Extending Photographic Steganography to Android: Real-time Camera-Display Messaging using a Smartphone Camera

Independent Study Report

Joseph A. Boyle

Abstract

We implement a real-time visible-light communications technology which can transmit messages embedded in images, without requiring a highly powered display or camera. The system can send messages of arbitrary length and uses differential metamers to make message transmission nearly invisible to the human eye. We were able to consistently achieve at least 95% retrieval accuracy.

1. Introduction

Given the universality of electronic displays and smartphones, there has been a natural evolution of communication technologies relying on household display-communication. QR codes, VRCodes, and other technologies [1,4,5,6,7] have made giant leaps in the space of data wireless data communication between two devices.

Some of the existing technologies fall short when trying to send long messages in a constrained area -- barcodes, for example, are limited in message length given a fixed size for the image to appear in. VRCodes [5] and Kaleido [4] both are capable of sending messages of variable length given a constrained area, but require high speed displays which are not commonly used in most environments. While requiring a specialized display is suitable for lab environments, they make the technology inaccessible for everyday use. PixNet [6] and Cobra [7] remedy the need for a high-speed display by utilizing intensity modulation and 2D color codes, respectively, but are only able to produce a machine-readable channel.

Differently, we seek to create a communications technology which is non-obvious to observers, capable of transmitting messages of arbitrary length in a constrained area, and requires no special hardware. By offering two separate channels - one that is human readable and one that is machine readable - it is possible to send messages in a non obtrusive way. A technology capable of sending long messages on consumer-grade hardware between has far-reaching applications in the security space, such as passkey-free authentication.

1.1 Visual MIMO and Challenges

Visual MIMO [2] is a visible-light communications technology which utilizes intensity modulation to embed messages into images for plain-sight message transmission. Similar to HiLight [8], Visual MIMO produces a machine readable channel *and* and human readable channel. Visual MIMO functions by displaying two frames in rapid succession. The first frame contains the base image with the message subtracted, while the second frame contains the base image with the message embedded on top (further explained in section 2.1). These frames should display at exactly half of the capture rate of the target camera [9].

In the interest of making the technology more accessible, Visual MIMO was ported to the Android operating system to be utilized on Android phones. Utilizing Android cameras presents a distinct set of challenges. Unlike high quality lab cameras, Android cameras are much less sensitive to changes in color and image features, and typically run at a much lower capture rate. These problems further compound pre-existing issues with Visual MIMO. It is largely documented that human vision is sensitive to even subtle changes in color [3]. Thus, the machine-readable channel, created via intensity modulation, is largely susceptible to being noticed by humans. Whereas VRCodes [5] and Kaleido [4] can curb some of these effects by displaying at a higher framerate than human eyes can perceive, Visual MIMO must display at a low frame rate (the Nyquist frequency) [9] to remain viable for retrieval by Android phones.

The existing infrastructure of Visual MIMO is capable of sending messages of eighty bits. Given that an ASCII character utilizes seven bits, the scope of messages that can be sent is limited to those that are at most eleven characters. Previous work to expand the length of messages that could be sent via Visual MIMO has generally involved using three of the eighty bits to identify which “index” the current frame belongs to [9]. This approach fails to consider the effects of a camera’s rolling shutter, which may cause portions of the image to be unusable, rendering the three extra bits unusable. Additionally, retrieval failure yields frames being mismatched and thus incorrect message extraction.

1.2 Semester Goals

Visual MIMO was limited in that message transmission was very obvious and only effective for messages of a maximum of eleven characters. To transmit a message of arbitrary length, such as a data stream, the system must be able to utilize multiple packets.

The effect of message transmission needs to be absolutely negligible to the human eye. Effectively, the system must be able to scale for a message of any arbitrary size, while still maintaining an accuracy $\geq 90\%$. Additionally, an application needs to be developed to allow for easier benchmarking of how changes to the system impact performance.

2. Message Encoding

2.1 Previous Techniques

Previously, Visual MIMO used intensity modulation to embed messages onto an image. By setting the intensity of an “on” region to α and the intensity of an “off” region to 0, a binary sequence representing a message could be embedded into an image.

