CipherCode: A Visual Tagging SDK

Ashish Mehta
Dept. of Computer Science
University of Cape Town
South Africa

Ramone Karodia
Dept. of Computer Science
University of Cape Town
South Africa

Steven Lee
Dept. of Computer Science
University of Cape Town
South Africa

Supervisor: Audrey Mbogho

{amehta|rkarodia|slee|ambogho}@cs.uct.ac.za

ABSTRACT
This paper describes the design and implementation of a visual tagging SDK which can be used to create customized tags and tag decoders. Its aim is to extend the usability of visual tags by increasing the efficiency at which the tags are processed and by incorporating encryption. The SDK consists of three core components namely, the Tag Generator module, Image Enhancement module and the Tag decoder module. The efficiency and accuracy of the SDK was evaluated under varying light intensities and the results are presented in this paper

1. INTRODUCTION
The ubiquity of powerful programmable mobile phones with integrated digital cameras has lead to the development of mobile visual tagging decoders.

A visual tag is simply an image which contains encoded data. A typical example of encoded data is a website address.

A major application of this can be seen in the tourism industry. A visual tag (in such an application) would be attached to a building's façade so that tourists can take a snapshot of the tag with their camera phone. Decoding the snapshot provides a URL to an online encyclopedia containing important facts about the building, company or organization it represents (e.g. Semapedia [15])

This technology enhances the usability of mobile phones by allowing vision based input.

The first stage of this project was to evaluate existing visual tag decoder designs in order to identify algorithms and techniques which offer the best tradeoffs between speed and accuracy. In addition, an encryption module was developed so as to allow tags to be encrypted using a secret key. This is a feature that has not been provided by other tags evaluated.

The results of the research led to the development of a visual tag decoding SDK for both desktop applications and mobile phones. The SDK provides a complete framework for creating tags, as well as an accurate, robust and high performance decoder.

For the rest of this paper our motivation for working on this project is explained, after which a short discussion on related work is presented. This is followed by a discussion on the development of the SDK and finally the paper ends with the conclusion and possible future work.

2. MOTIVATION
Mobile phones have truly revolutionized the way we communicate and have proved to be an invaluable tool in today's hi-tech world. The mobile phone possesses qualities which make it a popular tool in many facets of life including business, fashion and entertainment. However, due to its small compact design, the degree of usability remains low and input via small keypads remains a major problem. Innovations such as predictive text and voice commands have improved this but the HCI (Human Computer Interaction) situation is still far from ideal.

Thus visual tags were developed, to alleviate some of the usability issues affecting mobile phones.

Organisations such as SemaCode[10] and ShortCode[11] currently use Visual Tagging technology to improve the manner in which web address are inputted.

Our primary concern was to extend the usability of visual tags by:

- Increasing accuracy and versatility of tag decoding in varying light conditions.
- Use encryption to provide flexibility in the type of information that a tag can store.

3. RELATED WORK
Provided below is a short description of works related in the field:

CyberCode [6] is a Visual Tagging System which is designed to be used in several augmented reality systems. CyberCode uses the tag system to create a link between physical and digital spaces. These links may be attached to specific data or activate some associated action on a digital device.

[10] talks about a system for ubiquitous computing, which is known as SemaCode. Using the SemaCode SDK you can create visual tags for objects and contexts, and read them using a mobile camera phone. The SemaCode software running on the

phone will then deliver the appropriate mobile content. This system works by embedding a URL (web address) into a two-dimensional barcode (the tag). The SDK software contains the capability to detect and decode tags that are obtained from cellular phone cameras. It extracts the URL and sends you to that address using the phone's built-in browser.

[12] Describes a 2D Bar code system known as QR code. This system was released in 1994 and was built to try and meet the growing demands for codes that were capable of storing more information. This system was an upgrade from the bar code system that was being used at the time, which only stored information in one direction. The QR code, on the other hand, stores information both in the vertical and horizontal directions and thus has greater volume in which to store data.

