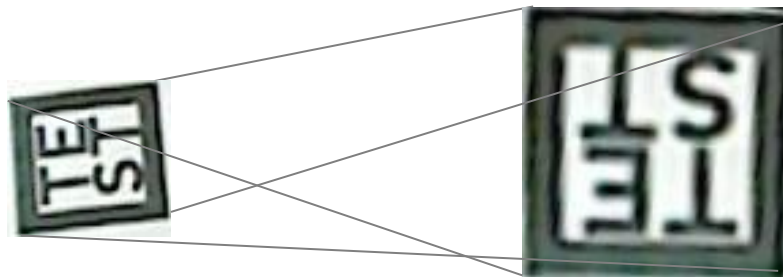


Figure 3.3.7. Perspective transformation and cropping example

Enhancements:

Executing the perspective transformation on whole input image has a direct impact on the whole algorithm performance. The speed can be significantly decreased as the camera capturing resolution can be very high. The solution for this issue is to process only small regions of the input image which has a real influence on the result. Having the quadrangles corners computed, the bounding box has to be evaluated based on the minimal and maximal corner x and y values. The bounding box coordinates are used to define proper ROI which acts as a region that needs to be cropped. Introducing this method greatly increases the overall performance as only found quadrangles are transformed leaving whole input image unchanged.

3.3.5. Template matching.

Correct marker recognition process is the essential step of this algorithm as its effectiveness has a major influence on the virtual object visualization - improper marker recognition can provide to incorrect placed rendered object and a lack of ability to discern the original marker amongst multiple different ones. Basically not only the correct marker should be found but also its rotation angle. To achieve this goal, the template image should be rotated by 90°, 180° and 280°. Then a rotation value has to be set for each pattern to obtain four possible rotation positions of the found square. Then every detected square has to be compared with four templates pixel-by-pixel and the one with the highest matching value should be chosen as the correct marker and the corresponding rotation value would define the rotation angle of the rendered object (Figure 3.3.8.). The matching procedure is performed with `cvMatchTemplate()` function basing on normalized square difference matching method giving the opportunity to compute a reliable similarity percentage value. It matches a template relative to its mean against the image relative to its mean, so a perfect match will be 1 and a perfect mismatch will be -1. A value of 0 simply means that there is no correlation. [7] Additional requirement of minimal similarity ratio should be satisfied to ensure that the rendered object would be superimposed only if the correct marker is found.

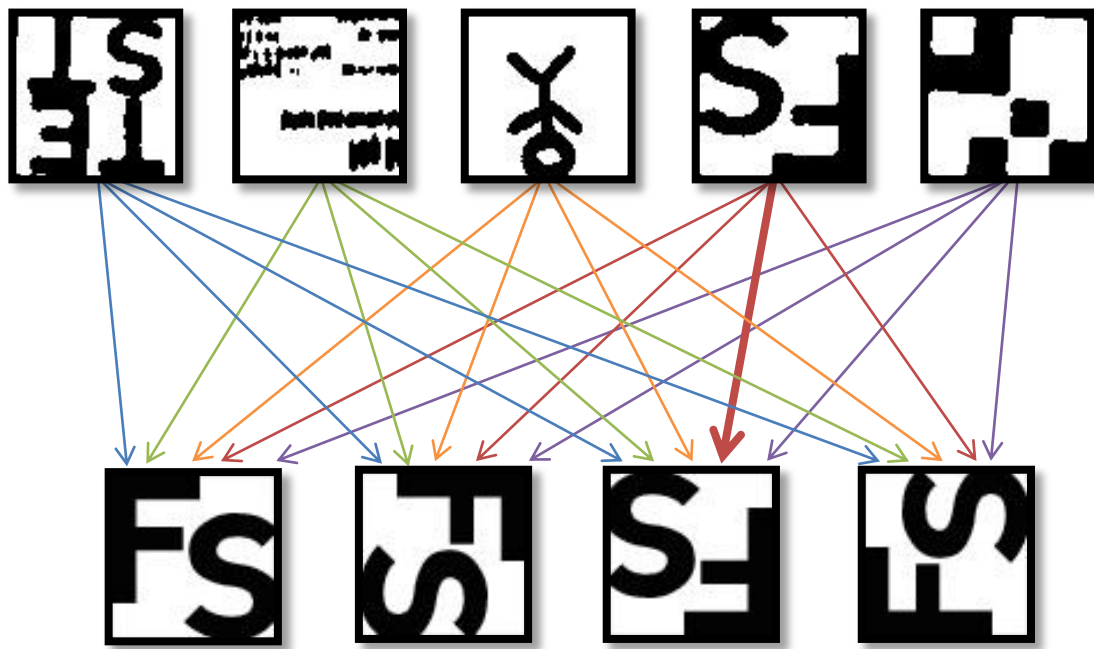


Figure 3.3.8. Exemplary template matching procedure diagram

	-0.03	0.05	-0.07	0.38	0.12
	-0.09	-0.04	0.07	-0.08	-0.11
	-0.12	-0.02	0.06	0.94	-0.02
	0.02	-0.10	0.01	-0.08	-0.12

Table 3.3.1. Exemplary template matching procedure results table

Enhancements:

Two major improvements have been added to this step increasing the marker comparison accuracy. The first one uses the knowledge that each marker has a black frame with fixed size (this can be set in this project algorithm as a parameter). Each of the extracted squares is cropped by means of setting the ROI as the region inside the frame. The same procedure is applied for the template image in the

initialization phase. This way the frame region which is exactly the same is not compared so the performance and accuracy of template matching are much higher. The second enhancement is connected with the binarization step. Originally the extracted image should be taken directly from thresholded input image. However as it occurred during the tests the obtained image not always is accurate enough (first binarization was contour-oriented). To solve this issue the extraction of the quadrangles is performed on the original RGB input image and after transformations and setting the final ROI the threshold filter is applied. As the result image has a fixed size (in this project 70x70px), the block size parameter can be predefined vastly improving the result image quality.

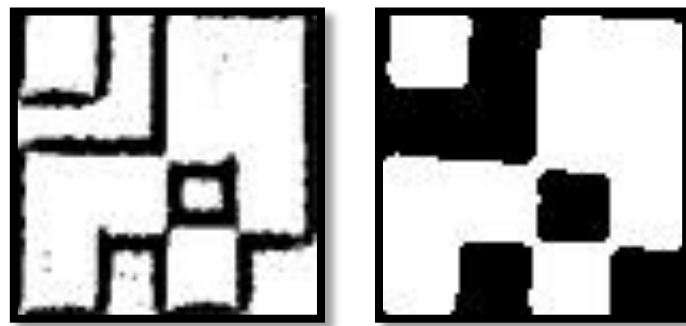


Figure 3.3.9. Comparison of the original method and the enhanced thresholding used for square extraction.

3.3.6. Render virtual object.

In order to superimpose the specified image onto captured frame the perspective transformation should be applied once again. This time, however, the situation is opposite as the image of the fixed template's size has to be fitted into boundaries evaluated in corner detection step. Having the transformed shape the only thing need to be done is to display it in front of the captured camera image. To achieve this goal the exact shape of the quadrangle is copied and filled with black (luminosity value=0). Then it is applied to original image with simple `cvAnd()` function. Finally by means of `cvOr()` function the processed virtual object image is pasted into the blank area and the final merged output image can be displayed on screen with the desired Augmented

Reality effect (Figure 3.3.10.). Analogously the same procedure should be applied for displaying a streamed video file. The only difference that should be implemented is the virtual image source change for each cycle of this algorithm. As the display process does not decrease the overall performance level there is no need to enhance it with further improvements. Changes can be applied only for GUI mode as the UI thread has to be synchronized with the main algorithm thread, hence the displayed image should be either buffered or precisely synchronized with algorithm cycles.

