SUB\versions
A Journal of Emerging Research in Media and Cultural Studies

Shifting codes: Locating the Intersections of the Real and the Virtual Cultures of Photography

Ashwin Nagappa

Ashwin Nagappa works as Lead Technologist for a unique Education Technology project anchored at TISS. Graduated as Computer Science Engineer, he has experience of working for Multinational Companies in India and the US. His interest in photography and film making led him to pursue higher studies. He recently completed MA in Media and Cultural Studies from Tata Institute of Social Sciences, Mumbai, India. He is currently working on managing software development for high school students. His research focuses on the intersection of information technology, software, communication, and learning.

Abstract

Software and digital photographs induce the ideas of spectacle as they produce control. Irrespective of class locations all individuals are interacting with the above. There is certain reconfiguration in the nature of producing, seeing, and sharing photographs due to the intervention of software. The convergence of camera into a smartphone has defined 'sharing' as the default function of a photograph. This convergence is on one hand the progress in technology, on the other its nature has been determined as a consequence of neo-liberal measures that have come into place in the last two decades. Behind 'sharing' there are several algorithmic discourses (in turn defined by hegemonic discourses in the society) which govern our relationship with photographs and the new ways of the communication.

This study attempts to understand the relationship between users and digital photographs in a communication system based on the calculation (and transformation) of information, by looking at the process of producing photography software. It argues that the decisions defined in a camera software is driven by the hegemonic discourses

and institutions of the society, rendering digital images more than just a remembering tool.

Keywords: digital photography, software, algorithms, algorithmic discourses, hegemonic discourses, virtual cultures

---

Among the several neo-liberal changes we have experienced in India over the past 15 years, the way we look at things around us using electronic devices is significantly on top of the order. We are constantly using electronic screens in one form or the other, we are capturing images with small handheld devices or images of us are being captured by several CCTV cameras. We are now accustomed to experiences mediated through electronic devices operated by software. We live in an age where cameras, software and digital networks are ubiquitous. Many of these changes have come to existence after the political and economic changes across the world in the past three decades.

Photography has evolved over the years from an enterprise of science to an apparatus of science, it serves as an instrument in scientific experiments unlike the time photography itself was being discovered through science. It is also an instrument of art (Winston 1993), a surveillance/control mechanism, a producer of incontrovertible proof of things that occurred, and a social rite (Sontag 2008). In spite of a shift from a photo-chemical process to a photo-electronic process, photography retains its prime function of repeating mechanically "what could never be repeated existentially" (Barthes 1993:4). Enabled by software, these functions converge seamlessly in digital photography, transforming both the nature and the future of the form itself.

Digital photography as a 'way of seeing' has transformed how we understand the term 'image'. Mark Hansen (2001:58) defines the digital image as being beyond the "position of an observer in a 'real', optically perceived world". Discussing the digital image Crary (Crary 1990 cited in Hansen 2001: 58) notes, "If these images can be said to refer to anything, it is to millions of bits of electronic mathematical data". Digital camera in Winston's (1993) words is an 'instrument of inscription' which produces data for modern science. With the convenience to disseminate the

electronic data easily, digital photographs can be effectively used for the function of surveillance, power, social rite, etc.

Today, a smartphone camera can not only capture the light, it can also capture and transmit other metadata about the user and her/his location along with the image. Therefore, photographs in the age of digital imaging are not just 'experience captured' as Sontag (2008) opines. While experience is one aspect in the digital camera, other appropriations can be derived from the bits of electronic data in the photograph. Digital photograph allows appropriation of aspects lost to the optics of the camera allowing a larger degree of power to the possessor of information. While Sontag's (2008:5) description of photography as "a social rite, a defence against anxiety, and a tool of power", applies to digital photography, another aspect of power namely the control over data becomes significant. I would like to argue that a digital photograph can therefore be referred to as an assemblage, with multiplicities which "has neither subject nor object, only determinations, magnitudes, and dimensions that cannot increase in number without the multiplicity changing in nature" (Deleuze and Guattari 2013:7). This assemblage in a digital photograph allows it to be used as forms of control much as Sontag (2008:) points, "Photographs were enrolled in the service of important institutions of control, notably the family and the police, as symbolic objects and as pieces of information". Consumers of digital photography apparatuses are producers of information that is consumed by software elsewhere for purposes they are oblivious of.

Software

In the above discussion, there was a constant reference to the entity that enables images to be digital, i.e. the software. Electronic devices/cameras enabled by software are now ubiquitous. We inevitably interact with software in some form or the other. Chun (2011) elaborates how the hardware of a computer will only be useful when it has been programmed to work according to the instructions of its user. Software is necessary to make the machine work for our needs. As Chun (2011:19) puts it, "Software emerged as a thing — as an iterable textual program — through a process of commercialization and commodification that has made code '*logos',* code as source, code as true representation of action, indeed, code as conflated with, and substituting for, action". Thus, software is placed in a position of power. The knowledge-power structure can be considered as the core of the software.

'Software' as we refer to it here, is an assemblage of several coding languages, compilers and an executable code that allows users to run a series of instructions. Users of proprietary software can experience and consume the software, yet they can neither understand nor transform this assemblage completely. The ability to access and modify this software assemblage is restricted to a few people across the world, whilst the 'end products' reach a large more number of people. Software developers hence acquire positions of power in society. The hierarchy/control is subtle and accepted unquestioned.

