

Optical Wireless: Chromatic Encoded Visual Data Transfer

Arthur Reloj, Gregory Vert
Department of Computer Science and Engineering
University of Nevada Reno, Reno NV 89557

Abstract - This paper explains a new technique that takes advantage of advancements in digital imaging technology to visually transfer large amounts of data quickly between computing machines. Rather than seeking to improve upon well developed systems already in use to move information between computers, such as data compression, broadband cable, or high-speed wireless, this paper explores the potential usefulness of devices not traditionally used to send and receive data between machines.

Keywords: Chromatic Encoded Visual Data Transfer (CEVDT), chromatic data image (CDI), data compression, data transfer rate, digital imaging

1 Introduction

The ability to move data between machines is an essential and evolving part of computing. As systems continue to grow in sophistication and complexity, so too does the volume of data which they are able to process and create. Because of this, methods of sending and receiving information between computing machines, at increasing speeds and in ever greater quantities, are necessary for efficient use of current and developing systems. A wide variety of devices and techniques sufficient for the needs of the past, current, and the immediate future are in used and are being refined. However, most are based on the same premise of sending 1s and 0s (binary language) in rapid succession, known as *serial transmissions*, over a wire(s) or by way of a broadcast signal, from one machine to another.

The innovative approach this paper introduces is called Chromatic Encoded Visual Data Transfer (CEVDT). This design differs from other techniques in that it is not based around traditional serial transmissions. Rather, it takes advantage of the vast improvements in digital imaging technology to chromatically symbolize binary data stored on one machine, and then reproduce it almost immediately on a separate, unconnected device(s).

2 Chromatic Encoded Visual Data Transfer

Our data transfer technique allows a machine(s) to acquire data stored on a separate machine, without the use

of a physical connection or contemporary wireless connections and is called Chromatic Encoded Visual Data Transfer. This innovative approach steps away from conventional data transfer paradigms, and comes from the assumption that there are still many unexplored methods for quickly and efficiently transferring data between machines. Specifically, this research looks at the human visual system for inspiration. The premise stems from Kupfmüller's law [3] which states, "Humans receive most information through the visual system..." By representing binary data in a visual form, a *chromatic data image (CDI)*, it is our assumption that the very rapid acquisition of physical information the human visual system achieves, can be imitated in machines with digital imaging technology, to acquire binary data.

At this time, the research is focused only on the time it takes to move data from one machine to another (i.e. how many viable bits/second are transferred), and not on any additional time it may take to encode/decode binary files and CDIs. This is because once the data exists on a system it is up to the user to decide when they wish to process it. However, encoding/decoding of binary files and CDIs are essential parts of the research; therefore, basic algorithms for these functions have been developed.

3 Current Technologies

While computers accomplish much more than simply moving information around, without their ability to send and receive data, they simply become elaborate word processors and calculating machines. Data transfer is defined as the movement of information from one location to a different location either internally or externally, between a computer and an external device(s). The amount of information moved over time is called the data transfer rate, and is usually measured in bits per second. For the research at hand, we are interested in the external transfer rate of data between separate devices. External transfers are generally improved through either data optimization or hardware enhancements.

4 Hardware

For our purposes, hardware describes all the physical machinery that is used to transfer data between computers

and the networks that connect them. Improvements in this area involve increasing the overall speed at which data is sent and received. They are divided into three general categories: external storage devices, wired technology, and wireless technology.

The term “external storage device” covers floppy disks, CDs, tapes, external hard drives, etc. With these, the time it takes to physically carry and connect to a different machine, plus the time it takes for the information to be accessed from the device is the transfer time. Moving very large amounts of data (giga/tera bytes) this way is often much faster than streaming data can accomplish over short distances. One example of this is moving music data between friends. Stored on an external hard drive of 100 gigabytes or more, transferring the data to another machine over a wired/wireless network would take hours, if not days, to send/receive all the data (depending on the speed of the connection). A much faster solution is to simply walk the external hard drive over and connect the external hard drive directly (this constitutes the transfer of data). However, over large distances, the cost of such physical transfers can often outweigh the cost in time saved. In these cases wired and wireless transmissions are the preferred methods.