In [4], a visual tagging system called TRIP (Target Recognition using Image Processing) is presented as a solution to the costly installation of Sentient Computing in living spaces. Sentient Computing provides computers with the ability to perceive the location or action of a user so that the computer system may assist or react to the user's activities. Typical installations require specialized hardware like infrared sensors to be installed throughout the user's environment. The cost of such installations would be in the order of $1000-$2000 per room. TRIP provides a visual tagging system which provides the location detection properties of Sentient Computing using off-the-shelf webcams and computers. The major part of TRIP is the TRIPtag which is a circular 2D black and white identifier. The computer system would be constantly taking snapshots of the living space through the webcams and would scan the images, in real-time, for the TRIPtag. The orientation and depth of the tag within the image would determine the user's location in the living space.

In [1], an infrared LED tag called "Balloon tag" is introduced. This tag works by emitting invisible signal patterns which an ordinary video camera can recognize. The tag would emit a signal which identifies the user. The authors combine this idea with Bluetooth wireless communication. The Balloon tag receiver (computer and video camera) selects the nearest Bluetooth transmitter and create a connection by using a part of the Bluetooth device address. The Balloon tag receiver can then choose whether or not it wants to communicate with user based upon the ID.

4. IMPLEMENTATION

In order to make our SDK as universal as possible the system was broken down into a number of modules, each of which can be customized or replaced. The core modules are:

- Tag generator module: This essentially creates tags which encodes information which a user provides.
- Image enhancement module: This enhances images of tags that are taken with a camera, so as to improve the performance of the tag decoder in terms of accuracy without negatively affecting speed.
- Tag decoder module: This module decodes the information stored within a tag.

Together these modules provide the main functionality needed to successfully create and decode tags. Additional optional modules are also included which enhance the capabilities of the system. These include the Encryption/Decryption Module, the Error Correction/Detection Module, the Camera Interface Module and the ASCII/Binary Converter. Figure 1 shows how these modules relate.

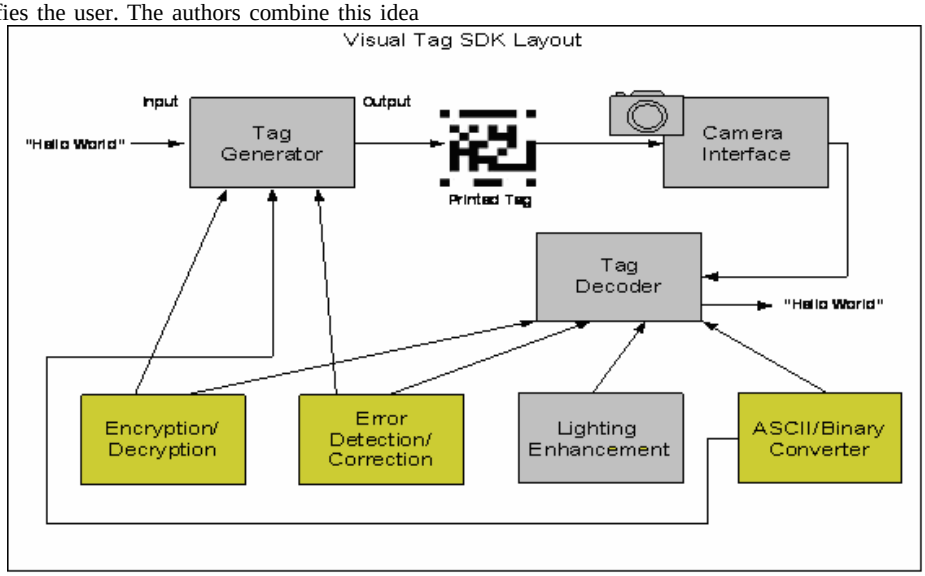


Figure 1: The Core (gray) and Optional Modules (gold) of the Visual Tagging SDK

4.1 Tag Generator

The aim of this module was to design a visual tag that provided efficiency in terms of space, and allowed for quick detection (within an image), and accurate decoding.

Following is a description on the design of the tag.

4.1.1 Design

The **Tag Creator module** is responsible for creating the CipherCode tag. The only input to this module is an array of bits that need to be embedded into the tag. Each bit is represented by a square; black squares representing the value '1' and white squares representing the value '0'.