Thus, the digital 'way of seeing' can be argued as seeing through the software or experiencing imaging through software rather than experiencing the images as they are, i.e. without applying any manipulation when the image is created (in a film camera the photographer had to decide the parameters to compose an image as compared to digital where the algorithm predefines the parameters). This way of seeing is internalized by the software developers by determining the ways of composing photographs using the camera software embedded in the electronic device, what I would like to refer to as digital camera assemblage. In this assemblage, I would like to argue, supplementary to the combination of optical components, electronic hardware and software, several social and cultural practices are algorithmically embedded. The intervention of software reconfigures the process of producing, seeing, and sharing photographs. This study attempts to explore the intervention and reconfiguration we experience in our interaction and communication with digital cameras.

Rationale

There are several studies that explore the relationship of software and photography, influences of software on photography or the life of a digital photograph; they do not make sufficient connections between the individuals or corporations who create the software and the individuals who consume them. Thus, it is necessary to study the nature of software development and factors that drive the consumption in digital photography, given that both software and photography are being driven across the world by few technology giants.

Stuart Hall (Hall 1997 cited in Rose 2007: 2) notes, "Primarily culture is concerned with production and exchange of meanings". "The algorithmically enabled interplay between the viewer's position in the physical world and this virtual information layer is transformative, creating sites of meaning and enabling action" (Uricchio 2011:33). These sites of meanings are governed by corporations and hegemonic forces through algorithms. Meanings are being

determined objectively, hence influencing our cultural practices, which are otherwise construed from our subjective experiences. It is therefore necessary to understand the role of algorithms in digital communication and their influences on our discourses or cultural practices.

Research Method

As a software developer and a photographer, I could locate myself in a space where I could draw connections between the two practices. Here I perceive software as an intervention rather than a tool of convenience because, my actions and experiences in photography were often determined by functions embedded into the camera. This study is based on semi structured interviews with two software engineers working on developing software for smart phone cameras. Bound by non-disclosure agreements these engineers have provided generic information about the steps involved in developing camera software. In addition, I analysed a basic OpenCV[1] Algorithm to understand the functioning of the algorithm in camera software.

Developing an Eye

Though the first computer was invented in the 1830s, around the same time as photography came into existence, it did not gain popularity like the latter since, computer unlike a camera did not reproduce reality yet, nor did it have utilities or purposes in human life yet. After over 150 years, computers could perform as a camera by capturing, processing light, and storing it as information. Photographs are created by capturing the light incident on a semiconductor sensor and storing them onto a storage device. Photographs as media became programmable since an image could be described as a mathematical function and could be subjected to algorithmic manipulation (Manovich 2001). This convergence of camera and computer has got us to a point where the photography device is almost invisible. Photographs can be captured by devices that are almost invisible to human eyes and at unimaginable speeds.

[1] The OpenCV library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV' ("ABOUT | OpenCV," n.d.).

Digital Camera Assemblage

Most electronic devices that are popularly used today such as digital cameras and smartphones are loaded with software that cannot be easily deconstructed and understood. Unlike early photographic technologies like Daguerre's device[2], digital cameras are much smaller and faster; thus, obscuring from human vision the process of capturing light. The assemblage in digital cameras (referring to both digital cameras and smartphones) allows us to communicate with the electronic hardware using human senses of vision, speech, listening and touch. Often an electronic device is practically of no utility if there is no software to make the components work[3]. Similarly, digital cameras in all forms today are assemblages of different kinds of hardware and software. With increasing magnitude and dimensions of hardware capabilities, complimented by software that can perform instructions at higher speeds the camera is able to achieve vision that can match the principle of accommodation of human eyes[4].

Keeping aside the hardware section of the camera if we are to look at the software alone, the structure of the software in itself is an assemblage of several languages, encoders, and compilers. The software that runs on a device is a compiled version of a program written using language such as C, C++ which can be interpreted in English. This is possible due to comments that are embedded in the source code, and also due to English-based commands and programming styles designed for comprehensibility (Chun 2011). These programs are a set of instructions which are to be executed by the device. However, the program in its original form cannot be interpreted by the electronic hardware, hence the C or C++ program is compiled in order to translate it into an intermediate assembly language and then into machine language. The final compiled program is the software that is embedded into the device, allowing the functioning of the device. Given the competition among camera companies, the final software product is encrypted so that the algorithms and functions cannot be deconstructed. Additionally, this encryption can conceal from the users the functions of surveillance and control embedded into the camera. Thus, the digital camera that we use to see the world is, paradoxically, itself a 'black box'; its functioning is hidden from view.

---

[2] The device was bulkier and involved chemical coated plates. Image was captured by the chemical process which took considerable time.

[3] Unless a sequence of steps can be programmed in the hardware using electronic components, it is unlikely to build functions of a camera using hardware programming alone

[4] The iris and the muscles of human eye adjusts itself to adapt to any given light situation to be able to see the surroundings.

We have arrived here due to the contribution of hundreds electronic scientists /engineers, thousands of software developers and millions of workers making components for these cameras. Studying the entire cycle, from the conception of a camera, to manufacturing of components, development of software and the eventual release and use of the product, could give a deeper insight into the influences of technology on the transformation of visual cultural practices. This paper will focus specifically on understanding the process of developing camera software since it seemingly has a significant influence on the visual cultural practices today. Dijck (2008:58) notes,

> *Recent research by anthropologists, sociologists and psychologists seems to suggest that the increased deployment of digital cameras – including cameras integrated in other communication devices – favours the functions of communication and identity formation at the expense of photography's use as a tool for remembering.*

Berry's (Berry 2008 cited in McCosker and Milne 2014:6) concept of code "as a literature, a mechanism, a spatial form (organisation), and as a repository of social norms, values, patterns and processes," could allow in understanding its implications on cultural practices.