Wired technologies cover all physical connections between machines and wireless technologies cover all signal based hardware that does not require a physical connection, for example: IR, radio signaling, satellite, etc. Both have good transmission speeds, with fiber optic cables having the best speeds. Wired and wireless technologies have the added advantage over external storage devices of allowing many machines simultaneous access to data. On the negative side, bandwidth limitations can greatly affect the speed of transmissions. Also machines not connected to these networks cannot take part in the data sharing.

Much less hardware is needed to connect wireless systems, but unlike wired systems, these are very susceptible to interference and disruption of signal by other wireless systems, physical objects/structures, as well as the environment itself. Security for both of these systems is a major issue; more so for wireless systems, because data can be more easily intercepted while in transit than with wired systems. With wireless systems, because it is necessary to broadcast the data, any system in range of a signal can “eavesdrop” on a transmission without the knowledge of the sender or intended receiver.

5 Data Optimization

Data optimization, as we refer to it, involves improving data transfer rates by decreasing the total amount of information to be sent in a transmission. Algorithms are used to analyze data, looking specifically for repeating patterns. Once analyzed, the data is compacted; shorter strings replace lengthy repetitions. In most instances,

information held on computers does not need to be optimized before it is sent to another computer. However, as the amount of data to be sent increases, the time required to complete a transmission becomes less tolerable.

Various compression algorithms are used to mitigate this problem. Generally speaking, compression algorithms are divided into two main categories: lossy and lossless. *Lossy* algorithms are so named because during the compression process, some of the data is actually, irreversibly “lost”. How much data that is actually lost depends on the degree of compression being applied. While this may seem unacceptable, lossy techniques have the best compression ratios, dramatically decreasing file sizes. These techniques are most often used in digital imaging, where other techniques have been developed to “approximate” the lost data, so that no data is apparently missing when an image is viewed.

Lossless compression is exactly what the name implies; no data is lost during the compression process. Since nothing is lost, this type of compression is completely reversible. These algorithms are useful in compressing all types of data. Naturally, lossless algorithms cannot offer the same degree of compaction as lossy algorithms, and their ability to compress a file is often affected by the type of data that is to be compressed. However, using lossless compression on data still allows for better transmission times than sending uncompressed data.

6 General Design

Our system makes use of advancements in digital imaging technology, specifically digital photography, which captures large amounts of data from a distance. This is coupled with the increase in rendering power of computers to create digital images and the standardization of visual imaging formats throughout the industry. Essentially, a file stored on a computer is selected. The binary data of the file is read by an encoding program. After reading the binary data, the encoding program creates a visual representation of that data in the form of either a monochromatic, or a multi-hued image (i.e. a Chromatic Data Image).

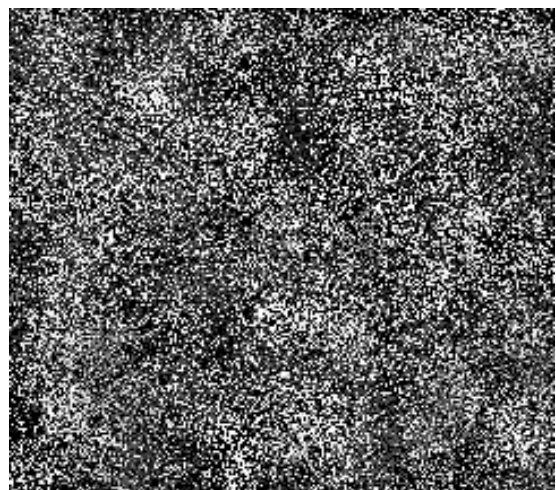


Figure 1. Visual Representation of Binary Data

Once a CDI is created, it can either be stored like any regular image file, or it can be displayed so that another machine can *acquire* the information.

To transfer the data, the CDI is presented on a monitor. A digital camera is then positioned to take a picture of the displayed CDI. The immediate acquisition of the CDI by the digital camera happens when the picture is taken. This constitutes the complete transfer of the data.