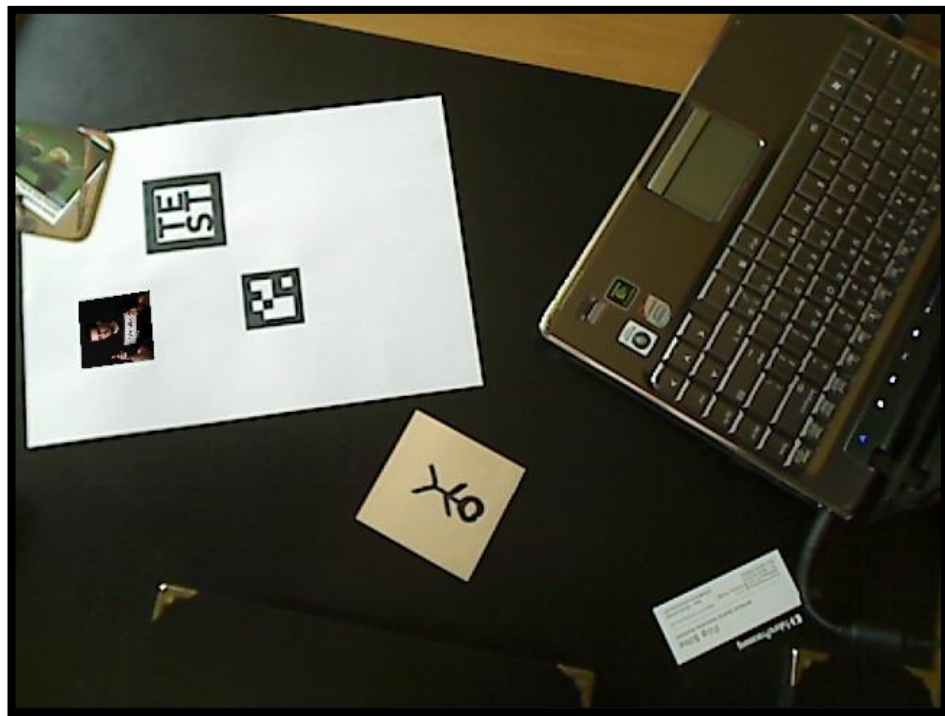


Figure 3.3.10. Final output image with superimposed virtual element.

3.4. GUI

Application graphical user interface (GUI) was created only for user convenience purposes as the raw AR algorithm supports every single area of application requirements. Taking it into account, it was designed as a simple user interface focused on real-time interaction functionality rather than aesthetics. Bearing in mind the future algorithm extensions and further graphical design features the Windows Presentation Foundation (WPF) have been chosen as a GUI development environment. This code part written in .Net was designed to split the project into two synchronized threads. The first one handles the direct AR algorithm represented by the squareLib library. Second thread is responsible for displaying the UI elements. The View-Model-ViewModel design pattern (MVVM) has been used as it is considered to be the most suitable solution for WPF applications[17]. The View element is responsible for presentation and rendering of the graphical elements and their properties. The Model defines the main logic and real state content. ViewModel though is used for the data management and communication between the both elements by means of exposing the data objects from Model, properties bound to View elements and passing the commands from View to Model. This way the UI elements are easy to configure and ready for any future redesign linked directly with the original Augmented Reality algorithm.

Chapter 4

Internal Specification

4.1. Main program functions

Although the step-by-step algorithm has been explained in the previous chapter the direct overall internal specification still needs to be presented. To fully understand the algorithm internal structure only the main functions used in project should be introduced. As some of them has the common application area they should be divided into subgroups focused on defined area of usage.

External- main algorithm functions used externally:

int Initialize()

Initializing function that responds for creating display windows (for debug mode) loading the marker template, displayed image and initializing the video streaming and camera capture and parameters. Returns the integer values corresponding to the initialization result: 0-success, 1-streaming error, 2-image load error.

void Main()

Main algorithm function which calls every step function to perform full AR procedure cycle. It also is used to switch the display modes between image and video.

bool Finalize()

Called when the application has to be closed and the algorithm cycles should be terminated. It clears the memory storage, terminates the video and camera streaming and the display windows created for debug modes. Basically it finalizes the SquaresLib working process.

Internal- main algorithm functions that create external ones:

void LoadTemplates()

Supports the initialization procedure as it loads the display and marker template images and saves them as the corresponding global IplImages.

void Rotate(IplImage * input, float angle, IplImage * out, CvSize SizeRotated)

Used for the input image rotation- supports initialization and display process as the template needs to be rotated by 90°,180°,270° angles and the rendered image should be rotated according to matched template. Gets the input image, defined rotation angle and size of rotated output image. Using the evaluated rotation matrix it performs the rotation transformation and saves the processed image into the output image.

IplImage* GetFrame(), IplImage* GetVideoFrame()

Using the initialized earlier CvCapture structure these functions returns single frame (directly from the camera or from the video file saved on disk) as IplImage that can be processed by the utilized OpenCV functions.

void Threshold(IplImage * input)

Responds for the binarization of the input camera frame. Initializes the gray, thresh IplImages needed for the binary image computation.

The input image is converted to grayscale and then depending on the bounding box size value detected in previous frame the proper input bounding box/frame size ratio is evaluated. Finally according to this ratio correct block size is set for adaptive Gaussian threshold filter. Created binary image is stored in global thresh IplImage.

void FindContours(IplImage * input, int approx)

Computes the overall contour finding from the binarized image. Using the polygon approximation with the input accuracy parameter it performs the quadrangle check (number of sides, polygon area and shape convexity). Finally each four found CvPoint corner coordinates are saved as a sequence in CvSeq squares.

void DetectSquares(IplImage* input, CvSeq* squares)

This function responds for the detected quadrangles preparation for the matching process. It read the corner coordinates from the CvSeq squares and from the template image and evaluates the warp matrix.

Having it the shape bounding box parameters are calculated and the perspective transformation is performed on the cropped image. The

result is saved into template sized image, binarized and then can be matched with marker template. The best fit is saved as the recognized marker along with the rotation parameter and the evaluated bounding box size can be used in thresholding procedure for the next frame as the parameter.

void Crop(CvPoint2D32f* corners, CvRect* bbox)

Used to evaluate bounding box parameters and crop the selected shape from the input image.

double MatchMarkers(IplImage* input)

Match the input processed image with 4 marker templates (original, and 3 rotated ones). The best fit is returned and the corresponding rotation parameter is saved.

void VisualizeSquares(IplImage* input,int angle, IplImage* display)

Basing on the rotation parameter and the template matching result it decides whether and how the display image should be superimposed into original captured frame.

void Visualize(IplImage* input, IplImage* copyinput2, CvPoint corners[4])

According to read input and destination corners evaluates the warp matrix. Having it calculated the perspective transformation is performed and the fitted image is superimposed into the original camera frame

void calcFPS()

Using the timer and counter it evaluates FPS ratio updated every second.

UI - external functions used only for Graphical User Interface purposes:

char* GetImg()

Returns the finally processed frame image data to be displayed within GUI window.

int Width(), int Height(), int Stride()

Used to get the captured image frame width, height, stride parameters to display the output image with correct proportions.

void SetImg(char* path), void SetVid(char* path)

Sets the display image or video streaming files location according to the input path.

void SetMark(char* path)

Sets the new marker template image location and performs the template loading procedure (rotation and cropping)

void TogMode()

Toggles the display mode between the static image and video streaming.

void TogVideoPause(), void RewVideo()

Video controllers that are used to manage the displayed video file. Action that can be performed are pausing/starting and rewinding the superimposed video respectively.