Software Development

Given the challenges discussed earlier, gathering information from engineers was a difficult task. I was able to contact two engineers working on two different aspects of camera software. Engineer1 was part of a team that developed application which can be used to preview and click photographs remotely. Engineer2 was part of the testing team where camera software was tested before it was embedded into the smart phone.

Engineer1 is part of a team which had earlier developed a camera trigger[5] app[6], to remotely control a camera using a smartphone. He is continuously involved in developing applications around the camera component for smartphones which can be downloaded and installed by users themselves. There are several stages of development, each of them handled by different teams. Broadly the development begins with the research and requirements gathering team, they define

[5] Trigger here refers to remotely releasing the camera shutter. Earlier the triggers were wires attached to camera or wireless radio frequency remote control. In digital photography, a trigger is a software in which the photograph can be previewed

[6] The software applications that are running on the portable devices are being refered to as app.

the work to be done. It is then passed onto the development team who develop the software. The developed software is then tested by a testing team, who test for the accuracy of the functions. The software could have multiple iterations of development and testing before it is finally given to the marketing team for dissemination.

Engineer1 was able to describe the entire cycle of development since he was also overlooking the execution of the complete workflow. Before any new app is developed the research team would identify the potential features or functions that could be built into software which is not already present. Their research includes an in-depth analysis of apps already available in market and information gathered from the Internet regarding the same. The method employed by this team was not exhaustive; they were only looking to develop features that would be new and could be extensively used. The larger goal of the requirements team is to develop a software product that would profit the organization, without much consideration of its social or cultural implications.

Based on the requirements, developers check the feasibility of building the final software. Factors such as hardware abilities, firmware versions, available APIs (hardware and camera manufacturers often offer software to configure the hardware), primarily determine the feasibility of developing the software, apart from the cost and labour considerations. If the project is considered feasible, the development process begins. Most of the development is being done using Object Oriented C or C ++ languages. In some cases, the SDK[7] (software development kit) is provided by the hardware firm. In the process of development, the engineers develop use cases (potential user profile), develop flowchart and information architecture before these artefacts are coded in the necessary programming language[8]. Once developed, the program is compiled into an executable format and tested in a local environment[9] for errors. The primary objective of the development team is to develop a program that can carry out the specified number of functions defined by the requirements team.

Once the development is finalized, the program is given to the testing team to test the features and check its compliance to the requirements determined in the initial phase. The first phase of

[7] Software Development Kit (SDK) is a collection of software used to develop applications for a specific device or an operating system.

[8] The software backend that executes crucial functions are developed with C or C++. A frontend interface to access these functions is developed using programming languages like Java, HMTL, CSSS etc.

[9] The programming is done using a programming interface in which code can be written in a particular language, compiled and tested to see if the program runs as expected. Small units are coded first and tested independently to check if they give appropriate output, only then it is integrated with other modules.

testing is conducted in a simulated environment, where a computer program behaves like the camera hardware[10]. The software is then tested on the final hardware where it is supposed to be used. The testing team also checks if the software products carries out the specified functions as required or not. A visual expert is consulted once during the testing to get a final approval with regard to the quality of the output. Throughout the process this software is perceived as a product rather than a visual tool. The development is done to build functions that will be used as visual tools in photography. The software product is tested to check if it performs within the stipulated time to produce machine vision that resembles human vision.

Two significant components of the process were, developing image manipulation features and social media integration. Basic image manipulation functions were built into the software by using ready to use modules. And as a default, features such as Facebook and Twitter buttons were integrated into the app. thus providing the convenience to click, edit, and share the photograph on social networking platforms. This enhances interactivity since, in a networked environment, users not only interact with the media but also with other users using the media. As Engineer1 says, "it is one of the essential elements a camera app is expected to have".

Engineer2 was a testing engineer who was concerned only with testing the modules he was assigned. He worked primarily with camera software on smartphones, thus limiting the scope of the development to the specified hardware. He would be given the specifications of the software and the expected performance. All the expectations of the software are quantitatively defined. Several parameters such as camera on time, shutter snap time and colour levels are all precisely measured. Only when all the parameters meet the required levels, the software is considered ready for launch. Throughout this process the camera is never physically used. The camera hardware is determined based on the design and requirements identified in the earlier stage. The software development does not wait for the hardware to be ready; software is developed in parallel to the hardware development. As Engineer2 says'

> *Developing app is the last task we do. We begin with hardware abstraction. We are more concerned with the APIs and function calls. We are also dependent on the Software Development Kit (SDK). Depending on the requirements and hardware specifications we need to determine what function call corresponds to capture photograph or record video or stop the recording.*

[10] Electronic devices that are themselves computers with powerful CPU (central processing unit), RAM (random access memory), storage and display, can be simulated on other computers with similar or better specifications. One computer can, therefore, be used to simulate several camera models.

Based on hardware specifications a camera is simulated on a computer which would behave as the final camera is expected to. The software can work independent of the optical components of the camera. This substantiates the earlier argument that photography is now an electronic process. Once the software is considered to be good enough, it is loaded onto the hardware and tested to check the actual performance. It is then tested by photographers or experts on design, before the smart phone is launched into the market. While Engineer2 was only concerned with the testing of the camera component, the device goes through a series of developments and tests that define the other functions of the smartphone. This device could deceptively become a mechanism of control. Software can thus encapsulate our actions and experiences as metaphors, thereby proliferating mediation.