Figure 2: CipherCode Tag Design

The tag consists of two guide bars and three cornerstones. The top guide bar is always four blocks shorter than the length of the entire tag. If the tag dimensions were 10 x 10, the top guide bar would be made up of 6 blocks. The bottom guide bar is always half the length of the top guide bar. It is oriented in such way that the centre of the bottom guide bar is inline with the centre of the top guide bar. The three cornerstones are situated in the top left, top right, and bottom left corners of the tag. The black square at the bottom right corner of the tag in Figure 10 signifies the data bits are encrypted. If there is no black square in that corner then this signifies the data bits do not require decryption. The geometry of the tag requires the tag decoder to find two parallel bars, one bar half the length of the other, with cornerstones situated collinearly with respect to the guide bars at specific relative positions. For decoding simplicity, the guide bars and cornerstones are separated by white space with respect to each other and the data area of the tag. The area in-between the row of white space adjacent to the top and bottom guide bars and cornerstones is reserved for data. For a tag of dimensions m x n (row by column), the data area of the tag will always be (m-4) x n. The '4' in the formula represents the four rows reserved for the guide bars, cornerstones, and white space.

In order to make the tag more versatile and robust the following optional modules (indicated in gold in figure 1) are used by the Tag Generator:

The **Character Encoding module** is responsible for converting characters into ASCII binary bits and vice-versa. Each character within the input data is represented as a 7-bit ASCII character.

The Error **Detection and Correction module** is responsible for implementing the Hamming Code algorithm [16, 17]. This algorithm provides 1-bit error correction and 2-bit error detection within each block of data. Typically a block of data will be seven or eight bits.

The **Encryption module** is responsible for encrypting data before it is embedded within a tag, and decrypting data read from a CipherCode tag. This module implements the Advanced Encryption Standard [18] (AES) algorithm for encrypting and decrypting data. A 128-bit secret key is generated from a user-defined passphrase every time data is encrypted and decrypted.

The **Parameterisation module** can come in use when multiple data items are required to be embedded within a visual tag. An example of this would be in creating an ID card. The tag could encode the owner's name, home address, and telephone number. Each data item would be separated by a splitter and would be decoded separately by the tag decoder. It is up to the tag developer to decide what splitter to use.

4.2 Image Enhancement

Once the tag has been created it may be used in a variety of applications, all of which require a picture of the tag to be taken, at some point, for further processing.

The camera interface module (indicated in figure 1) is used to capture an image of a tag, and it is at this point that image enhancement module comes into play.

Image enhancement is used as a pre-process to tag decoding.

The aim of this module is to enhance the image of a visual tag so as to improve the accuracy at which tags are decoded in poor light conditions. This will make a visual tag system more robust.

4.2.1 Design

Three image enhancement techniques were considered: Histogram Equalization (HE), Adaptive Histogram Equalization (AHE) and Contrast Limited Adaptive Histogram Equalization (CLAHE).

These techniques were tested under varying light conditions, and the best technique (in terms of processing time and accuracy) was then selected to be part of the visual tagging SDK. The design and results of these tests are shown in sections 5.2 and 6 (respectively) of this paper.

The image enhancement module consists of two core components (which exclude the techniques mentioned above), these are:

- Gaussian smoothing component: This is essentially used to remove any noise that has been introduced during the acquisition of a tag.

- Laplacian edge enhancement component: Used to enhance the edges of a tag.

Both the Laplacian and the Gaussian filters were used as pre and post processing techniques during image enhancement.

4.2.2 Properties of Image Enhancement techniques

HE, AHE and CLAHE all make use of an image's histogram to perform enhancement.

The Histogram of an image represents the relative frequency of occurrence of gray levels within an image. Histogram modelling techniques are used to modify the grayscale (and colour) range and contrast values of an image such that its intensity histogram fits a desired shape.

Histogram Equalization is used to modify an input image's intensity histogram in order to obtain an output image with a uniformly distributed histogram. The resultant effect will be that the output image will have a perception that overall contrast is optimal (thus giving an enhanced image).