To validate that the data was successfully received, the picture captured by the digital camera is processed by a decoding program. The decoding program uses the values of each of the pixels or pixel areas to create a binary file identical to the original file selected at the beginning of this process.

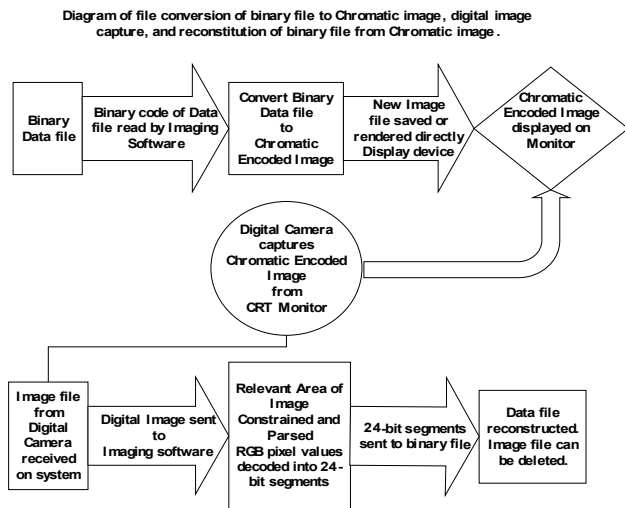


Figure 2. Map of Chromatic Encode Visual Data Transfer process

CEVDT also differs from most other transfer rate improvement method by how it approaches the problem. While other methods for improving transfer rates focus on only the software side, or the hardware side of the problem, CEVDT looks at affecting the issue of transferring data as a whole. Specific software is used to preprocess binary data to be transferred in a visual format, and then software is used convert a seemingly unintelligible image into useful binary data. Digital imaging hardware is essential to the transfer process, and is used to facilitate a very rapid movement of information between locations.

7 Transfer Projections

Prior to system testing we calculated the potential usefulness of this design. For our projections we used a

camera resolution of 640 x 480 because it is easily within the display ability of most monitors and the capture range of most digital cameras; most camera also support a low shutter speed of 1/8th of a second; and lastly RGB color space is used for the pixels:

- I. Resolution: 640 x 480 image = 307200 pixels
- II. RGB conversion:
24 bits/pixel x 307200 pixels = 7372800 bits/ (8 x 1024) = 900 Kbytes of data
- III. Shutter speed: 1/8th sec
900 / (1/8) = 7200 Kbytes/sec
7200 / 1024 = 7.03 Mbytes/sec

The wired and wireless transmission times are generally measured in bits/sec. Compared with bit rates of 1.5 to 9 Mbps for ADSL lines and 41.5 Mbps for T-3 carrier (twenty-eight T-1 lines) [9], the calculated 56.25 Mbps (7.03 x 8 bits) is an enormous potential increase given the relatively small image size.

8 Conversion Software

To our knowledge, there are no published works or experiments involving the uses of digital cameras to captured visual representations of binary code as a means of transferring data from one machine to another; nor have we found software used to encode and decode binary data as an image to be used in this manner. For these reasons it was necessary to design specialized software in order to turn standard binary code into a CDI and back again. The Java programming language was selected for the coding language for several reasons. It is platform independent; an important feature when designing a method of transporting data between unknown machines. It's large, and well developed methods base helped to streamline the coding process. Most importantly, while supporting a variety of imaging formats, Java uses a standardized coloring convention as its default color space know as "sRGB [8]".

One of the major problems with early digital imaging was that there was no set color standard which the computing community / digital imaging industry used. Different cameras taking pictures of the same scene would all product the same image, but with different coloration. Even monitors displayed images differently. In 1996 a joint proposal was put forward by Microsoft and Hewlett-Packard which defined a coloration convention, i.e. a *color space*, to be used as a standard for all digital imaging devices. This meant that all cameras, printers, monitors, etc. using this convention would now reproduce the same image with the same exact coloration. While this is not the only color space that the imaging community recognizes, the sRGB color space is widely accepted throughout the industry, and most imaging devices now support it. Since Java also supports sRGB, and uses it as its default color space, potential errors in

colorization from images created using Java are greatly reduced.