Two ends: Engineers and Photographers

Both the developers called themselves 'casual' photographers who often took photographs with smartphone camera. Talking about hacking camera software, Engineer2 states, "perhaps only a few curious engineers would work on hacking the camera software to try new things. I am not so keen about photography, so I do not think so much in terms of hacking the software".

Both engineers were confined to a standardized process, and their task was to develop standardized products. Though these products which contribute to visual cultural practices, in this standardized environment engineers carry out tasks as they are expected to, without reflecting on the broader functions of the software. In the case of both engineers very little consideration given to visual cultural practices while the software was developed. Though contemporary visual practices induced by technology feedback into the system for developing newer software products, it is however largely within standardized framework defined by profit oriented technology or industry.

Given this nature of software development, definition of variables is restricted to the individuals who define the requirements to the developers. The variables defined by them are largely influenced by their cultural location which is propagated to many other individuals who use the software. The sense of freedom or choice is much limited; perhaps it imbibes values into users as defined in the software.

There are attempts in the free software movement[11] to develop software not driven by profit. Though the free software movement attempts to gain political importance, they cannot match the hardware production or integration as done by big corporations. With the availability of low cost labour (Fuchs 2014) in countries like India, software firms are growing at much higher rate as compared to the growth of the free software movement through a collaborative network.

Reuse: Programs and Programmers

It is interesting to note that a software code is rarely written from scratch. Many re-useable programs and modules are assembled together to build the desired product. Object oriented languages are now widely used since the language allows us to write several modules independently, each of which can perform different functions depending on the context in which it is invoked. Additionally, there are several modules to perform some standard operations built into the library[12] (basic functions of mathematics, graphics etc.).

In the context of reuse, coding includes knowing what exists in the library that can be used to achieve the new functionalities. One such library called OpenCV (Open Computer Vision), that was developed by Intel Corporation and made available in the public domain, contains several programs and algorithms for the functioning of a camera., a major technology multinational developed and shared OpenCV (Open Computer Vision) code in public domain. It had several programs and algorithms for the functioning of camera. Engineer1 mentioned that OpenCV is one of the libraries they refer to while developing the software. With modular programming practice small functions can be developed and tested independently and these smaller modules can be integrated into one large program by joining them like pieces of a jigsaw puzzle. Often, for software developers the social functions of the software[13] or its larger consequences are not primary concerns. They simply follow instructions given from a superior. An independent developer or smaller firms have the freedom to think about broader perspectives or larger consequences of their software. Here I would like to argue that developers can themselves be treated as re-useable modules that can be invoked or replaced as per the needs of the product or

[11] Free software movement is a social movement started by Richard Stallman to develop and disseminate software that ensure user's freedom.

[12] A predefined program/s or piece of software code that performs a defined set of functions and available for reuse.

[13] Individuals interact with software in many ways directly or indirectly, like billing systems, Kiosks, Smart phones, Government systems.

business[14]. Engineer1 or Engineer2 are trained such that they have certain competences to perform specified tasks. They are one among the many who can code the program as needed for the software product. One can be replaced by the other conveniently to keep the development process going or keep the product functioning over its lifetime. There is a certain level of abstraction with respect to the labour aspect of software development. In fact, technology can be rendered so efficient that a significant part of the development or testing process can be automated, eliminating the need for engineers to intervene. In contemporary coding practices, use of graphical user interface to write a code[15] reduces the effort required for coding, subsequently reducing the overall cost of software development. These abstractions are "compatible with the overall trajectory which governs computers development and use: automation" (Manovich 2001); the nature of automation is governed by the hegemonic structure persisting in the society. In the Indian context, I believe the functions of software and the process of automation would be defined by the upper class and upper caste section of society. Their political and economic privilege empowers them to define the functions of technology and the patterns of consumption. Though individuals from all classes are influenced by software directly or indirectly, only the higher classes/castes are reaping the benefits of technology. There are fewer avenues for a large section of lower class/caste individuals to indulge in software development. I believe even if a small number of software developers from lower class and lower caste manage to be part of the system, they have to adhere to the framework defined by the upper class/caste individuals. Also, they retain attributes of the former only to strengthen the hegemonic structure. In an Indian context, the "disciplining of programmers" (Chun 2011:35) disempowers the developer from using technology to address social issues or inequalities, whilst strengthening corporations and hegemonic frameworks. Though I would like to argue about the presence of caste hierarchy in the software industry, I cannot prove the same empirically in this study. A dedicated study introspecting the caste and class economy of software developers could substantiate this argument or bring forth new findings.

---

[14] 'In US crisis management and crisis communication literature and research, the connection between crisis and risk is framed not as how to protect people from organisational disasters—the catastrophic events that harm environments and people—but rather how to protect capital or the organisation, its managers and their reputation from the disdain, anger and rage of those who have been wronged or even hurt, within or outside the organisation. In this way organisational crisis management moves quickly through crisis communication to seek recovery and resolution, the restoration of its image and profitability' (McCosker and Milne ibid.)

[15] For Example: Android's Intelligent Code Editor is described thus: 'At the core of Android Studio is an intelligent code editor capable of advanced code completion, refactoring, and code analysis. The powerful code editor helps you be a more productive Android app developer. There are several other coding platforms for other coding languages as well' ("Download Android Studio and SDK Tools | Android Developers," n.d.).