At low α values ($\alpha \leq 3$), the difference between regions of intensity 0 and α is difficult for the human eye to differentiate. To consistently achieve recovery accuracies above 90%, however, α must be ≥ 10 (Figure 5). Thus, we needed to either minimize the visibility of messages embedded with a high α value, or maximize the recovery accuracy at low α values.

2.2 Error-Correcting Code

The texture and color distributions of images are generally not equally spread throughout the entire image. This inequality, coupled with an unequal distribution of light throughout the image, leads to certain areas of the image yielding higher bit recovery accuracies than others.

To help circumvent this issue, each character in the message is embedded twice, the second sequence being the inverse of the first:

[ASCII: **b**; BINARY: **1100010**] → 1100010 0011101

Message retrieval depends on there being an equal distributions of “on” and “off” bits in the message to more accurately differentiate the on and off states (as described in section 3.1). Inverting these redundant bits ensures this equal distribution while also increasing the accuracy of message retrieval (having two sources curb the effects of ineffectual [saturated] sectors of an image).

2.3 Color Encoding

Color encoding differs from intensity modulation in that instead of setting the intensity of each channel (RGB) of a pixel to α or 0, each channel’s intensity changes relative to a color pair, referred to as a differential metamer [1], which is close to the original color. The pixels value is then incremented by the color pair multiplied by α or 0.

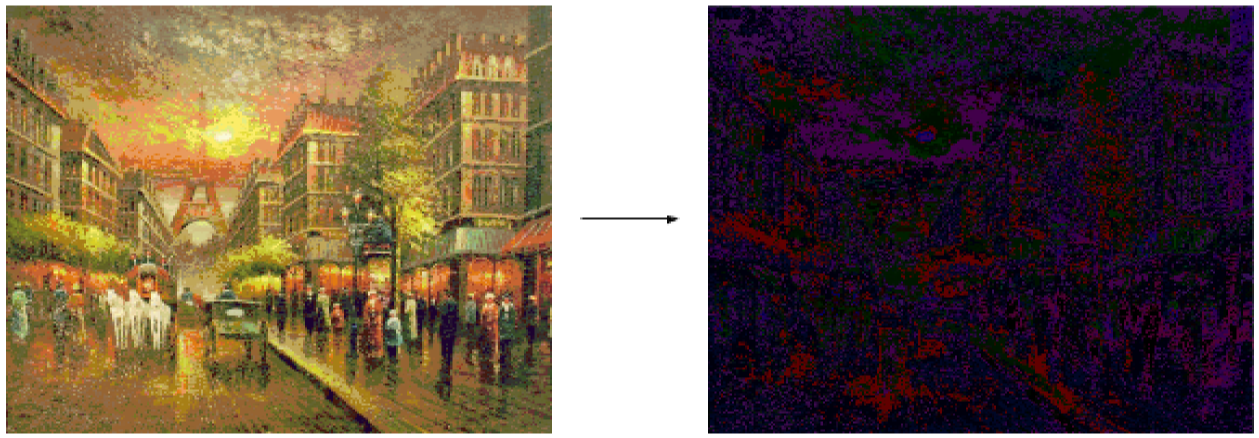

Figure 1. The base image (left) and the mask of matched color pairs at an α of 100 (right).

Using a close color pair instead of black yields equal performance while minimizing the visual impact of message transmission. This takes advantage of the human eye's sensitivity to some shades over others -- the eye is most sensitive to changes in green/yellow [3].

2.4 Color Pair Training

In the interest of finding pairs which perform well with Android cameras, we encode a message of alternating 1s and 0s (10101010…) at an α of 5, and evaluate the message that an Android phone retrieves. Any pairs with an accuracy < 95% are dropped from the data set.

Color pairs which perform well for the camera aren't necessarily invisible to the human eye. For this reason, we must then go through each pair and determine if there is a noticeable difference between the two frames. Color pairs which both perform well and aren't noticeable are deemed "good" and used in our algorithm. All other pairs are deemed "bad" and ignored in our algorithm.

Once a collection of "good" pairs are identified, several thousand more may be generated by evaluating colors within the same ellipsoid cluster[4]. Those colors can then be verified to be accurate performers for the device.