4.2. Parameters

The application performance and algorithm improvements can be easily configured by a global parameters. They were created to simplify the testing analysis process as well as support the debug mode visualization for testing purposes. This way the various markers and environments could be tested and the algorithm is kept extendible with new features.

bool displayMode

Used to toggle between the displays modes. Value 0 stands for the image and 1 stands for the video streaming.

int accuracy

Responds for the matching accuracy level that needs to be satisfied to recognize the correct marker within the found squares.

int count

Defines the number of found polygon sides. For this project purposes it is set to 4 as the quadrangles need to be found, however it creates the possibility to extend this algorithm.

int templateFrame

Sets marker template frame thickness used to crop the variable part of the marker that need to be matched. It was implemented as a parameter for test purposes mainly as the different libraries use different 2D markers.

int markerSide

Sets the overall marker template side size value. It creates the possibility of using any square template image.

4.3. Algorithm block diagram

Simplified algorithm cycle block diagram should be presented for better problem understanding.

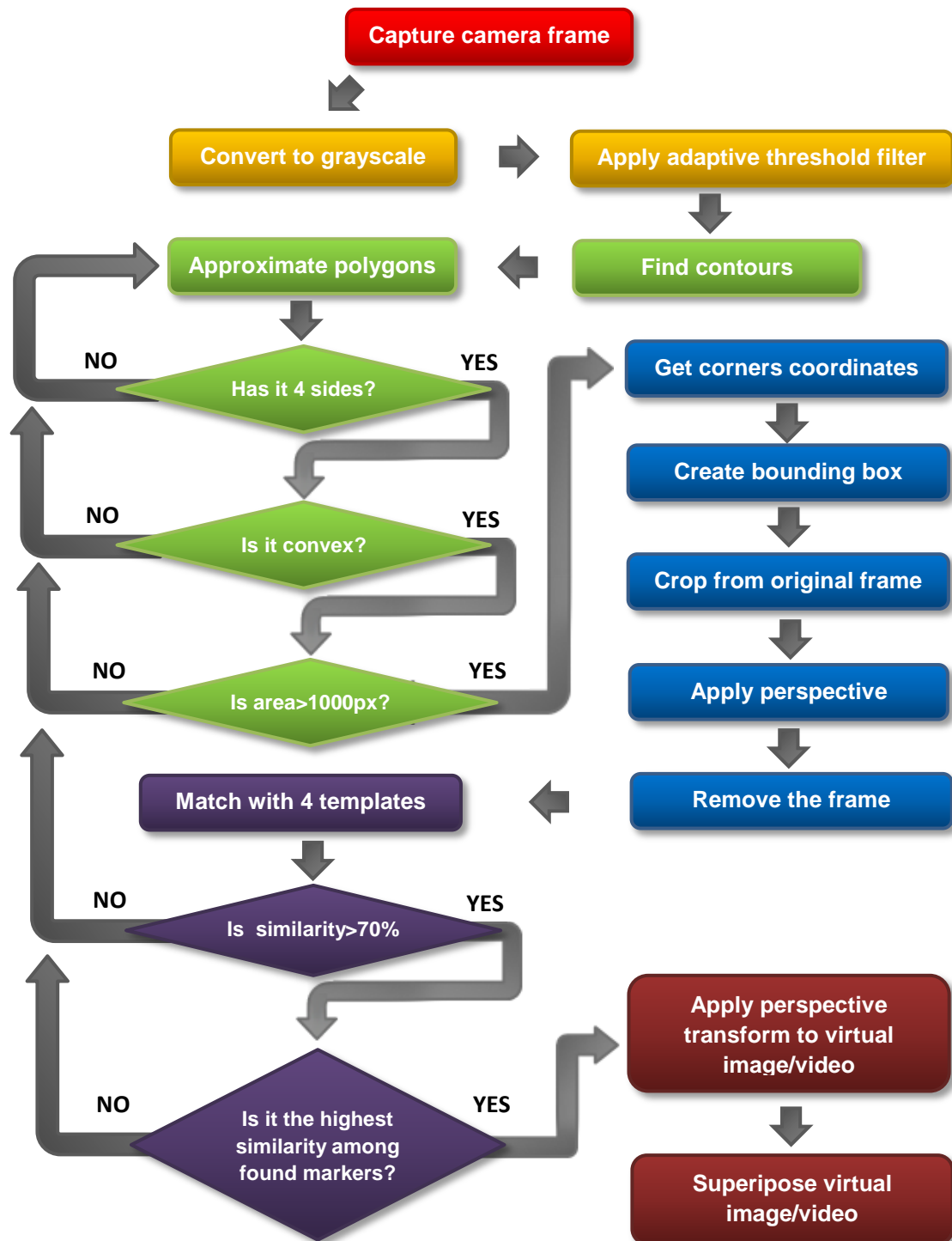


Figure 4.1.1 Simplified algorithm block diagram.







-  **GetImg()** - capturing the camera frame
-  **Threshold()** - binarizing the camera frame
-  **FindContours()** - finding squares
-  **DetectSquares()** - transforming squares
-  **MatchMarkers()** - marker-template matching
-  **Visualize()** - superimposing virtual image/video

Figure 4.1.2. Algorithm block diagram's color legend.

Chapter 5

External Specification

5.1. 'How to' instruction

To use this application the camera device and a printed marker template are required (default marker pattern in .pdf can be found in 'Images' folder). After plugging in the camera device and starting the application the simple graphical user interface will appear.



Figure 5.1.1. Final output image with superimposed virtual element.

Image captured from the camera is displayed and the application is ready to use. To observe the interactive Augmented Reality effect the user has to bring the printed marker into the area of camera's view. The virtual image would be superimposed automatically and can be observed in real-time.

There are several additional option buttons placed on the window top menu which allows the user to influence the displayed objects:

Display Mode:

Image/Video (1): This option allows to toggle the display modes. The superimposed object can be switched between the static image and streamed video file.

Browse:

Image (2): Responds for changing the displayed image to the one selected from the open file dialog window. The choice is narrowed to the most common image files: .jpg, .png, .bmp.

Video (3): Responds for changing the displayed video file to the one selected from the open file dialog window. The choice is narrowed to the most common video files: .avi, .mpeg, .mov.

Template (4): Responds for changing the marker template file to the one selected from the open file dialog window (the new custom marker pattern can be created according to the example MarkerTemplate.psd which can be found in Images folder). The choice is narrowed to the most common image files: .jpg, .png, .bmp.

Video Control:

Rewind (5): Allows to rewind the displayed video file.

Pause/Start (6): Toggles between pausing and resuming the displayed video file.

5.2. Errors

The most error vulnerable step of the application is the initialization process as the project has to use square library .dll and get the default images and video data. There is a possibility that user will not build the project correctly or delete the default images provided for this application. To handle this issues and support the user with proper information the detailed issue description are displayed in a message box so the user can fix the problem and start the application as it is designed. The possibility of changing template patterns, displayed images and videos introduces file types error, the extension choice possibilities were filtered to the most common ones related to the desired file types.

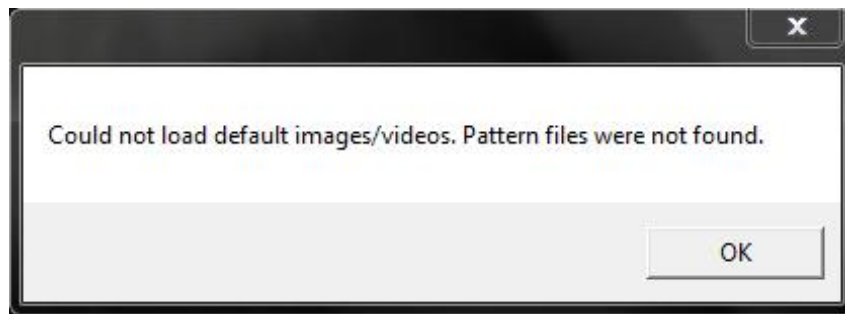


Figure 5.2.1. Example of the error message box.