Though technology gives individuals the freedom to develop their own solutions, popular software products/services determine the patterns or features[16] that are being developed. Hence camera software developed by companies like Canon, Nikon, Apple, Samsung, Microsoft etc. hegemonize the industry. They are constantly working to converge the single purpose digital camera into a smart phone[17]. Therefore, I would argue that these global camera and software manufacturers influence the way visual cultural practices are being shaped across the world. Unlike a film photograph, digital photographs from a certain model of camera would have certain sameness. A photograph developed on film with same model of camera would not have the sameness since, the chemicals used to develop and paper used to print the photographs would vary from one place to another. With a given model of the camera the film being used varies. Similarly, when a film is being developed the nature of chemicals also varies. And later when the film is developed into a print the kind of paper and ink will differ. It is almost as if each photograph on a film or print is unique. However, certain camera/camera phone model, will always produce same result each time. If two identical digital cameras are used side by side the results produced will be same. This sameness in a digital photograph, I would argue is a result of the algorithmic discourse that persists in the system. I would define algorithmic discourse as a series of events that attempts to achieve the same result for a given set of conditions, by repeating a finite number of steps with precision. Though developing a film also involves repeating the same number of steps the external conditions or the precisions cannot be repeated every time, it does not behave algorithmically. Therefore, I believe algorithmic discourse can be considered as blinkers that restricts our visions.

Blinkering Vision

In digital photography unlike the chemical reaction on incidence of light, light incident on a sensor (a semiconductor device) is captured and translated into zeros and ones (rather, the light is captured as an electronic signal which is represented with these arbitrary values). As Hansen (2001:60) notes, "The digital image has only an 'electronic underside' which 'cannot be rendered

[16] Most photography apps today come with readymade filters and standard editing functions. Camera controls are seldom manual on smartphones nowadays. Easy to use/one touch features are introduced by global leaders in phone manufacturing such as Apple, Samsung etc.

[17] Samsung's Galaxy camera is a suitable example. It runs on Android (software running on smartphone) it promises faster connectivity and easy sharing ("Samsung Galaxy Camera Price India, Galaxy Camera Features, Specifications," n.d.),

visible' precisely because it is entirely without correlation to any perceptual recoding that might involve human vision".

The electronic/binary data has to be recoded back into a representation that is recognizable to human vision. Computer vision achieves this by passing the binary information through a series of steps so that the binary information can be transformed back to seem like a natural scene on the display. Perception is disembodied into a device (Hansen 2001) through the automated vision achieved in digital camera assemblage (receiving light at the sensor and displaying on a screen), whilst producing 'an acceptance semblance of reality'[18] (Goldsmith 1979 cited in Manovich 2001:171).

In automated vision, the sensor is made up of units called pixels. Light incident on each pixel is saved as binary information into a storage unit designated as a pixel. Information for each pixel is processed through a series of algorithms before they are represented on a screen constituted of pixels. The algorithmic process is crucial since the light captured on the sensor has to be transformed into a form that can be emitted through the screen. "Algorithms take many forms, and no digital camera or computer-based viewing system would be complete without them" (Uricchio 2011: 31). Algorithms are a sequence of steps where the pixel is subject to a mathematical function and a numerical value is assigned to it. The pixel is processed by several algorithms. The decisions of photographic parameters are mapped mathematically in the form of algorithms, and are applied to all the pixels. The decision-making mechanisms are embedded into the device. If one is using a camera in automatic mode, the camera determines all the parameters for the user. In manual mode, the device adapts to the exposure, white balance, metering determined by the user, nonetheless algorithms function here as well. While algorithms that control exposure can be experienced through the interface, there are several other algorithms that run behind the scene to enable the computer vision. Therefore, it is almost as if an algorithmic discourse determines what we can see and capture through the screen of a digital camera.

Edge Detection Algorithm

One of the essential algorithms in a digital camera would be the edge detection algorithm. Since, "if the edges in an image can be identified accurately, all the objects can be located, and basic

[18] Manovich draws from Goldsmith's initial discussion of the term is in relation to film industry.

properties such as area, perimeter, and shape can be measured. Since computer vision involves the identification and classification of objects in an image, edge detection is an essential tool" (Parker 2010:21). Detecting edges is one of the initial mediated steps in the process of seeing in the digital camera. Given this importance of detection of edges, there are several edge detection algorithms. Each of them involve defining the edge (the output/result to be achieved), defining different variables that are necessary to prepare and solve the equations, eventually to be able to detect edges. Developing an algorithm is purely a mathematical process. Digital camera algorithms create the possibility of seeing, through a mathematical process. The objective of these algorithms is to make a computer capable of reproducing the experience of perception by human beings and to match computer vision with human vision to the extent possible. Only then the photographs could perhaps come close to 'real' and attain its functions. One would always prefer a sharp photo over a pixelated phot. An edge detection algorithm is therefore important for mapping the edges appropriately.

Among several edge detection algorithms Canny edge detection algorithm is known to perform optimally and it is widely discussed (Parker 2010). By analysing the flow chart and the code for Canny edge detection algorithm, I attempt understand partly the algorithmic discourse that is embedded into the device.

The edge detection algorithm detects the edges by estimating the pixel density and narrowing down the edges based on the gradient. Canny edge detection algorithm addresses three issues that persisted in the earlier algorithms in order to optimise the performance in detecting edges. It ensures all edges are detected without missing one. There is little deviation between the edges detected from the actual edges. And multiple edge pixels should not be detected in place of single pixel edges.

The flowchart in Fig. 1 shows the broad sequence of steps a photograph or information passes through. Every scene we see on a digital camera display is a photograph in itself, but it is never saved until we press the shutter release. The continuous scene we see through a screen is constantly being processed within the device to produce images that appear real. In the first step of the flow chart the excess information, called noise is reduced so that the processor can easily isolate the pixels at edges in scene.