3. Message Retrieval

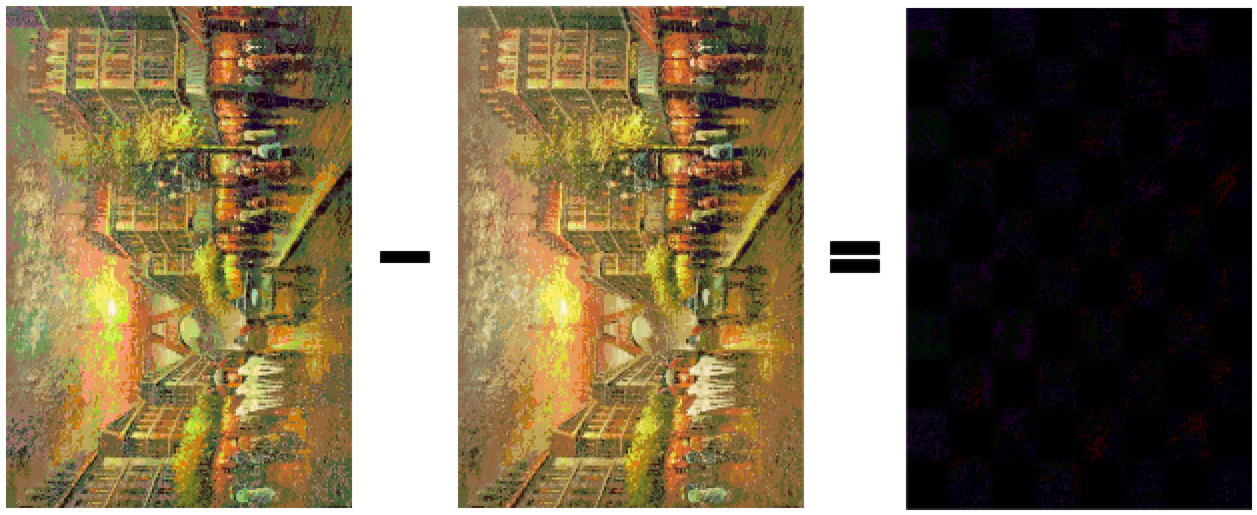

Figure 2 The subtraction of '01010101…' at $\alpha = 40$.

3.1 Color Decoding

To extract the full message from an embedded frame, we must first subtract the frame containing the base image from the frame containing the base image plus the embedded message:

$$\frac{(frame+message)-(frame-message)}{2}=message$$

Dividing this message into a ten by eight grid yields eighty regions which need to be evaluated. For a given pixel in a region, the threshold by which the color changes is not uniform among all channels. Each channel (red, green, and blue) in the resultant message image has a value between 0 and 255. Thus, we define the α value of the region as the average Euclidean distance of all of the pixels (M = total number of pixels) in the region:

$$\alpha = \frac{\sum_{n=0}^{M} \sqrt{(red_n)^2 + (green_n)^2 + (blue_n)^2}}{M}$$

The threshold by which we determine which regions are "on" or "off" is the median of the α values for all regions (equal number of "on" and "off" on each side of the threshold). Any value greater than the median is said to be a one and all other values are zero.

3.2 Message Reconstruction

Since each character is embedded twice, the system can make an educated guess of what each bit in the message should be by comparing the values of the character and inverted character. Since we embed each bit and its inverse (section 2.2), it follows that a bit n and its redundancy bit should have opposite values. If these bits are opposite, the bit that goes into the message is the value of bit n.

If the values of the bits are equal, there must have been a retrieval error of one of the bits. To alleviate this error, the absolute differences between the value of the bits and the median are compared. Whichever bit produces a greater |α - median| should be the value of the bit in the message.

4. Multi-framing

The previous Visual MMO system could transmit messages of 80 bits, which meant a maximum transmission of 11 ASCII characters, limiting our scope to 26^{11} possible messages.