Chapter 6

Testing and results analysis

Testing and application debug mode analysis is the essential part of the algorithm design. Various sets of test have been performed and analysed to provide the best quality for this application. Algorithm performance and marker detection accuracy has been greatly improved due to the various marker and environmental condition tests. Each set has been grouped according to the area that it improves and the detailed results analysis can be found in this chapter.

6.1. Marker choice analysis

The marker template itself has a major influence on the detection accuracy, as it has to be recognized to superimpose the virtual object. This project provides almost infinite possibilities of marker templates design limited by the only three basic requirements: template must be black and white, square and bounded by black frame. Having these requirements satisfied the choice of the pattern is bounded only by the user's imagination. However there are still some properties that influence the algorithm performance and recognition accuracy. According to single marker template property the sets of tests were performed and analyzed to define the best solution for the final application version to make it more convenient for the user.

6.1.1. Size.

The template size loaded has a vast impact on the algorithm speed as the bigger template resolution is, the more pixels have to be processed. Smaller images however provide too small amount of data and the marker could be not recognized. The tests were run for 5 default templates of different sizes: 25x25, 50x50, 100x100, 250x250, 500x500. Each test

timespan was equal to 10 seconds. Marker average recognition accuracy was measured as an average marker-template similarity ratio presented as percentage value. The mean average has been evaluated for the results obtained within one second period and presented in table (Table 6.1.1.) Tests were performed for standard real-time motion (horizontal 45° angle rotation and zooming in and out). The algorithm performance was measured by the FPS parameter (Table 6.1.2.). Test results were presented also in graph form for better visualization.

Average recognition accuracy(%)					
	25x25	50x50	100x100	250x250	500x500
1	50.36	86.77	89.67	90.61	93.21
2	52.92	85.67	89.44	87.93	91.55
3	46.88	71.08	84.90	92.64	91.86
4	39.29	84.01	85.49	86.11	92.78
5	37.24	82.57	84.59	85.18	93.79
6	47.34	78.44	91.36	92.33	95.58
7	44.13	77.49	92.72	94.03	93.89
8	43.70	86.90	93.01	93.29	90.81
9	42.99	84.05	92.54	94.29	84.54
10	43.44	84.05	91.21	94.39	84.30

Table 6.1.1. Recognition accuracy tests results.

FPS rate					
	25x25	50x50	100x100	250x250	500x500
1	17	16	16	7	2
2	16	17	16	10	3
3	17	17	16	7	2
4	17	16	15	7	2
5	16	16	16	9	3
6	17	16	15	7	2
7	16	16	16	6	3
8	17	17	16	7	2
9	17	16	16	7	4
10	17	16	15	7	4

Table 6.1.2. FPS rate tests results.

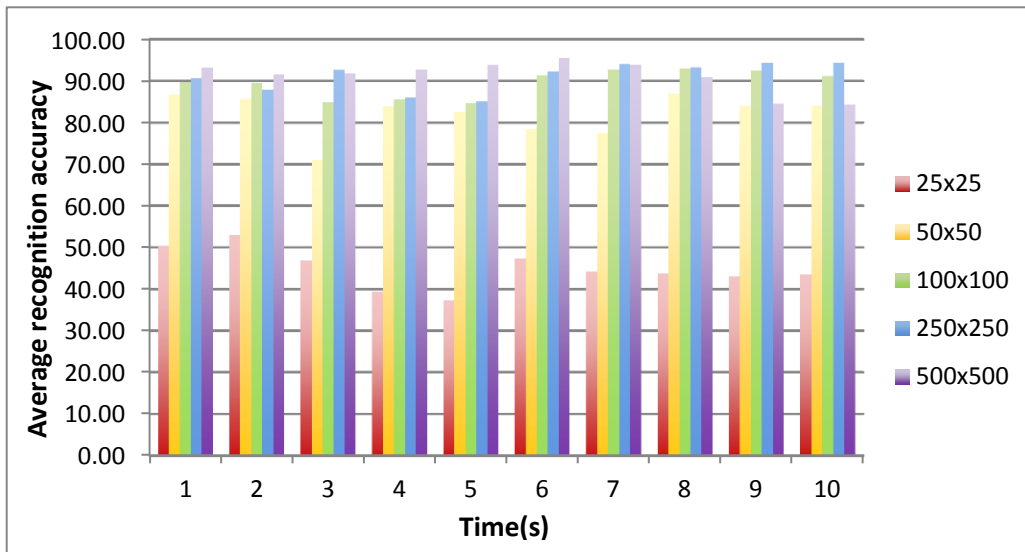


Figure 6.1.1. Recognition accuracy tests results graph.

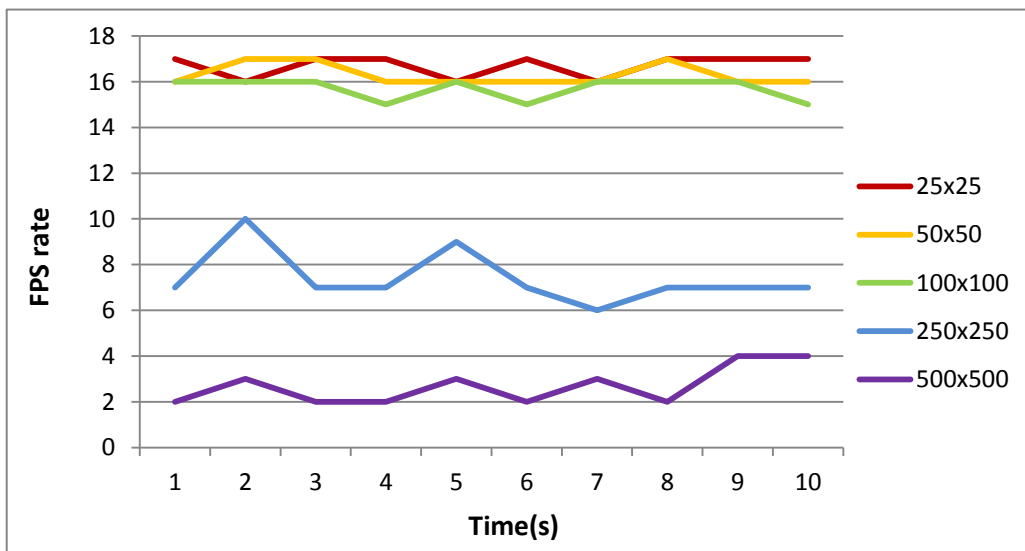


Figure 6.1.2. FPS rate tests results graph.

Analysis:

As it can be observed the only pattern size which cannot be used is the 25x25px marker as the recognition accuracy for it is poor and do not satisfy the requirements. All the others were correctly recognized as the average percentage ranged 75%-95%. The FPS rate is noticeably decreased for 250x250px and 500x500px patterns while the rest supports the satisfactory frame rate. Taking these observations into account the 100x100px marker template provides the best results which do not affect the FPS rate and the recognition accuracy, hence this size is considered to be the default template size that should be loaded by the application.

6.1.2. Content.

As the marker content is meant to be changeable and custom for each user the content recognition check should be performed. The analysis would be helpful to evaluate some guidelines for creating custom marker as there are certain limits and boundaries that should be identified. Every content creation property has been tested individually and the overall recognition test using all the markers on one white sheet of paper has been performed as the last one (see 6.1.4.). All the tests were run in the same environment and included the static marker recognition for different positions.

- Front (up to 10 cm from the camera device)
- Rotation (up to 10 cm from the camera device and angled by 45° horizontally – in respect of Z axis)
- Distance (50cm from the camera device).