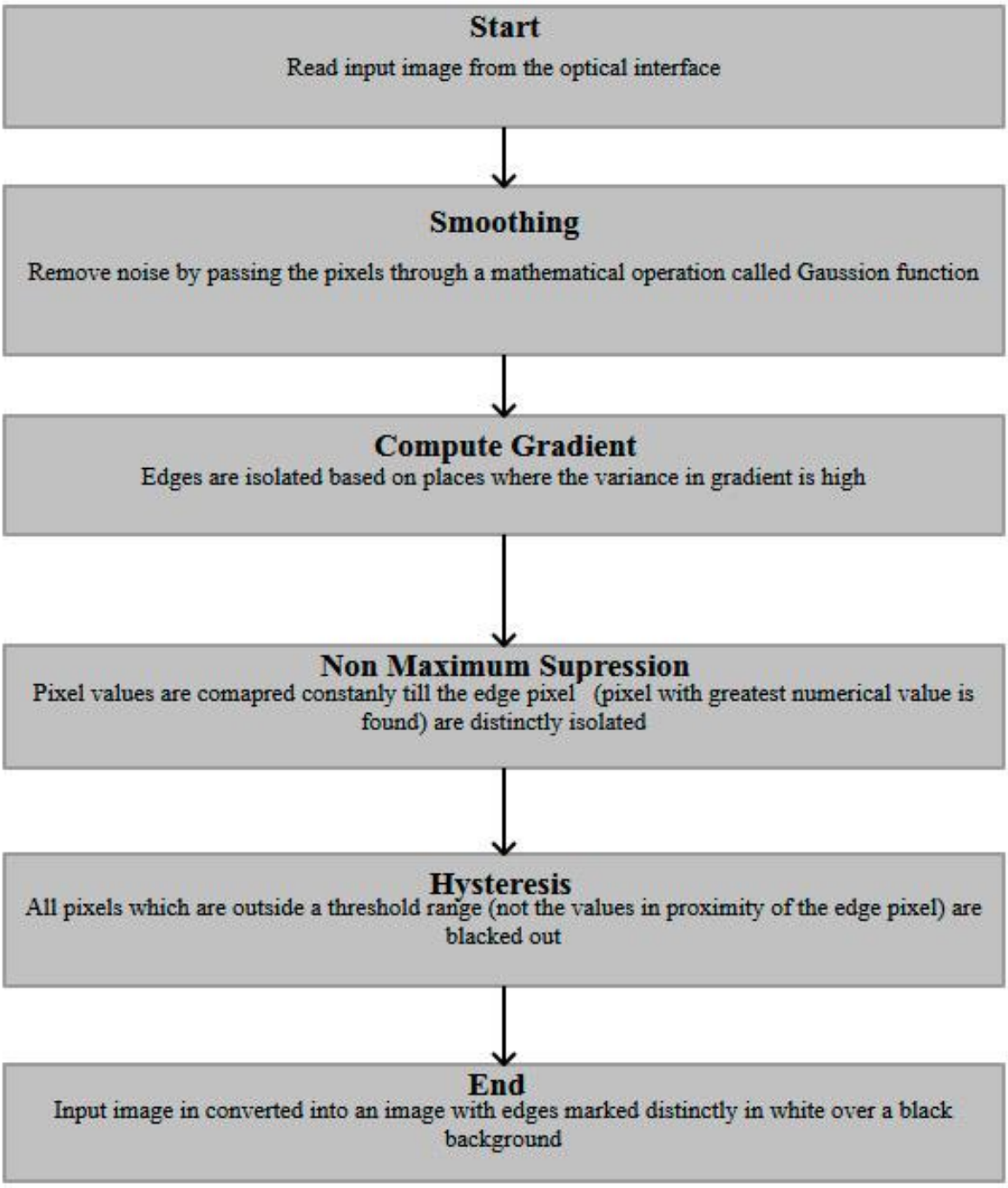

Fig. 1

This is done by passing all the pixel information of the photograph through a mathematical function that appropriates the numerical values of pixel, so that it assigns a higher number to the

edge pixel as compared to other pixels that do not form an edge. This noise reduction process is repeated again in the following step of the algorithm to further reduce undesirable elements.

Once unwanted (as perceived by the computer based on pre-set values) light information is reduced, in the next step the edges are isolated further and non-edge pixels are blackened. An edge is identified by recognizing a pattern in the scene. When light is incident on an edge it forms a solid line of a certain colour. The adjacent pixels get lighter as we move away from the edge. An edge pixel would have a greater numerical value as compared to the others. Each pixel in the photograph is passed through a mathematical function that compares the numerical value of all neighbouring pixels and continues processing in the direction of pixel with a greater numerical value. This is repeatedly done until pixels with maximum value is encountered, which is marked as an edge pixel (changed to colour white, since white has the largest value on RGB[19] scale). While the process keeps searching for the pixels with maximum numerical value, other pixels with lesser value are set to black. Thus, pixels on the edge are isolated as white lines on black background.

In the final step of Hysteresis the photograph (edges marked in white) is passed through a clean-up process. Two threshold values are defined; one, higher threshold, all pixels with value greater than the higher threshold are edge pixels. Two lower thresholds, all pixels above the low threshold are also associated to the edge pixel. Any pixel which has a numerical value less than this range is set to black. This process is repeated multiple times in a loop, till perhaps the difference between high and low threshold is minimum or till the scene changes. The final output photograph contains edges sharply distinguished in white over a black background. This output photograph can then be sent across to other algorithms depending on the flow determined. For example, a face detection or pattern detection function can use the edge details to determine the face or patterns within a photograph.