Several limitations and requirements were decided upon for this prototype system to create a more stable overall design:

1. The program will only be tested with ASCII text files. The wide usage of ASCII made this a logical source for our initial testing. Other formats can be added later
2. Size of test ASCII files should not exceed what could potentially be displayed on a 480 x 480 display. This size is supported by most monitors and is less than a standard input resolution size of 640 x 480 that most digital cameras support.
3. Test machines will be of the same platform. This streamlines the testing process, removing potential problems with subtle differences between platforms. A multi-platform enhancement may be added at a later time.
4. Monitor(s) used to display a Chromatic Data Image should support the sRGB color space (i.e. a factory default setting will adjust the monitor's appearance to comply with sRGB standards)
5. Digital cameras used in testing should support the sRGB color space.
6. The program will create a Chromatic Data Image (a visual representation of binary code) from an ASCII file.
7. The program will create an ASCII file from a Chromatic Data Image.
8. The Chromatic Data Image will be either a gray scale PNG, or a 24-bit color PNG (color images will be tested following successful testing of gray scale images).
9. A Chromatic Data Image will be square in shape and be composed of *pixel areas* of varying shades and/or colors. At this point in testing, the prototype system is not intended to deal with errors that arise from warp/skewed CDI. Systems to deal with homographic errors may be added at a later time.
10. A *pixel area* is defined for the program as a pixel or group of pixels in a gray scale image that visually represents one byte of binary code. In a 24-bit color image, a *pixel area* will represent 3 bytes of binary data.

9 Conversion from ASCII to CDI

This section details how the software converts an ASCII file into a gray scale CDI.

The process start with the selection of a file, if the file is not a text file, it is rejected. If it is accepted, the program notes the size of the file in bytes (ByteLength) plus two

addition bytes to be used for metadata. Having a valid file and ByteLength, several preprocessing steps are necessary before the CDI is drawn. Since the CDI is square in shape, the ByteLength can be used to calculate the dimensions (D) of the CDI to be created.

$$(1) D = \sqrt{\text{ByteLength}}$$

To ensure that the *total number of pixel areas* (TPA) is large enough to account for every byte in the ASCII file and the additional metadata bytes, D is rounded up the nearest even integer (D^a). This ensures that the value is not prime; therefore, there exists 2 numbers that multiply to equal D^a . This is necessary because it is possible that the value of D^a can be greater than 255. However, an 8-bit integer value can only range from 0 to 255, which is also the number of possible shades a pixel can take. For this reason two metadata bytes are added to the beginning of the CDI. During processing, by multiplying the integer values of the first two pixel areas in the image (byte 0 and byte 1), the system can determine the actual number of pixel areas in the rows and columns. D^a is thus equal to both the width (x) and height (y) of the CDI:

$$(2) \text{Byte } 0 \times \text{Byte } 1 = D^a$$

$$(3) x \times y = \lceil D^a \rceil \text{ such that } D^a \text{ is even}$$

$$(4) TPA = D^a \geq \text{ByteLength}$$

Since D is rounded up to the next even whole number the question of extra/unused pixel area arises. This is a minor point because the rate of increase of actual pixel areas used greatly out paces the increase in unused pixels area by the following equations:

$$(5) \text{ Give two integers } N \text{ and } M \text{ are even and } N < D \leq M \text{ and } M = N + 2$$

$$(6) \text{ The actual number of pixel areas needed} = D^2$$

$$(7) \text{ The number of pixel areas provided} = M^2$$

$$(8) \text{ The difference of } N^2 \text{ and } M^2 \text{ is: } 4N + 4$$

$$(9) \text{ The difference between } N^2 \text{ and } M^2 \text{ is: } M^2 - N^2 = 4N + 4$$

$$(10) \text{ Therefore, with the rate of increase in of needed pixel areas is } D^2 > N^2, \text{ and the increase in provided pixels areas is } M^2 \geq D^2, \text{ the number of unused pixels is at most less than } 4N + 4$$

The default pixel area for a byte is 2 x 2 pixels. The user can choose to increase the size of the pixel areas (PAsize) to be used for the CDI. At this point the preprocessing is

complete, and the program begins creating the CDI:

- (11) # of columns of pixel areas = x
- (12) # of row for pixel areas = y

Actual width and height of image in pixels:

- (13) width = x * PAsize
- (14) height = y * PAsize

Byte 0 and Byte 1 are first padded to the beginning of the image. Next, each byte of code from the ASCII file is encoded as a pixel area with a gray scale value from 0-255. Each new pixel area is added to the image from left to right, top to bottom.