The markers were printed on 4x4cm sticky notes and the timespan for each test was equal to 5 seconds. The result values were obtained for each video frame and then the mean average was calculated for one second period.

The applications parameters were set to the default values- the same for each test procedure as they have been found out to be most effective in majority of cases (see next sections of this chapter):

- Loaded template size: 100px, template frame: 15px
- Polygon approximation: 6px
- Marker recognition minimal boundary value: 65%

All the results were presented as results table and two images set. The table consists of percentage value of marker-template similarity ratio. The table rows are corresponding to: minimum (min) and maximum (max) value of the sampled similarity ratio mean averages and the standard deviation of obtained samples (s). The image set represents the original template pattern (bigger image) and the imaged obtained by the algorithm from the captured camera frame (smaller one).

Default camera (Logitech Quickcam Pro 9000) resolution was set to 640x480px as this setting has been proved to be most effective (for more details see section 6.1.3.). It should be realized that the better results for individual cases could be obtained for higher camera resolution or better environment light conditions, however these test cases are based on the default parameters and environment setting.

Text:

First tests were run for the markers which consist of big letters which can be recognized by the human eye on first sight. This case corresponds to user's initials or project abbreviations that could be used. The main task was to state whether the letters are binarized properly to be recognized and differed from each other.

Marker-template similarity matching results

	front	rotation	distance
min	85.09	82.27	75.34
max	88.72	86.18	81.33
s	1.37	1.66	2.13



Figure 6.1.3. Text template recognition results. Ex1.

Marker-template similarity matching results

	front	rotation	distance
min	89.87	76.45	75.49
max	91.20	81.06	84.00
s	0.57	1.76	3.43

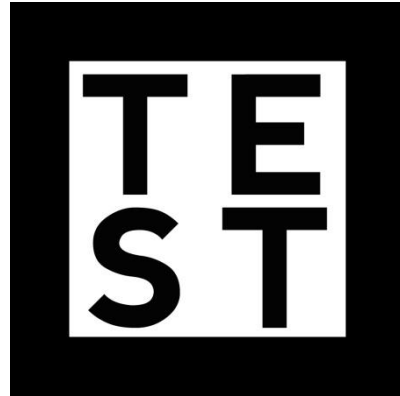


Figure 6.1.4. Text template recognition results. Ex2.

Analysis:

Majority of results ranged 75-90% recognition accuracy and the processed binary images in fact are very similar to the original ones. This proves that simple text markers can be used for this project purposes.

Small fonts:

These tests were performed in order to define the level of markers details. Small font words and even full sentences were printed on the markers. Two markers were very similar to each other and although they consist of different sentences they could not be differed from the distance by the human eye so easily. The additional question that should be asked here is ‘whether the algorithm would recognize the correct marker having both of them in the view area’

Marker-template similarity matching results

	front	rotation	distance
min	64.02	44.55	52.07
max	67.82	49.94	61.48
s	1.40	2.20	3.40



Figure 6.1.5. Small text template recognition results. Ex1.

Marker-template similarity matching results

	front	rotation	distance
min	56.94	44.07	45.85
max	61.58	51.74	54.82
s	2.34	2.79	3.96

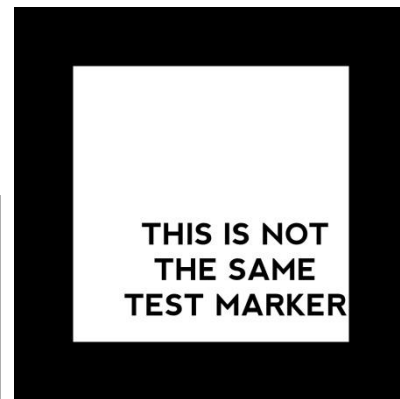


Figure 6.1.6. Small text template recognition results. Ex2.

Analysis:

The results clearly indicate that the small fonts are hardly recognized by the algorithm and should be not used as the marker patterns. Despite of 45-65% recognition range the two markers were correctly matched with the original templates if the minimal market-template boundary value was lowered and no mistakes occurred even when the both of them were placed in the camera device view range.

Matrix:

The most common marker templates used for modern AR libraries are the 2D matrix matrices. They are valued for simplicity and low resolution patterns what greatly increase the algorithm performance giving multiple combination possibilities, however no custom shape can be drawn with them. These tests should prove that the matrix markers can be used for this project purposes and the shapes directly merged with the frame do not decrease the recognition accuracy. All test have been provided for 4 x 4 matrices.

Marker-template similarity matching results

	front	rotation	distance
min	88.78	85.68	83.93
max	89.82	86.60	86.96
s	0.37	0.33	1.31

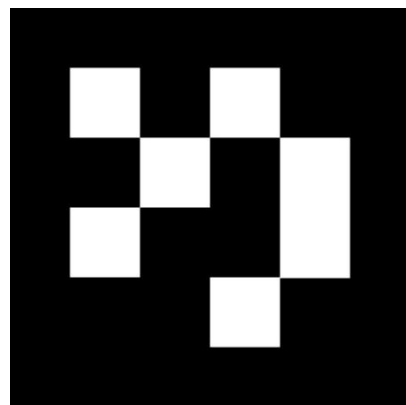
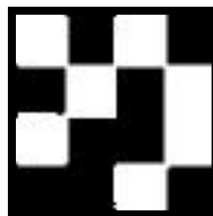


Figure 6.1.7. Matrix template recognition results. Ex1.

Marker-template similarity matching results

	front	rotation	distance
min	91.16	90.07	87.40
max	92.42	91.12	90.87
s	0.46	0.40	1.32

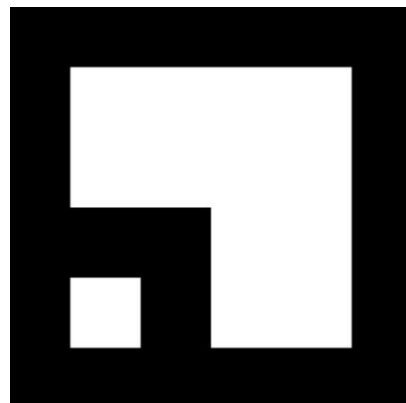
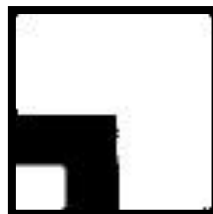


Figure 6.1.8. Matrix template recognition results. Ex2.

Analysis:

Very high recognition rate (85%-95%) for the matrix markers indicate that they are perfect for this project and as expected the merged black squares and frame did not create any problems for the algorithm.

Symmetry:

Symmetry against the square's diagonal is a very extraordinary marker case. In fact even if the marker would be recognized there is no possibility to define the angle it was rotated. Every 180° rotation has no effect on the pattern. These tests were performed on two type of symmetrical markers: 1 and 2 diagonal symmetry. It should prove that these type of markers should not be used even if the recognition accuracy would be high.

Marker-template similarity matching results

	front	rotation	distance
min	86.61	83.63	80.57
max	89.76	86.15	85.61
s	1.34	1.00	2.02

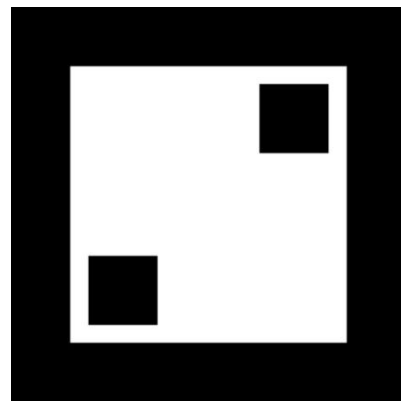
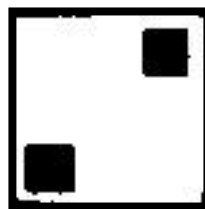


Figure 6.1.9. Symmetry template recognition results. Ex1.