4.1 Differentiating Packets

In the original setup, we could begin capturing frames at any arbitrary point in message transmission and decode whichever two frames produced the greatest absolute difference, as each frame was used to transmit the same message. Each packet in the new setup, however, transmits a different message, and thus we need to differentiate frames from packet A versus packet B.

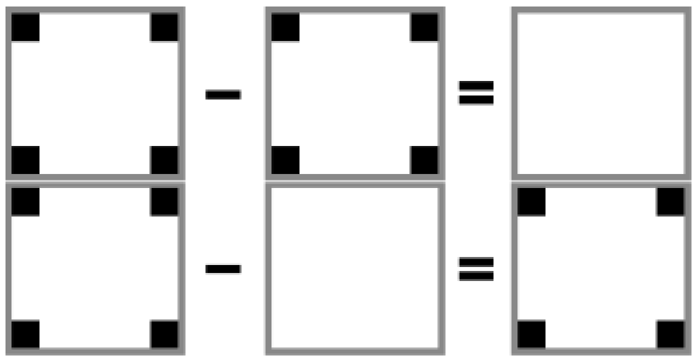
Figure 3. In the top portion, the parity bits of two "even" frames cancel out (well matched). In the bottom, parity bits from an "even" and "odd" frame are subtracted, yielding parity bits remaining in the corners (mismatched).

Our implementation uses a single parity bit in each corner of the frame. These parity bits are set to "on" for every even numbered packet, and "off" otherwise. Thus, we can differentiate whether two subsequent frames are from the same packet by subtracting the parity bit regions and evaluating whether they are "on" or "off".

4.2 Ordering Packets

Four parity bits are embedded in each corner to indicate what the current index of the packet in the message is. Thus, it is possible to determine the order in which packets should go in final message formation, as well as detect which packets are still needed to complete the message.

5. Benchmarking

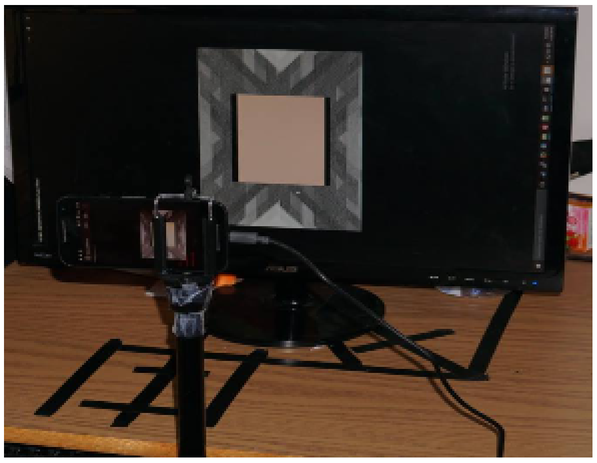
Figure 4. Test bench used to test accuracies of various color pairs. The border is subtracted.

5.1 Test Bench

To reliably measure how effective changes to the decoding algorithms are, there was a clear need for a testing application which eliminates human error from experimentation (camera movement). For this reason, we developed a system which displays the videos with messages embedded, communicates with the phone via sockets, and analyzes the results of message recovery.

Using this system, we are able to capture over ten thousand messages without requiring human interaction. We conducted these tests with a Motorola G Android Phone (Model XT1540, 5 Megapixel camera) and an Asus VS248H monitor. All tests were conducted on a machine with an AMD Athlon X4 processor, Radeon R7 260X graphics card, and 4GB of RAM. The camera was positioned twelve inches away from the screen at a zero degree angle (perpendicular to the screen).