The process of histogram equalization involves the use of a transfer function which reassigns the brightness values of output pixels based on the input image histogram. The process does not affect individual pixels brightness order (that is they remain brighter or darker than other pixels) but only modify/shift the brightness values so that an equal number of pixels have each possible brightness

In more complicated cases, the image histogram may not be a good representation for local statistics in two separate parts of the image. In such a case Histogram Equalization may not enhance the image well enough to represent the two areas. In such a case **Adaptive histogram equalization**, is more appropriate. In this algorithm the image is segmented into several rectangular domains (windows), the histogram equalization is then applied to each of these domains. Once this is completed the brightness levels are modified to match across boundaries.

CLAHE is an extension to AHE which limits the maximum contrast adjustment that can be made to any local histogram. This limitation is useful so that the resulting image does not become too noisy (which is a problem with adaptive histogram equalization).The limitation is performed by allowing only a set maximum number of pixels within each gray level associated with a local histogram. After clipping the histogram, the pixels that were clipped are equally redistributed over the whole histogram to keep the whole histogram count unchanged.

Table 1 below summarises the properties of the image enhancement techniques used.

Table 1: Properties of image Enhancement Techniques

Technique	Property
Histogram Equalization	Adjusts the global contrast of an image, by sharing out the intensity levels of each pixel across the image. Thus entire image is enhanced. Downside of techniques is that it cannot provide local enhancement of specific regions within an image. It also tends to introduce undesirable artefacts and noise
Adaptive Histogram Equalization	Used for local enhancement of a region within an image. Is an extension of Histogram Equalization. The downside of technique is that enhancement is so strong that it has the tendency to amplify noise in flat (uniform contrast levels) regions of an image and create ring artefacts at strong edges.
Contrast Limited Adaptive Histogram Equalization	Provides for local enhancement of regions within an image. Is an improvement over Adaptive Histogram Equalization. It reduces undesired noise amplification and reduces boundary artefacts.

Figure 3 (appendix A) shows the effect of applying each of these techniques (in combination with Gaussian smoothing and edge enhancement) on a sample tag image taken in poor light. The images are shown after binarization (conversion to black and white) has been performed since that is the image used by the tag decoder. As can be seen HE does well to enlighten the image but in the process it has also diminished the tag. AHE does well in bringing out the contents of a picture, but the process introduces too much noise for it to be useful. CLAHE, on the other hand, seems to enlighten the image without diminishing the tag. Visually CLAHE seems to produce the best images.

4.3 Tag Decoder

After the tag image has been processed by the image enhancement module, the decoder module is used to extract the information stored within the tag.

The aim of this module is to efficiently (in terms of time) detect a visual tag within an image and then accurately decode the information stored in it.

Following is a description on its design.

4.3.1 Design

Figure 4 (Appendix A) shows the outline of the decoding process.

The decoding process is divided into 11 stages, each of which performs a different image processing task. Four of the stages (shown as blue boxes in the Figure 4) are actually implemented in external modules and are simply used by the decoder module. Background research revealed that many tag decoders have common designs and share numerous components. Based on these commonalities, 7 key stages (shown as gray boxes in the Figure 4) were identified and integrated into the decoder design.

- Grayscale Conversion: The system begins the decoding process by first converting the colour image (obtained from the Camera Interface Module) to a grayscale image. This conversion uses the ITU standard formula: *G = (222 * Red + 707 * Green + 71 * Blue)/1000*

- Image Enhancement : The decoder then uses the external Image Enhancement (discussed in previous section)

Module to improve the quality, contrast and clarity of the grayscale image as well as to enhance the edges of the tag.

- Binarization: The next step is binarization which thresholds the grayscale pixel values to either 0 or 1 for black and white respectively. Two thresholding algorithms, namely global thresholding and quick adaptive thresholding (presented in [14]), were implemented and tested. Experimentation revealed that quick adaptive thresholding yielded superior results for the majority of test cases. This algorithm calculates a moving average and sets a pixel to black only if it is significantly darker than this average. Otherwise the pixel is set to white.

- Region Detection: Binarization is followed by a two pass region detection algorithm which identifies large regions of connected black pixels. The first pass labels all black pixels according to the labels of its neighbours:

    - If all the neighbours have 0 labels (i.e. are all white) then the pixel is labelled with a new unique non-zero label.

    - If there is exactly one neighbouring pixel with a non-zero label then the pixel is assigned the same non-zero label

    - If there is more than one neighbouring pixel with a non-zero label then the pixel is assigned the smallest label and the conflict is recorded in a special equivalence data structure.

  Label conflicts are resolved during the second pass which re-labels pixels according to the equivalence data structure. This data structure is a table which stores pairs of adjacent (or conflicting) labels

- Guide Bar Identification: For each region identified in the previous step, the second-order moments [7, 13] are calculated. From these moments the eccentricity (measure of how long a region is) and orientation are calculated. Pairs of parallel and elongated regions (eccentricity greater than 6) where one bar is twice the length of the other are identified as candidate guide bar pairs

- Cornerstone Detection: The orientation and size of these guide bars pairs are then used to estimate the position of the three cornerstones. Since second-order moments provide the major and minor axis, the lengths of the bars as well as the lengths of a single cell are known. These lengths are then used to estimate the position of the cornerstones relative to the centres of the two guide bars.

- Projective Matrix Transformation: The positions of the second shorter guide bar together with the positions of the 3 cornerstones are then used in a texture mapping technique described in [2] to calculate the transformation matrix. Once this matrix is known, tag coordinates can be converted into image coordinates. The image coordinates estimates the centres of the corresponding block.

- Decoding: The final step in the decoding process is simply checking whether or not the pixel values at the calculated positions are black or white. An array is then built up and passed onto other modules for further processing.

- Other Modules: The decoder uses additional modules to convert the binary information into human readable text.

    - Error Detection/Correction: used to compensate for any decoding errors.

    - Encryption/Decryption module: If the encryption check cornerstone is filled then this module is used to acquire a key from the user and decrypt the text.

    - ASCII/Binary Converter: converts the decoded and possibly decrypted bits into ASCII characters

5. TESTING

This section describes the tests that were carried out to evaluate the performance of each of the core modules.

5.1 Tag Generator Test

Tests showed that the CipherCode tag is more space efficient overall than the Semacode[10] tag. This result is quite significant considering that semacode is one of the better known and more popular Visual Tag systems.

5.2 Image Enhancement Test

One of our main objectives is to develop a Visual Tagging SDK that is capable of decoding tags accurately under poor light conditions. Thus to provide this feature the image enhancement module was developed.

In order to effectively evaluate the performance of the visual Tagging SDK and the impact that image enhancement has in improving the processing (detecting and decoding) of visual tags, tests were carried out on separate sets of pictures taken at varying light intensities.

The time taken for each image enhancement algorithm to process an input image was recorded as the objective is to find an algorithm that enhances a tag image enough so as to provide accurate but fast decoding of a tag. This is necessary for applications that will be run on cell phones which, generally, have slow processors.

The enhanced images were then passed through the tag decoder module, where the output was compared to the input data (i.e. the data used to generate the tag). The number of matches was then recorded and categorized based on the measure of the light intensity that the pictures were taken. The results were evaluated based on accuracy and efficiency. The accuracy of an algorithm is defined as:

Accuracy =
number of matches / total no of comparisons 1

The efficiency of the results was based on a comparison of the accuracy and the time taken to perform the image enhancement. Hence the efficiency was recorded as a rank of the different algorithms used. Thus a ranking position of '1' was interpreted as the most efficient.

The experiment was carried out in a dark room. The dark room consisted of 40 Watt ceiling spot lights whose direction and intensity can be manually controlled. The room was closed off, and the only light source available is the one mentioned.

Pictures were taken at the following intensities:

- Full intensity
- 1/2 of full intensity
- 1/4 of full intensity
- 1/8 of full intensity

The evaluation was carried out on a Pentium 4, 3.0 GHz computer. The pictures were taken with a Nokia 6280 cell phone which has a 2 Mega Pixel camera embedded. The night mode and flash features of the phone were not used during the experiments.

For each of the light conditions considered 10 pictures were taken. Thus there were four sets of 10 pictures. Each set was used in the evaluation of image enhancement techniques. The size of the pictures used was 640 x 480.

5.3 Tag Decoding Test

The following tests were carried out to check the performance of the decoder:

- Rotation Tests: these were conducted by using a sample set of specially chosen images. The sample set consisted of 8 instances of the same image, each one differing by a rotation of 45°. This was needed to ensure that the all trigonometric based calculations were correct. Figure 5 (Appendix A) Shows the sample set tested. All the tags in the sample were decoded successfully

- Tilting Tests: Tilting tests were conducted in order to determine the tilting angles at which decoding would fail. During the implementation of the decoder, black box testing revealed that the dimensions of the tag could not be accurately calculated when tilted by large angles due to the perspective distortion. However, if the tags dimensions are fixed then the decoder works at larger tilt angles. The decoder was therefore modified to be able to decode both fixed and dynamic tag sizes. The Table below shows the result of the tests.

Table 2:The maximum tilting angle (that allows for successful decoding) for the two different tag types

Tag Dimensions	Maximum Tilting Angle
Fixed	45°
Dynamic	20°

6. RESULTS

Table 3 in appendix B provides a summary of the results obtained from performing the image enhancement test. The processing time indicated in table 3 is that of the image enhancement technique used and not the total time for decoding the tag.

In all cases tested CLAHE proves to be the most efficient image enhancement algorithm, due to its lower average processing time and higher accuracy. In the ¼ and ½ light intensity scenarios the algorithm was limited to an accuracy of 70%, at the time of testing due to the decoder module not being able to decode tag images that were taken at certain angles away from the normal of the image. (This problem was fixed at the time of writing this paper but is yet to be tested).

In terms of processing time CLAHE is the fastest, this may be due to the fact that the algorithm accesses the image pixels less frequently than the other the other algorithms during the construction of the image histogram.

The processing time for all the algorithms shortens as the light intensity increases. This may be due to the increasing light providing certain regions within an image an even spread of contrast across the grayscale range (i.e. a certain region may have an even set of pixels that have grayscale values that range from 0-255). This would reduce the processing time of the algorithms implemented since their intended effect is to spread the entire range of grayscale values across the pixels of an image.

7. CONCLUSION

This project has led to the development of a contemporary open source visual tagging framework which can be easily integrated into other PC and mobile phone applications. The modular design of the SDK has resulted in a flexible system, where any module can be enhanced and improved independently.

Based on the experiments carried out, the use of CLAHE as an image enhancement technique successfully met our objective of improving the accuracy of the decoding process under poor light conditions.

A head-to-head performance comparison with Semacode revealed that our mobile phone version of the SDK is more efficient than the popular propriety decoder. Table 4 (in appendix B) summarises the results obtained.

On average the decoding time for CipherCode was found to be 4.5 seconds on a mobile phone and 0.390 seconds on a PC, whereas that of Semacode was 6.5 seconds on a mobile phone.

8. FUTURE WORK

As indicated, due to time constraints, not all optimizations could be implemented. Thus as an extension to the work already completed the following could be done:

- Conducting further tests with other image enhancement techniques.
- Testing the efficiency of the Visual Tag SDK under different environmental conditions such as fog.
- Adapt the tag decoder to handle tilted images.
- Improve performance of decoder by considering sub-images where potential tags are thought to be located.

9. REFERENCES

[1] Aoki H. "*Balloon Tag: (In)visible Marker which tells u who's who*", Proceedings of the Fourth International Symposium on Wearable Computers (ISWC'00), 2000.

[2] Heckbert P. S., "*Fundamentals of Texture Mapping and Image Warping*" Master's Thesis, Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, 1989.

[3] Iannizzotto G. et al. Badge3D for Visually Impaired. Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'05, 2005

[4] Ipina D, Mendonca P, and Hopper A "*TRIP: A Low Cost Vision Based Location System for Ubiquitous Computing*" Personal and Ubiquitous Computing (2002) 6:206–219, 2002. Available online: http://paginaspersonales.deusto.es/dipina/cambridge/PUC/PUCpaper.pdf

[5] Mbogho A. et al., "*Towards reliable computer vision based tangible user interfaces"* proceedings of the IASTED international conference, Human Computer Interaction, 2005.

[6]. Rekimoto J. & Ayatsuka Y. "*CyberCode: Designing Augmented Reality Environments with Visual Tags*". Sony computer Science Laboratories, Inc. Available Online: www.csl.sony.co.jp/person/rekimoto.html

[7] Rohs M., Gfeller B., "*Using Camera-Equiped Mobile Phones for Interacting with Real World Objects*". Institute for Pervasive Computing, Department of Computer Science. Available: http://www.vs.inf.ethz.ch/res/papers/rohsgfeller-visualcodes-2004.pdf

[8] Rohs M., "*Visual code widgets for marker based interaction"* proceedings of the 25th IEEE international conference onn distributed computer systems workshops (ICDCSW'05) 2005

[9] Scott D., et al., "*Using Visual Tags to bypass Bluetooth device discovery*" Mobile computing and communications review, volume 9, number 1. 2005.

[10] Semacode Technical Paper. Semacode Corporation. Available: http://semacode.org/about/technical

[11] Shotcode FAQ. Available: http://www.shotcode.com

[12] QR code. Available: http://www.denso-wave.com/qrcode/index-e.html

[13] Veltkamp R. C., Hagedoorn M., "*State of the Art in Shape Matching"*

[14] Wellner P. Interacting with paper on the DigitalDesk. Communications of the ACM 1993; 36(7): 87 96

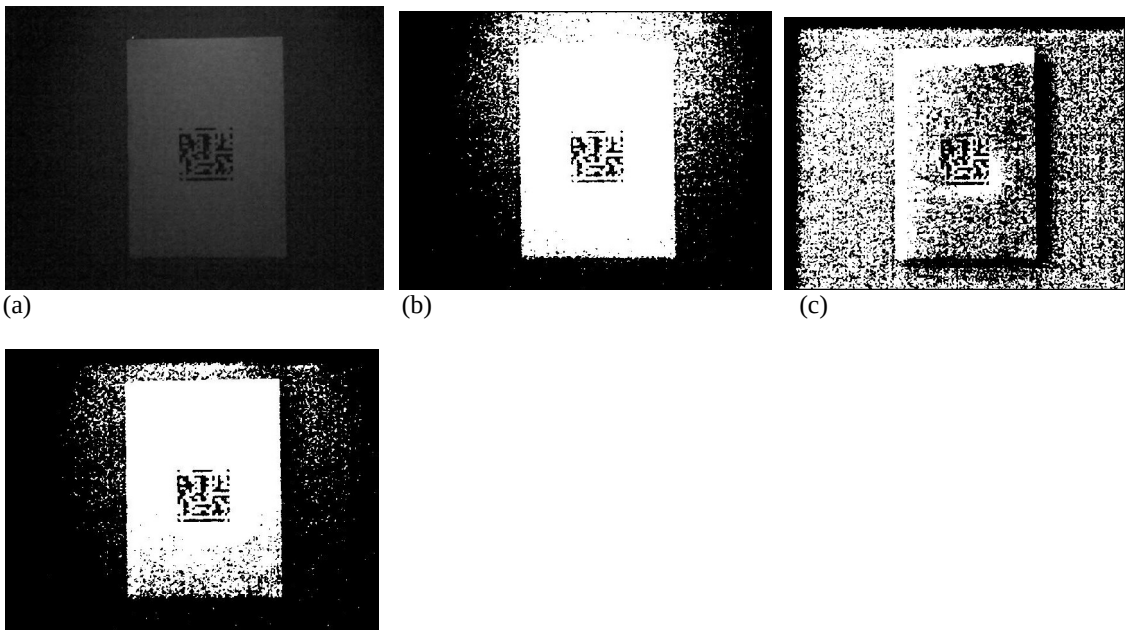
[15] Semapedia Website, Available http://www.semapedia.org

[16] Thomas G. Dietterich and Ghulum Bakiri. Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, pp. 263-286, January 1995.

[17] Wicker, Error Control Systems for Digital Communication and Storage. *Prentice-Hall*, 1995.

[18] Stallings, W. Network Security Essentials – Applications and Standards. *Prentice Hall*, Second Edition, 2003.

APPENDIX A: LIST OF FIGURES



(a) (b) (c)

(c)

Figure 3: Results from using image enhancement: (a) Original image;(b) Result of performing edge enhancement then HE then smoothing then applying binarization to (a); (c) Result performing AHE then smoothing then to (a) then applying binarization;(d) Result of performing edge enhancement then CLAHE(with 16x16 windows and a histogram clip limit of 3) then smoothing then applying binarization to (a).

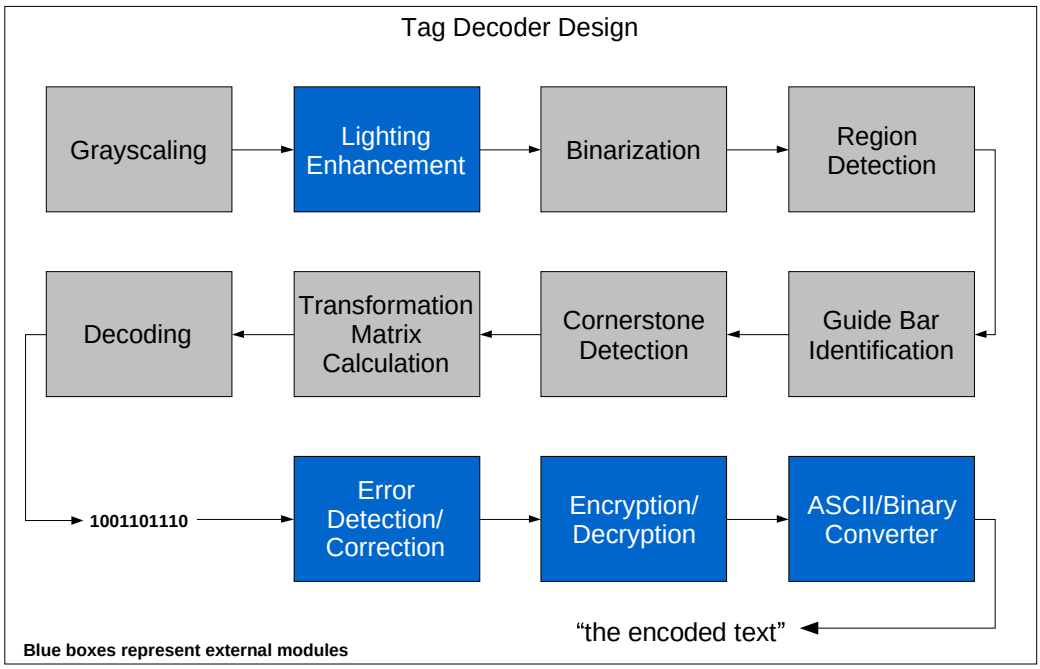


Figure 4:The Internal Design of the Decoder Module

Figure 5: Sample set used for rotation tests

APPENDIX B: LIST OF TABLES

Table 3: Summary of Results obtained from testing procedure

Enhancement Technique	1/8th light intensity			1/4 Light Intensity			1/2 light intensity			Overall efficiency (rank)
	Avg. processing time (secs)	Accuracy (%)	Efficiency (rank)	Avg. processing time (secs)	Accuracy (%)	Efficiency (rank)	Avg. processing time (secs)	Accuracy (%)	Efficiency (rank)	
HE	1.82	0	2	1.70	40	2	1.44	60	2	2
AHE	1.96	0	3	1.91	0	3	1.44	30	3	3
CLAHE	1.70	0	1	1.53	70	1	1.46	70	1	1

Table 4:Performance Comparison between CipherCode and Semacode

Type of Image	CipherCode	Semacode
Close-up of tag	4.5 seconds	6.5 seconds
No tag with few regions	5.1 seconds	10.3 seconds
No tag with many regions (complex scene)	14.3 seconds	22.1 seconds