Marker-template similarity matching results

	front	rotation	distance
min	95.06	95.06	91.11
max	96.30	95.96	93.34
s	0.47	0.37	0.93

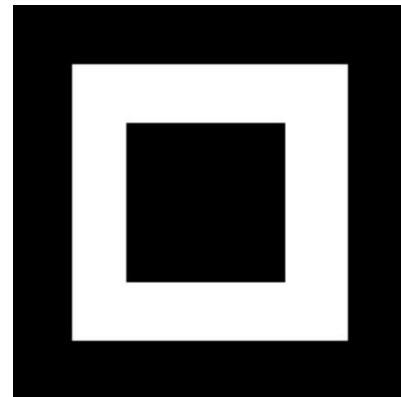


Figure 6.1.10. Symmetry template recognition results. Ex2.

Analysis:

Despite of very high recognition rate (80-95%) the markers should not be used for this project. Marker patterns are recognized perfectly, however there is no possibility to define the rotation parameter so the displayed image flickers between original and upside-down position.

Custom images:

The main advantage of this work solution is the custom marker creation possibility. Every non-symmetric shape can be used as a pattern, however the higher detail level the less accurate the recognition should be provided. These tests were run to define whether the real custom made shapes can be utilized and how do they influence on the algorithm performance. Two highly detailed shapes were printed on the sticky notes to evaluate the proper results.

Marker-template similarity matching results

	front	rotation	distance
min	72.25	77.12	60.93
max	75.27	79.11	69.26
s	1.26	0.71	3.31

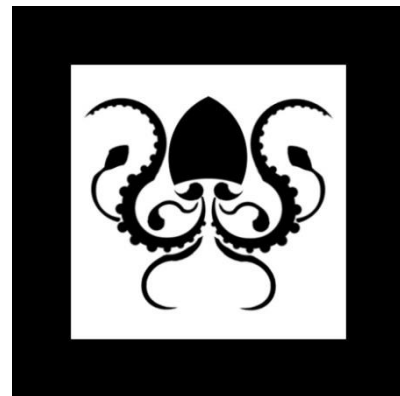


Figure 6.1.11. Custom template recognition results. Ex1.

Marker-template similarity matching results

	front	rotation	distance
min	86.17	79.98	80.46
max	87.16	81.26	84.13
s	0.41	0.53	1.43

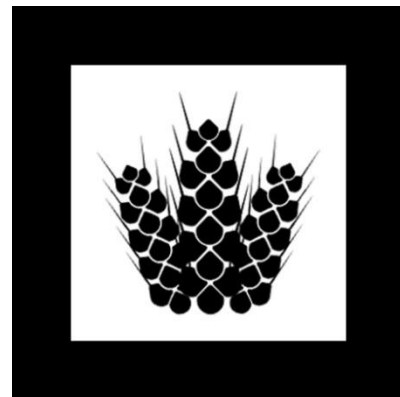


Figure 6.1.12. Custom template recognition results. Ex2.

Analysis:

Test results observation shows that the highly detailed objects are simplified as the very thin lines or holes are flood-filled or removed. However, it can be clearly noticed that medium recognition ranges (60-85%) allows to use them as template patterns. As the objects are

initially simplified there are no big differences in recognition level for different angles or distances. Seeing the advantages and disadvantages of these templates it is not recommended to use them as proper markers.

Line thickness:

The last tests were performed to observe how the shape's outline thickness influence the recognition process. It would help to design the guidelines how thick the line should be to correctly recognize the marker and do not mistake it with another. The leaf-shaped patterns were chosen and the 2px and 12px outlines were applied.

Marker-template similarity matching results

	front	rotation	distance
min	42.41	32.07	3.33
max	43.70	37.66	5.84
s	0.52	2.23	1.18

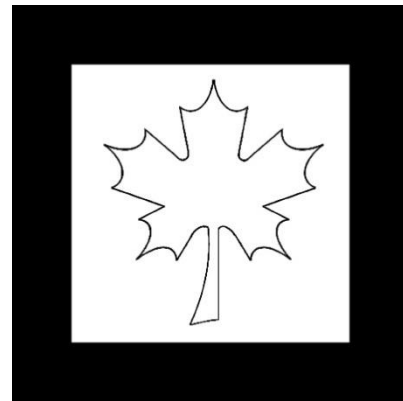


Figure 6.1.13. Pattern line recognition results. Ex1.

Marker-template similarity matching results

	front	rotation	distance
min	87.23	68.96	74.40
max	89.27	76.35	81.16
s	0.85	2.84	3.13

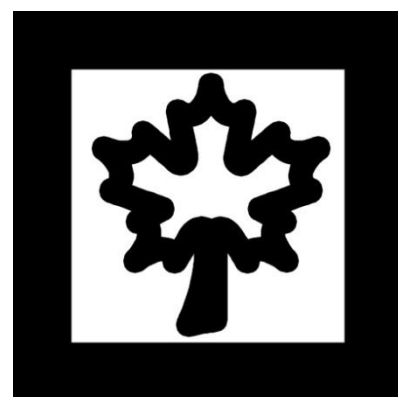


Figure 6.1.14. Pattern line recognition results. Ex2.

Analysis:

The result for thin line were very poor and the marker was not recognized. Finally- the outline should not be thinner than 4px.

6.1.3. Frame thickness.

The marker black frame thickness has big influence on the square finding accuracy. Depending on the thickness and the threshold filter applied the frame can be not detected as the square sides can vanish if the frame is too thin or it can deform its content when the frame is too wide. The most commonly used marker template in AR libraries has the frame thickness equal to a quarter of marker width. That is because most of libraries uses 16x16 matrices rather than image content and the frame thickness does not influence the shape inside as much as in this project. However the thickness tests were performed to choose the best fit for marker recognition especially for finding the angled markers. Tested markers frame thickness was equal to: 5px, 15px and 25px. Each of the printed markers was tested for 3 cases:

- Front (up to 10 cm from the camera device)
- Rotation (marker angled by 75° horizontally – in respect of Z axis)
- Distance (up to 50 cm from the camera device)

As the detailed recognition results are not needed this time only the camera processed images would be presented to get the general detection idea (Figure 6.1.15.).

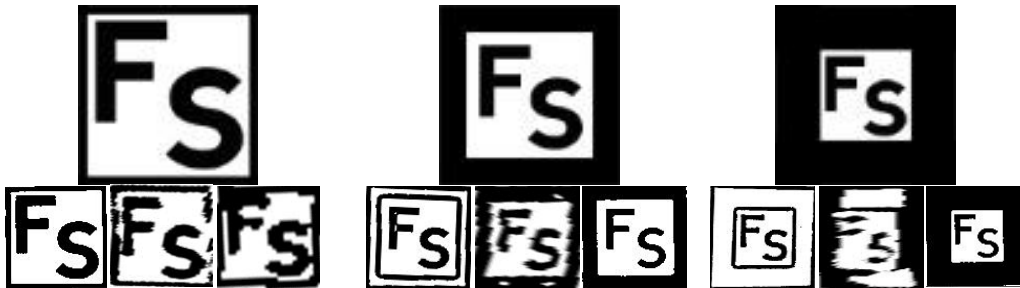


Figure 6.1.15. Marker frame thickness test results.

Analysis:

Analyzing the tests results (Figure 6.1.15.) it can be observed that the thin 5px frame could not be detected if rotated by more than 60° as the sides were vanishing and the marker had to be extracted by hand. The thick 25px frame was always detected, however it can be noticed that for big angles the binarization affected by the frame is not accurate enough for its content and the distant marker is affected by the reflections. The best detection features were observed for the 15px frame which only distorts the marker content even for angles wider than 60°. That is the reason why the 15px frame was selected as the default one.