5.2 Initial Results

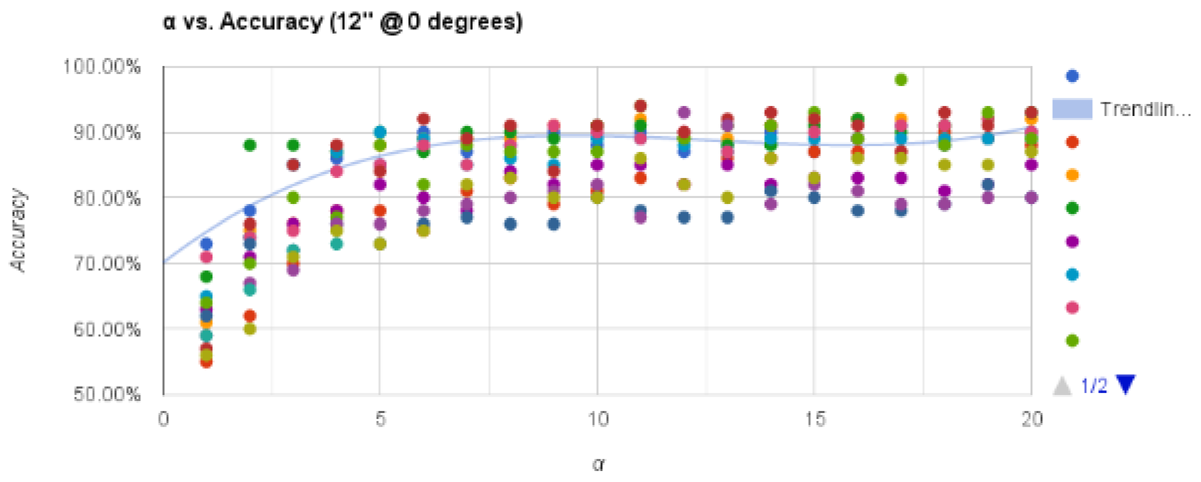

Figure 5. Messages embedded with intensity modulation have recovery accuracy ≥ 85% at α ≥ 8. 14 images tested with 6 messages 3 times each per message per α value. Full data in Table 1.

For a given α, we found an standard deviation of 5.20% which is a clear indicator that some messages have features which cause message retrieval to fail. Images that lack the texture variation that Vuforia expects tended to be very difficult to track and are thus excluded from our data set, as they may have added human error.

Images with regions of high color saturation (very light) tended to perform much worse than those with a more uniform distribution of color. This is almost

5.3 Final Results

Initial testing of a system with multi-framing and color encoding yielded an accuracy of 100% at an α of 30, and nearly 96% accuracy at α values as low as 20. The accuracy dropped to nearly

60% at an α of 15. Message recovery for a message of 4 packets took approximately 40 seconds at low α values.

The aforementioned results were obtained using color pairs that a high quality lab camera were easily able to discern. After finding the color pairs which perform well on the Motorola G phone (section 2.3.1), we obtained 100% message retrieval accuracy at α values as low as 10, and 97% accuracy at an α of 5. Message recovery for a message of 4 packets took approximately 15 seconds.

5.4 Data Analysis

We found that the minimum α value needed to *consistently* obtain ≥ 90% message retrieval accuracy for intensity modulation is 10. Using color embedding and our system of redundancy, however, we found that an α value of just 4 is needed. More importantly, messages embedded with intensity modulation were almost always more noticeable than those that used color embedding. Thus, we can consistently obtain higher recovery accuracies while further minimizing the visual impact of message transmission.

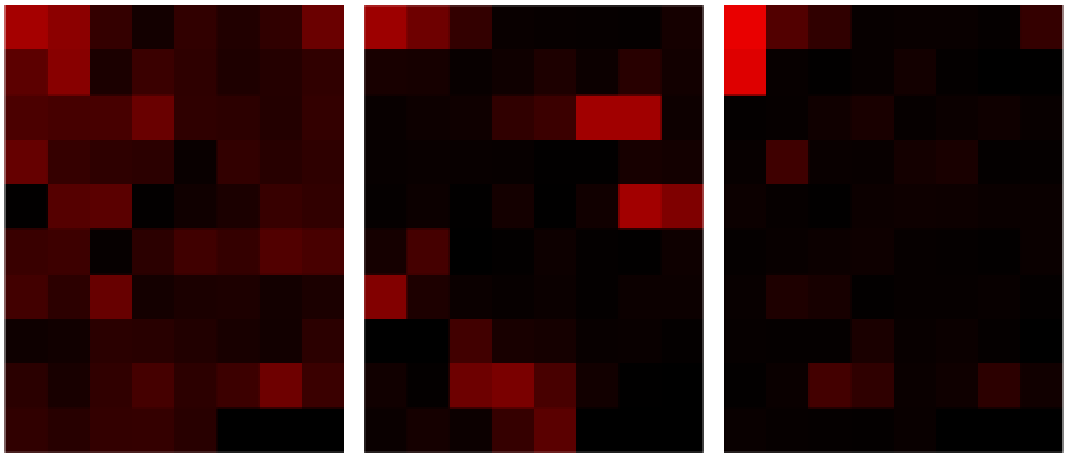
Figure 6. An image representing the frequency at which bits are recovered incorrectly for images 4, 5, and 13, respectively, using intensity modulation. Areas of bright red indicate low recovery accuracy.