Once all bytes from the ASCII file have been encoded and added to the CDI, any remaining pixel areas are encoded with zeros, essentially adding nothing to the file, but maintaining the square shape of the CDI.



Figure 3. Graphical User Interface

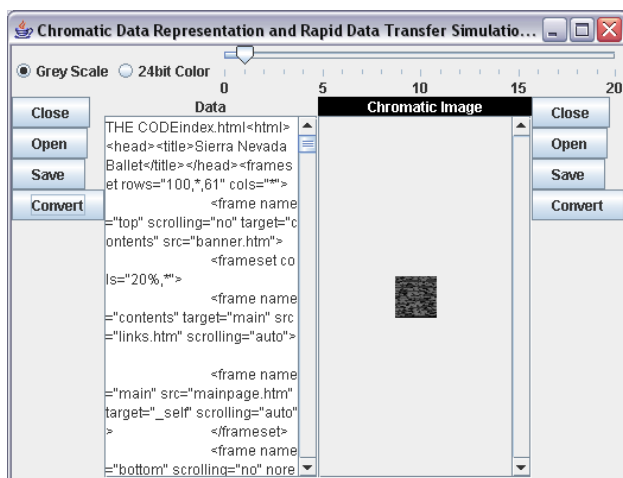


Figure 4. 1.16 KB text file converted to 1x1 pixel area, 35x35 PNG gray-scale data image

10 Conversion from CDI to ASCII

This section details how the software converts a CDI image into an ASCII file. Having captured a valid CDI image the user selects 3 points on the image prior to decoding of the image:

- (15) The first pixel area of the first row
- (16) The second pixel area of the first row
- (17) The last pixel area of the first row

The system records the position of the first and last pixel areas. Since the image is square the dimensions of the image can be determined from these:

- (18) top/left = first pixel area x & y position
- (19) top/right = last pixel area x & y position
- (20) bottom/left = first pixel area x, y = y position + (value of top/right x position - top/left x position)
- (21) bottom/right = last pixel area x, y = y position + (value of top/right x position - top/left x position)

The system learns the number of pixel values of the first and second column from the multiplied integer values of the first and second pixel area (D^b). The sampling step between pixel areas is determined by dividing the number of pixels in the row between the first pixel area and the last pixel area by D^b .

By getting the dimensions and number of pixel areas from the image itself using the above mentioned methods, the program automatically compensates for any changes in image scale due to distance between the camera and the monitor.

Finally the program creates a new binary file by moving left to right, top to bottom converting each pixel area integer value into binary, end result being a copy of the original ASCII file.

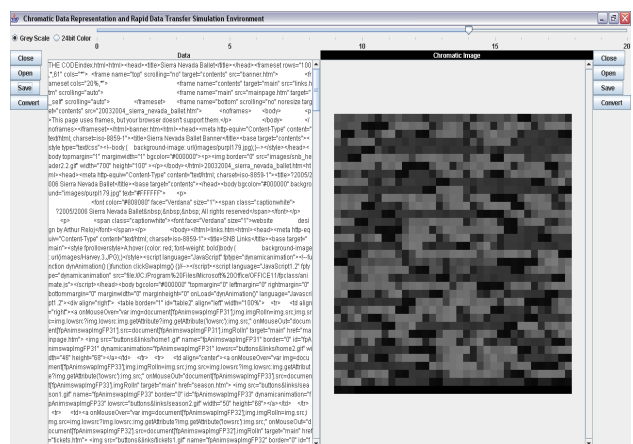


Figure 5. 1.16KB text file converted to a 13x13 pixel area, 455x455 PNG gray-scale data image

11 Observations & Future work

There are several characteristics of the system that distinguish it from contemporary transfer methods in terms of security and functionality. One reason it is secure is because it requires a “good” line-of-sight between the display medium and the capture device, this can be explained using a *highway* analogy with a one sided billboard on the side of the highway, as a display device. Anyone on the highway may know the billboard is there, but only those vehicles with a view of the image side could know that there is something to be seen. Moreover, before those vehicles that know that there is an image there can get information from the billboard, they must either be close enough or have sharp enough distance vision to make out all the details of the image. Most importantly, everyone within that scope of view can collect information from the billboard simultaneously, without having to slow down. This line-of-sight can be constrained so that only those who know where and when to look can see the image, or it can be expanded to allow everyone to see the image.

Another quality is that it is nearly silent compared with broadcasts from radio and IR systems. There is virtually no detectable signal sent out that can be notice by a system listening for a wireless transmission. Attempting to detect a transfer of this sort would require several conditions to fall into place; including knowing where the CDI is to be displayed and a good line of sight of it.

The immediate transfer of large blocks of data, if not entire files, is another security feature of this system. The short amount of time needed to transfer information helps to eliminate the possibility that the data can be acquired by an unintended third party. Moreover, the nature of the system, i.e. encoding of data as an image, adds a layer of encryption. Unless an individual knows what they are looking at, it is unlikely they will know how to decode the image. We are also considering algorithms that can be added for optical encryption. Lastly, this system may not be limited to visual spectrum images. By using other spectra as the basis, “invisible” data image transfers could be possible.

This system shows potential in other areas as well. One possible use is in the area of rapid programming, downloading, and uploading of data for robotic machinery. On an assembly line, rather than having to spend time loading data onto hard disks or establishing either cable/wireless connections for each robot (which also requires a fair amount of preloaded data and can be very slow if many machine are downloading data), the visual processing units of many robots can acquire all their programming data at the same time and rate (similar to a thousand people watching a movie in a theaters). Processing of the images collected can be done while the machine is still being constructed / transported with little

preloaded data, but without the need for further external programming support. Exploration and retrieval robots that gather data as well as physical samples, when docked to off-load their cargo, could display their collected data for the docking computer to acquire, and at the same time upload new instructions and programming with little or no time lost before heading out again.

Our research into this exciting new form of data transfer is only in its preliminary stages. Further tests will allow us to elaborate on any difficulties or advantages current digital imaging hardware has to offer.

12 References

- [1] “Welcome to Net Beans,” <http://www.netbeans.org/index.html>
- [1] Efford, N.; *Digital Image Processing: a practical introduction using Java*; Addison-Wesley, Pearson Education; 2000
- [3] Endres, A.; Rombach, D.; *A Handbook of Software and Systems Engineering*; Addison Wesley, 2003
- [4] Fulton, W.; PNG – Portable Network Graphics; <http://www.scantips.com/basics9p.html> Accessed May 8th, 2005.
- [5] *Java API Specification*; Sun Microsystems, Inc.; 2005; <http://java.sun.com>
- [6] Data Transfer Rates Explained; http://www.gse.uci.edu/301c/data_transer_rate_s.htm
- [7] Paul, R.J.; Eldabi, T.; Kuljis, J.; “Simulation Education Is No Substitute For Intelligent Thinking”; *Proceedings of the 2003 Winter Simulation Conference*, Dec. 2003, Volume: 2, pg 1989-1993
- [8] Stokes, M. (Hewlett-Packard); Anderson, M. (Microsoft); Chandrasekar, S. (Microsoft); Motta, R. (Hewlett-Packard); “A Standard Default Color Space for the Internet – sRGB,” Version 1.10, Nov. 5, 1996 <http://www.w3.org/Graphics/Colors/sRGB>
- [9] Data transfer rates; Created and maintained by PF.; June 13, 1999 <http://ls.berkeley.edu.lscr/support/faq/general/spee.html>