6.1.4. Minimal recognition level parameter.

To obtain most reliable marker recognition the minimal similarity ratio should be tested and defined. If this parameter is set to low value there exist very high probability of false positive detection – images which are not correlated with the original template are recognized as a desired marker and the simulation is superimposed on false object eventually. On the other hand if this parameter is set to high value there is a risk that the correct marker would not be recognized as the camera device or transformations of angled marker causes some distortions and the detected marker never gets 100% similarity ratio. Then no simulated object would be superimposed on the captured frame even if the proper marker is placed in the range of camera view.

Tests were done for two most common and opposite light conditions:

- Normal (daylight)
- Dark (dark room with no light source)

Each tests set was performed for the following minimal similarity parameter values: 0%,20%,40%,50%,60%,65%,70%,75%,80%,90%,100%. Default marker was used as proved to be suitable for recognition tests. Logitech Quickcam Pro 9000 was chosen as used camera device. Each test set was performed according to the following sequence:

- Single - a single marker has been placed 50cm from the camera and the standard motion has been tested (front, 45° rotation in regard to every 3D plane axis, distance). The test result can be set as 'correct' if the marker has been recognized correctly for every position in standard motion procedure. If the marker has not been recognized in any motion procedure sequence – status 'incorrect' would be set.
- Multiple - multiple markers (Figure 6.1.16.) have been placed 50cm from the camera and standard motion procedure has been performed. Status 'correct' should be set if the proper marker is found. If marker is not found or the improper one is recognized as the template – status is set to 'incorrect'.
- FP - multiple markers have been placed 50cm from the camera and the correct one has been covered with white piece of paper to test the false positives occurrence. If no false positives presence occurred – the status is set to 'correct'. Otherwise it is set to 'incorrect'.

**Minimal recognition parameter tests results
(normal light conditions)**

	Single	Multiple	FP
0%	correct	correct	incorrect
20%	correct	correct	incorrect
40%	correct	correct	incorrect
50%	correct	correct	incorrect
60%	correct	correct	incorrect
65%	correct	correct	correct
70%	correct	correct	correct
75%	correct	correct	correct
80%	incorrect	incorrect	correct
90%	incorrect	incorrect	correct
100%	incorrect	incorrect	correct

**Minimal recognition parameter tests results
(extremely dark light conditions)**

	Single	Multiple	FP
0%	correct	correct	incorrect
20%	correct	correct	incorrect
40%	correct	correct	incorrect
50%	correct	correct	incorrect
60%	correct	correct	incorrect
65%	correct	correct	incorrect
70%	correct	correct	correct
75%	incorrect	incorrect	correct
80%	incorrect	incorrect	correct
90%	incorrect	incorrect	correct
100%	incorrect	incorrect	correct

Table 6.1.3. Marker frame thickness test results
for normal and extremely dark light conditions.



Figure 6.1.16. Tested markers set.

Analysis:

These tests helped to define the minimum marker-template similarity boundary value that should be set as default. The correct marker recognition without false positives occurrence has been found for 65-75% similarity ratio for casual and extreme light conditions. Studying the results (Table 6.1.3) it can be observed that dark light conditions narrowed the correct recognition range as casted shadows caused some threshold noises for wide marker rotation angles. For higher values the superimposed image flickers as the marker-template similarity can be reduced by distortions and camera frame resolution. Lower values do not influence the effect as long as the correct marker is placed in the camera view range. As soon as it removed the algorithm detects false positives. It should be noticed that tested markers were very similar to make these test more reliable, however for casual use of this application the minimal similarity boundary can be set a little bit lower.

6.1.5 Marker surface.

The next issue that should be considered is the material which the business cards would be printed on. Each paper type or other surface used for the marker presentation has the different reflection properties. Reflected light can create a high noise rate or even change the whole marker luminosity level so the image binarization can fail and the marker detection would not be possible. The tests were run on a different materials and surfaces to determine how the reflected light would affect the thresholded image. The surfaces tested were as follows: highly brushed paper, plain printing paper, glossy photo-paper and the iPhone glossy display. The test visualization consisting of the raw markers extracted from camera frame is presented below.

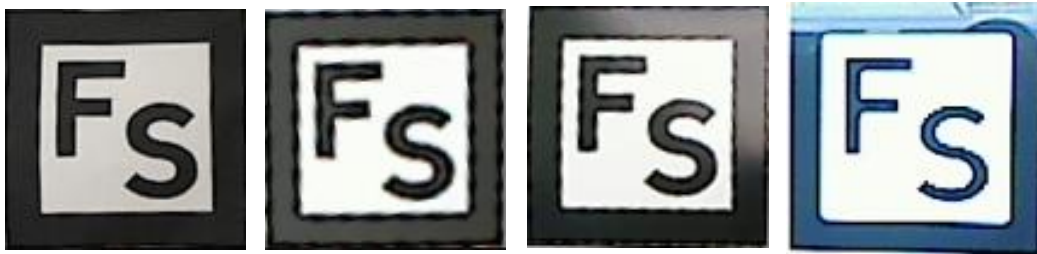


Figure 6.1.17. Marker surface tests visualization. Materials used from right: highly brushed paper, plain printing paper, glossy photo-paper, the iPhone glossy display

As the best results without reflections, noises and distortions can be observed for the highly brushed paper. Moreover, a little distortions can be noticed as the paper sheets are very flexible and the detected quadrangle sides were not perfectly straight. Taking it into account it is highly recommended to use highly mat, thick and inflexible surfaces for the printed markers as the less reflective the material is the more accurate the detection is.

6.1.6. View angle.

The wider the angle range of the detected and recognized marker is, the more realistic Augmented Reality effect is obtained. Taking these words into consideration the marker with wide viewing angle set (marker rotation motion according to the camera device position) should be recognized to provide most reliable simulation as it is possible. Simple recognition tests were performed for this project to verify the angle boundaries of marker recognition. Similar tests were run for two other popular AR libraries (ARToolKit and FLARToolKit) to spot the differences and state which library provides the best view angles recognition algorithm. It should be taken into consideration that these tests were done in the same good environment light conditions to obtain general angle range recognition levels for each library.

Maximum recognizable marker angle

	Z axis	Y axis	X axis
This	360°	70-75°	75-80°
ARToolKit	360°	70-80°	75-80°
FLARToolkit	360°	70-75°	75-80°

Table 6.1.4. Angled marker recognition tests results.

Analysis:

The tests proved that for the best environmental conditions each algorithm provides very similar, reasonable marker view range (70-80°). According to this statement this project has the same level of angled marker detection accuracy as the other common libraries. These results could not be obtained for other environment conditions as the other libraries did not detect the marker properly in bad lighting conditions. The angled marker recognition tests in different environments were also tested for this project further (Chapter 6.2).

6.2 Environment dependencies

The environment the camera device is placed in is another key factor for the square detection procedure. According to the details and objects placed around the marker it could be easier or harder to find. The main issue that should be considered is the environment light sources and their luminosity. According to the light source location and power the several test cases can be distinguished: bright location, casual location, dark location, focused light source, multiple shadows casted on marker (Figure 6.2.1.). To make these tests most reliable the same marker were tested in the exact location under different environmental light conditions. Marker was rotated along 3 axis and the true or false recognition status was denoted for each marker position. Similar tests were run for two other popular AR libraries (ARToolKit and FLARToolKit) to decide which library provides the best marker detection algorithm.