Contrary to trend of accuracy increasing as α increases, we find that at a certain point, increasing the α value actually *degrades* the retrieval accuracy, largely due to the resultant color saturation. Looking further into the results, we found that the top left corners and bottom portions of images tended to perform much worse than the interior regions of the images. It appears that regions of the image that abruptly change from one color to another yield lower bitrate accuracies. That is, saturation plays a large role in whether an image is viable to be used in our system. The images that tended to perform best were those that were uniform in texture and color distribution.

Additionally, we found that as the retrieval accuracy of a single packet utilizing color embedding increased, the time required to extract the entire message for a series of packets decreased. That is, when the system was able to extract a message without making as many “guesses” (thus leading to a higher accuracy, as described in section 3.2), the total time for message extraction decreased. Thus, messages transmitted as lower α values required more time for full retrieval, as more “guesses” had to be made for regions of the image that were hard to retrieve.

6. Future Work

Having more color pairs available for the system to select from generally yields a greater invisibility. This makes intuitive sense since all pixels in a region of intensity α must be mapped to a color in our data set, regardless of whether that match is close or not. That is, in a system with only one color pair, every pixel that is mapped will be mapped to that pair. It follows, then, that creating a more seamless integration requires a larger set of data.

6.1 Test Bench

The test bench has room for improvement in the way of stability and ease of use. Expanding its functionality to make training color pairs easier will ultimately result in more color pairs being tested in a shorter time period, enabling us to find color pairs that perform best for the lower quality Android cameras.

6.2 Human Study

To reliably determine which color pairs are visible requires a human study. Using a small selection of people to test color pairs yields bias towards the group’s collective visual traits. A larger study, however, will reveal general trends of which color sensitivities exist and help eliminate pairs that use these sensitivities.

6.3 Speed Improvements

While the system functions as is, message retrieval generally grows nonlinearly as message length increases and α decreases. The most immediate solution for this is to increase the amount of information that can be stored in a single packet, as failure to retrieve a packet correctly causes the system to have to wait for the packet to display again. Additionally, finding color pairs that the system perform extraordinarily well with increases the accuracy of the system, and thus the overall speed of retrieval.

References

1. E Wengrowski, K Dana, M Gruteser, N Mandayam. Reading Between the Pixels: Photographic Steganography for Camera Display Messaging. *arXiv:1604.01720*
2. W. Yuan, K. Dana, A. Ashok, M. Varga, M. Gruteser, and N. Mandayam. Photographic steganography for visual mimo: A computer vision approach. *IEEE Workshop on the Applications of Computer Vision (WACV)*, pages 345-352, 2012.
3. Robinson S, Schmidt J. FLUORESCENT PENETRANT SENSITIVITY AND REMOVABILITY - WHAT THE EYE CAN SEE, A FLUOROMETER CAN MEASURE. *Materials Evaluation* [serial online]. July 1, 1984;42(8):1029-1034.
4. Zhang, L., Bo, C., Hou, J., Li, X.Y., Wang, Y., Liu, K., Liu, Y.: Kaleido: You can watch it but cannot record it. In: Proceedings of the 21st Annual International Conference on Mobile Computing and Networking, ACM (2015)
5. Woo, G., Lippman, A., Raskar, R.: Vrcodes: Unobtrusive and active visual codes for interaction by exploiting rolling shutter. In: Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on, IEEE (2012)
6. Perli, S.D., Ahmed, N., Katabi, D.: Pixnet: interference-free wireless links using lcd-camera pairs. In: Proceedings of the sixteenth annual international conference on Mobile computing and networking, ACM (2010)
7. Hao, T., Zhou, R., Xing, G.: Cobra: color barcode streaming for smartphone systems. In: Proceedings of the 10th international conference on Mobile systems, applications, and services, ACM (2012)
8. Li, T., An, C., Campbell, A., Zhou, X.: Hilight: hiding bits in pixel translucency changes. In: Proceedings of the 1st ACM MobiCom workshop on Visible light communication systems, ACM (2014)
9. Sopher, R: Detecting Planes in Real-Time for Camera Display Communications, http://revan.io/research/

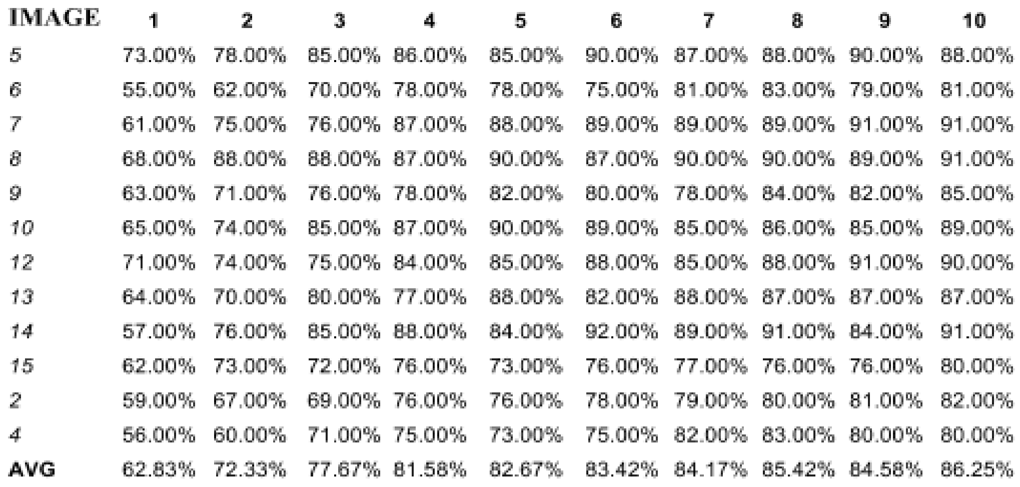

IMAGE	1	2	3	4	5	6	7	8	9	10
5	73.00%	78.00%	85.00%	86.00%	85.00%	90.00%	87.00%	88.00%	90.00%	88.00%
6	55.00%	62.00%	70.00%	78.00%	78.00%	75.00%	81.00%	83.00%	79.00%	81.00%
7	61.00%	75.00%	76.00%	87.00%	88.00%	89.00%	89.00%	89.00%	91.00%	91.00%
8	68.00%	88.00%	88.00%	87.00%	90.00%	87.00%	90.00%	90.00%	89.00%	91.00%
9	63.00%	71.00%	76.00%	78.00%	82.00%	80.00%	78.00%	84.00%	82.00%	85.00%
10	65.00%	74.00%	85.00%	87.00%	90.00%	89.00%	85.00%	86.00%	85.00%	89.00%
12	71.00%	74.00%	75.00%	84.00%	85.00%	88.00%	85.00%	88.00%	91.00%	90.00%
13	64.00%	70.00%	80.00%	77.00%	88.00%	82.00%	88.00%	87.00%	87.00%	87.00%
14	57.00%	76.00%	85.00%	88.00%	84.00%	92.00%	89.00%	91.00%	84.00%	91.00%
15	62.00%	73.00%	72.00%	76.00%	73.00%	76.00%	77.00%	76.00%	76.00%	80.00%
2	59.00%	67.00%	69.00%	76.00%	76.00%	78.00%	79.00%	80.00%	81.00%	82.00%
4	56.00%	60.00%	71.00%	75.00%	73.00%	75.00%	82.00%	83.00%	80.00%	80.00%
AVG	62.83%	72.33%	77.67%	81.58%	82.67%	83.42%	84.17%	85.42%	84.58%	86.25%

Table 1. Results of intensity modulation at various α values.