The algorithm when implemented on a C program works using a 'for' loop. For each pixel, the functions are applied to determine the edge pixel. The whole function could run within fraction of a second. The function based on comparison attempts to isolate the edge pixel by running in a loop till certain conditions are met. Often the conditions for the 'for' loop is determined

[19] RGB: Red Green Blue, the primary colours. In computers, each is represented by a number in the range 0-255 (2^8, 8 bit processing). The lowest colour in the range is black (0,0,0) and the highest colour is white (255,255,255).

dynamically based on the photograph in question. If the conditions are not determined correctly the algorithm would not function fully.

Examples

The descriptions of the steps involved in the Canny edge detection algorithm gives a sense of its function, but it can be better understood by seeing the results of the algorithm on a screen. OpenCVd2[20] , an application on android phone displays results of several OpenCV algorithms. Using this tool, I have explained the working of the Canny edge detection algorithm

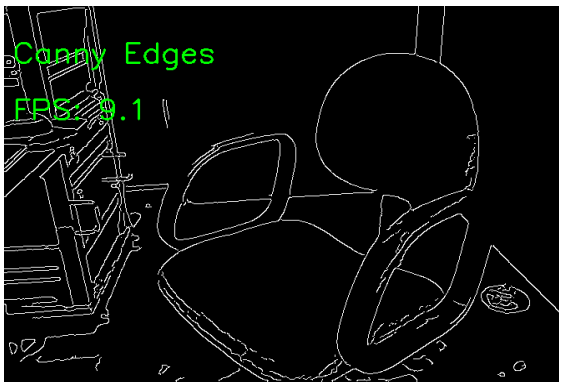

Fig 2.a

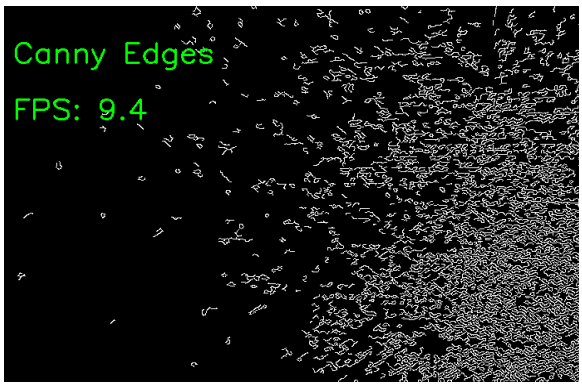

Fig 2.b

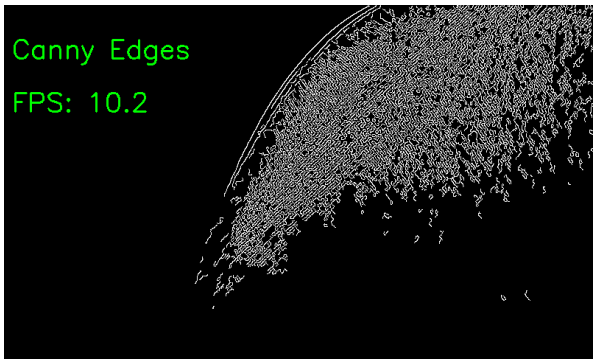

Fig 2.c

Here, Fig 2.a is a wide-angle photograph of a studio chair. The edges are more or less accurately marked. However, in a narrow shot of the chair the fabric seems to have much finer edges as seen in Fig 3.b. The fine edges are visible to the human eyes from the same distance where wide photograph was taken, but the algorithm cannot isolate them from that distance. Only when the

[20] Developed by Barry Thomas the app is available for free download, it 'takes the video feed frame by frame, processes the data and overlays the results onto the frame before displaying it on the screen' ("Machine Vision," n.d.).

camera is closer to the subject, the edges of the fabric are determined. Basically, the algorithm does not work to match the principle of accommodation of human vision. In case of a single input in Fig 2.b, the scene is constantly processed till all edges are identified. For instance, in Fig 2.b the edges of the threads have partially appeared. If the camera was held for a longer time, other threads would also emerge slowly. Fig 2.c is taken by holding the camera close to the chair and including the edge of the chair, the threads are clearer here. The input is repurposed constantly till a satisfactory result is achieved, the algorithm can never give an absolute result either i.e. computer vision cannot match human vision yet in form and function. The mediation is limited to definitions and capabilities of the device. This limitation of the algorithm results in an image that lacks an output recognizable by our vision.

Further, social networking platforms such as Flickr are developing new algorithms to identify the content of a photograph beyond the metadata. The new product running on Hadoop[21] can recognize the shapes or objects with in a photograph, this helps in tagging the photographs with appropriate tags so that the photographs have better search visibility. To train the system to recognise the shapes/objects correctly a sample set of valid cases are created and another larger set of invalid cases are prepared. The algorithm runs through both the sets and produces a set of features to validate certain shapes/objects. For example, to train the system to recognize flowers 10,000 valid examples are processed and 100,000 invalid examples are processed. Based on the calculation a set of features (shape, dimension, texture etc.) are listed down and saved to the database. Every time someone uploads a photograph of flower, the features of that photograph are mapped onto the list of features in the database and it is tagged as a flower automatically if certain predefined features match. The algorithm is perfected by processing many invalid examples to train the system to recognise what is not a flower. The scales of both the sets of examples are huge, hence letting the algorithm develop a comprehensive set of features.

A Conditional Realm

Manovich (2001: 60) defines algorithm as a "sequence of steps to be performed on any data...which potentially can be applied to any set of media objects". It is also a sequence of decisions that is applied to the data depending on the conditions that are met. Algorithms can thus be defined as a set of rules that is applied on data, rules that interpret the binary information

[21] 'The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing' ("Welcome to Apache™ Hadoop®!," n.d.).

and present them to us in forms we can recognize. A past moment in time can be captured as a digital photograph, giving that moment a material existence, whose form is determined by the algorithms that are embedded into the camera. Algorithms enable the processing of light information to be interpreted as a moment in time, thus making software a metaphor "for the mind, for culture, for ideology, for biology, and for the economy" (Chun 2011:55). Depending on the decisions that a data passes through it could attain a different meaning. It can be perceived as a different metaphor.

Relatively our lives are highly conditional, where all events are consequences of a series of factors or other events. Unlike algorithms, there are no concrete definitions of the conditions we come across. Therefore, by using devices with algorithms embedded in them, our experiences are being mediated based on a finite set of conditions defined in the algorithm. The number of possible outputs (metaphors) is also finite. By separating the interface from the algorithm software becomes unknowable, making it a metaphor for a metaphor as Chun (2011) argues. In contrast, I would like to argue that by obscuring the algorithms (conditions, decisions, and sequence of steps) software is made unknowable. Since the algorithm (mathematical equations and programming language) cannot be accessed and interpreted by everyone. It is not perceived as new media object like a digital photograph (light information captured, stored and appropriated). Algorithms cannot be consumed like a new media objects, nonetheless our actions and consumption patterns are constantly being guided by them. A knowledge practice that makes the algorithm visible is built out of the present practice of digital photography. To give an example, as a user to compose a digital image I would consider a limited number. Starting with turning on the camera, composing the frame through the digital display or eye piece, click a button and preview the end image. Within this process, the camera software would calculate the light and colours, and adjust the settings to the given light. There are several decisions the camera could manage. Additionally, more recent cameras can capture location details, the smart phone cameras could capture the user's details/login. This information allows easy sharing across several social media platforms. Though this is a convenience for the end users, it also means allowing the machine to take several decisions on the users' behalf. Further, this makes users vulnerable to systems and mechanisms of control.

The digital camera is an assemblage of algorithms, since algorithms are evolving to map human experiences mathematically. Moreover, the increase in the number of algorithms used in programming has drastically transformed human computer interaction. If we look at the stakeholders in the process of developing camera software, multiple individuals involved in the

process are not likely to have the nuanced eye that experts would possess with respect to light, colour or dynamic range. With an algorithmic discourse, the visual culture practice is reconfigured depending on the stakeholders or hegemonic forces that define the algorithms or rules of perceiving through a digital camera.

Entrusting the decisions making apparatus algorithmically to a system, makes it convenient for an institutional hegemony rather than allowing to "democratize all experiences by translating them into images" as Sontag (2008) writes. Especially, entrusting the functions of seeing and remembering in algorithms has made possible for the growth of several social networking platforms and Internet services, which is now hegemonized by technology giants like Facebook, Google, Yahoo etc. Only a tiny section of the population have a stake in these large organisations, whilst a larger section of the population are users of such technologies.

References:

Android Developers. Download Android Studio and SDK Tools. [WWW Document] (n.d.) URL http://developer.android.com/sdk/index.html (accessed 2.23.15).

Barthes, R. (1993) *Camera Lucida: Reflections on Photography*, Random House Publishers India Pvt. Limited. New Delhi.

Chun, W. H. K. (2011) *Programmed Visions: Software and Memory*, MIT Press.

Deleuze, G, Guattari F. (2013) *A Thousand Plateaus: Capitalism and Schizophrenia*. Bloomsbury Publishing India Private Limited. London, UK New York, USA.

Dijck, J V. (2008) "Digital photography: communication, identity, memory", *Visual Communication*. 7: 57–76.

Fuchs, C. (2014) "Theorising and analysing digital labour: From global value chains to modes of production",. *The Political Economy of Communication*. 1,2. Available at: http://www.polecom.org/index.php/polecom/article/view/19/175 [Accessed: 21st February 2015]

Hansen, M.B.N. (2001) "Seeing with the Body: The Digital Image in Postphotography", *Diacritics*. 31(4): 54–84#.

Manovich, L. (2001) *The Language of New Media*. MIT Press. Cambridge,Massachusetts London, England.

McCosker, A., Milne, E. (2014) "Coding Labour: *Cultural Studies Review*. 20 (1) . doi:10.5130/csr.v20i1.3834: 4-29

OpenCV (n.d.) "ABOUT", Available at: http://opencv.org/about.html [Accessed: 21st February 2015].

Parker, J.R. (2010) *Algorithms for Image Processing and Computer Vision*. John Wiley & Sons. Indianapolis.

Rose, G. (2007) "Visual Methodologies: An Introduction to the Interpretation of Visual Materials". SAGE. London.

Sontag, S. (2008) *On Photography*. Penguin Classics. 01 Edition.

The Apache™ Hadoop® project (n.d.) "Welcome to Apache™ Hadoop®!", Available at: http://hadoop.apache.org/#Who+Uses+Hadoop%3F [Accessed: 21st February 2015] .

Thomas, Barry (n.d.) "Machine Vision", www.barrythomas.co.uk Available at: http://www.barrythomas.co.uk/machinevision.html [Accessed: 23rd February 2015].

Uricchio, W. (2011) "The algorithmic turn: photosynth, augmented reality and the changing implications of the image", *Visual Studies* 26(1): 25–35.

Winston, Brian. (2012) "The Documentary Film as Scientific Inscription". In: Renov, M (eds.): *Theorizing Documentary*. Routledge New York, pg. 37-57.