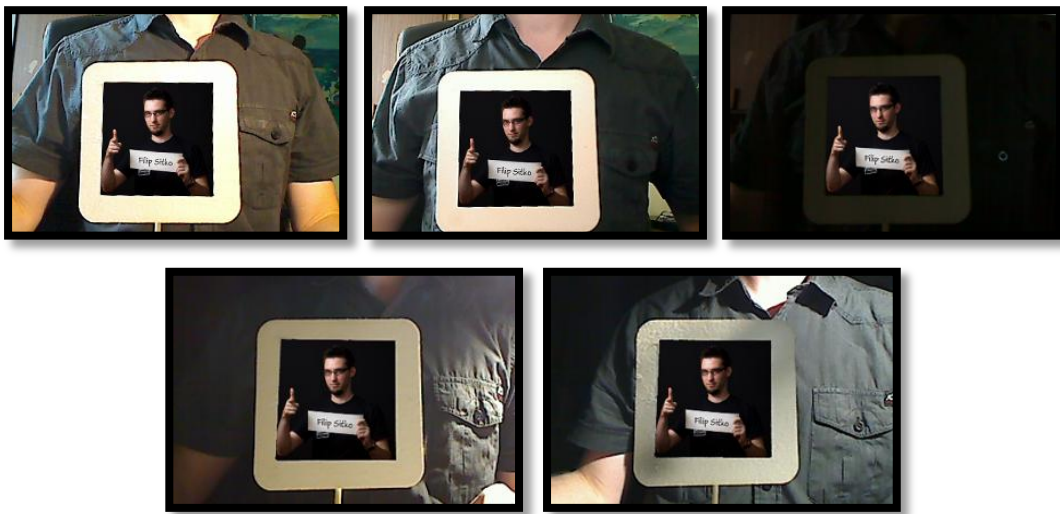


Figure 6.2.1. Environment lighting conditions examples.

Maximum recognizable marker angle

	Z axis	Y axis	X axis
Bright	360°	70-75°	75-80°
Normal	360°	70-75°	75-80°
Dark	360°	65-70°	70-75°
Focused	360°	60-65°	70-75°
Shadows	360°	65-70°	70-75°

Table 6.2.1. Environment lighting conditions tests results.

Analysis:

The tests helped to define the recognition of angled marker range for each type of environment. The focused light test results (Table 6.2.1.) may be different for different light source positions as the main recognition limit was caused by the light reflections appeared on the marker surface during the test procedure. It proves that for markers printed on mat surfaces the angles can be wider even for single focused light source and shadows casted on the marker. Moreover, it occurred that the best pattern detection was designed in this solution as other libraries could not detect the marker properly in adverse light conditions, hence the virtual objects flickered and vanished in the certain positions. This proves that this project's algorithm is unique and can be compared to other common AR open source libraries.

6.3 Camera parameters

Each camera device has its own features and configurable properties which can influence the algorithm performance. As the higher resolution means the bigger amount of pixels to process the fps rates should visibly decrease. Moreover, some camera devices provide the auto focus property which should have impact on the detection and recognition process. Simple tests were run for 3 types of camera device to determine what is the influence of their unique properties on the performance and which resolution should be used as default. Webcams used for test were as follows:

- Notebook integrated HP Webcam
(integrated low-resolution and low FPS webcam)
- Logitech Quickcam Pro 9000
(medium class camera device with standard fps rate)
- Creative Live! Cam Socialize HD 1080
(high-resolution camera device with integrated auto focus)

Each test set was provided for 4 different captured frames resolutions (640x480, 800x600, 1024x768, 1280x960) to determine which of them would be the most suitable for this project purposes.

FPS rate according to chosen frame resolution

	HP	Logitech	Creative
640x480	5 FPS	17 FPS	17 FPS
800x600	Not supported	14 FPS	12 FPS
1024x768	Not supported	6 FPS	10 FPS
1280x960	Not supported	3 FPS	7 FPS

Table 6.3.1. FPS measurements results table.

Analysis:

These tests helped to define the default input image resolution as the 640x480px. It provides recognition results that are good enough with low processing consumption. Another issue has been noticed however- for the webcam with integrated autofocus function the recognition accuracy was very low as the standard camera autofocus software were searching for the user face blurring anything else in the view area.

Chapter 7

Summary

As the new Augmented Reality technologies are delivered for the casual users and mass consumer it is almost definite that it would have a big impact on the almost every area of human life. Having that in mind the unique form of self- presentation can be valued and differ from the competition. This project has been proved to satisfy all the basic Augmented Reality concept requirements keeping the satisfying algorithm performance. Compared with other common libraries it occurred to have some key advantages:

- high reliability
- the best marker recognition in hard lighting conditions
- parameters configuration possibility

These arguments proved it to be perfectly suitable for the business cards interactive presentation. Having great environment adaptive properties it is certainly noteworthy application convenient to use and manage the detection properties. With the growing computer's computational power this solution would turn out more realistic and with a high-class camera device it can create the simulation that is hard to differ from the captured video.

Though the Augmented Reality is a very complicated problem itself, the new toolkits are constantly developed and support wider range of platforms. This business-cards oriented application can be easily extended by the new possibilities and bring the full 3D simulation in future to find its place among the commonly known AR libraries and toolkits.

Bibliography

- [1] Ronald Azuma. "A Survey of Augmented Reality", Presence: Teleoperators and Virtual Environments vol. 6, no. 4, pp. 355-385, August 1997.
- [2] P. Milgram, A. F. Kishino, "Taxonomy of Mixed Reality Visual Displays" IEICE Transactions on Information and Systems, vol. E77-D no. 12, pp. 1321-1329, 1994.
- [3] Joe Lamantia. "Inside Out: Interaction Design for Augmented Reality" UXmatters, August 17, 2009.
- [4] Christian Doppler. "History of Mobile AR", Laboratory for Handheld Augmented Reality. (accessed 12/07/2011)
- [5] Rick Oller. "Augmented Reality", Scribd.com, April 2010.
<http://www.scribd.com/doc/44664310/Augmented-Reality>
(accessed 14/07/2011)
- [6] Tim Perdue. "Applications of Augmented Reality", About.com Guide,<http://newtech.about.com/od/softwaredevelopment/a/Applications-Of-Augmented-Reality.htm> (accessed 14/07/2011)
- [7] Gary Bradski, Adrian Kaehler. "Learning OpenCV Computer Vision with the OpenCV Library", O'Reilly Media, 2008.
- [8] Jianbo Shi and Carlo Tomasi, "Good features to track", Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn., pp.593-600, 1994.
- [9] Andrés Bruhn, Joachim Weickert, Christoph Schnörr "Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Optic Flow Methods" International Journal of Computer Vision, vol.61, no. 3, pp. 211–231, 2005
- [10] Oliver Bimber, Ramesh Raskar. „Spatial Augmented Reality Merging Real and Virtual Worlds”, A K Peters, Ltd., pp. 71-90, 2005.
- [11] Hong Hua. "Stereoscopic Displays" Optical Sciences Center, University of Arizona, Tucson, AZ 85721, publications materials, June 2004.

- [12]Gianpierre Villagomez. “Augmented Reality” University of Kansas, lectures materials, vol.3, December 2010.
- [13]Jean-Yves Bouguet, Pietro Perona.”Visual Navigation” California Institute of Technology, November 1998.
- [14]H. Freeman, "On the Encoding of Arbitrary Geometric Configurations," IRE Trans. Electronic Computers, vol. EC-10, pp. 260-268, June 1961.
- [15]ARToolkit official project development page
<http://www.hitl.washington.edu/artoolkit/index.html> (accessed 24/07/2011)
- [16]Tomohiko Koyama.”Introduction to FLARToolKit”.
<http://saqoo.sh/a/labs/FLARToolKit/Introduction-to-FLARToolKit.pdf> (accessed 24/07/2011)
- [17]Josh Smith. “WPF Apps With The Model-View-ViewModel Design Pattern”. MSDN Magazine, issue 2009 Feb.

Contents of the CD

The content of attached CD directory structure is described by the following diagram:

