IDC HERZLIYA | Efi Arazi School of Computer Science

The Interdisciplinary Center, Herzlia
Efi Arazi School of Computer Science

Evaluation Tool for Halftoning Algorithms

M.Sc. Dissertation

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science (M.Sc.) Research Track in Computer Science

Submitted by Ariel Shmerling

Under the supervision of
Prof. Yacov Hel-Or and Dr. Carl Staelin

October, 2015.

Acknowledgments

I would like to express my deep and sincere appreciation to Prof. Yacov Hel-Or and Dr. Carl Staelin for exposing me to such a wonderful domain. Working on this research project was a very enrichment experience, which I could not perform without their assistance. Their knowledge, experience and profound understanding of the field led to excellent guidance, advises and references, and helped to bring this project into conclusion. During this journey, I learned from Prof. Yacov Hel-Or and Dr. Carl Staelin so much about Digital Halftoning in particular and Image Processing in general.

I would like to thank to Dr. Hila Nachlieli and Dr. Mani Fischer for their kind support, advices and patience. This assistance is very much appreciated.

Finally, I would like to thank my family for supporting and allowing me to focus on the project in order to achieve this goal.

Thanks You!

Abstract

Digital Halftoning is a well-known technique used to create the illusion of a continuous tone image that uses a smart arrangement of binary elements, and is widely used in the printing industry. Many approaches have been defined for both grayscale and color images. Each approach has advantages and disadvantages, but there is no single algorithm that fits all purposes. In fact, there is no general objective measurement for evaluating Halftoning algorithms. The main problem is that some algorithms can look very appealing to the designer and to the developer of the algorithm and look very good in the lab environment, but when using it in the real-life environment, it yields unexpected and unwanted results. This often happens due to the fact the designers generally assume perfect printer, but in reality, the printers have many distortions. Each halftoning algorithm is sensitive to a different set of distortions. The goal of this work is to build a new tool, which aims to address this issue by simulating the printing process of real life printers including their main known distortions. The tool measures the sensitivity of the target algorithms to these distortions. In this work, we measured six algorithms from various categories, executed the evaluation tool and demonstrated the results of these examinations.

Table of Contents

Introduction

Digital Halftoning

In this work, we widely deal with halftoning algorithms. Halftoning algorithms are a family of image processing algorithms, which are used to reproduce an image with devices with limited number of tone levels (usually bi-level) [27]. While the idea of halftoning was introduced in the 19th century, digital halftoning was first introduced in the 1970s. Digital halftoning uses a binary matrix where each pixel is 1 or 0 (binary representation), representing whether that spot has an ink drop or not. There are many types of digital halftoning algorithms, some of which are optimized for a specific printer model or technology, but they may generally be classified into two major categories [31]: Frequency Modulation (FM) screening and Amplitude Modulation (AM) screening. FM screening changes the density of fixed size dots to achieve the desired gray level, while AM screening changes the size, shape and angle of the dots, but keeps the spacing of dot centers unchanged.

Recently, hybrid methods, combining aspects of both AM and FM methods, were introduced [32] [33], in order to mitigate or avoid the limitations of each approach. Figure 1 demonstrates two examples of FM and AM halftoning performed with 300 DPI, along with an enlargement of each image [31].

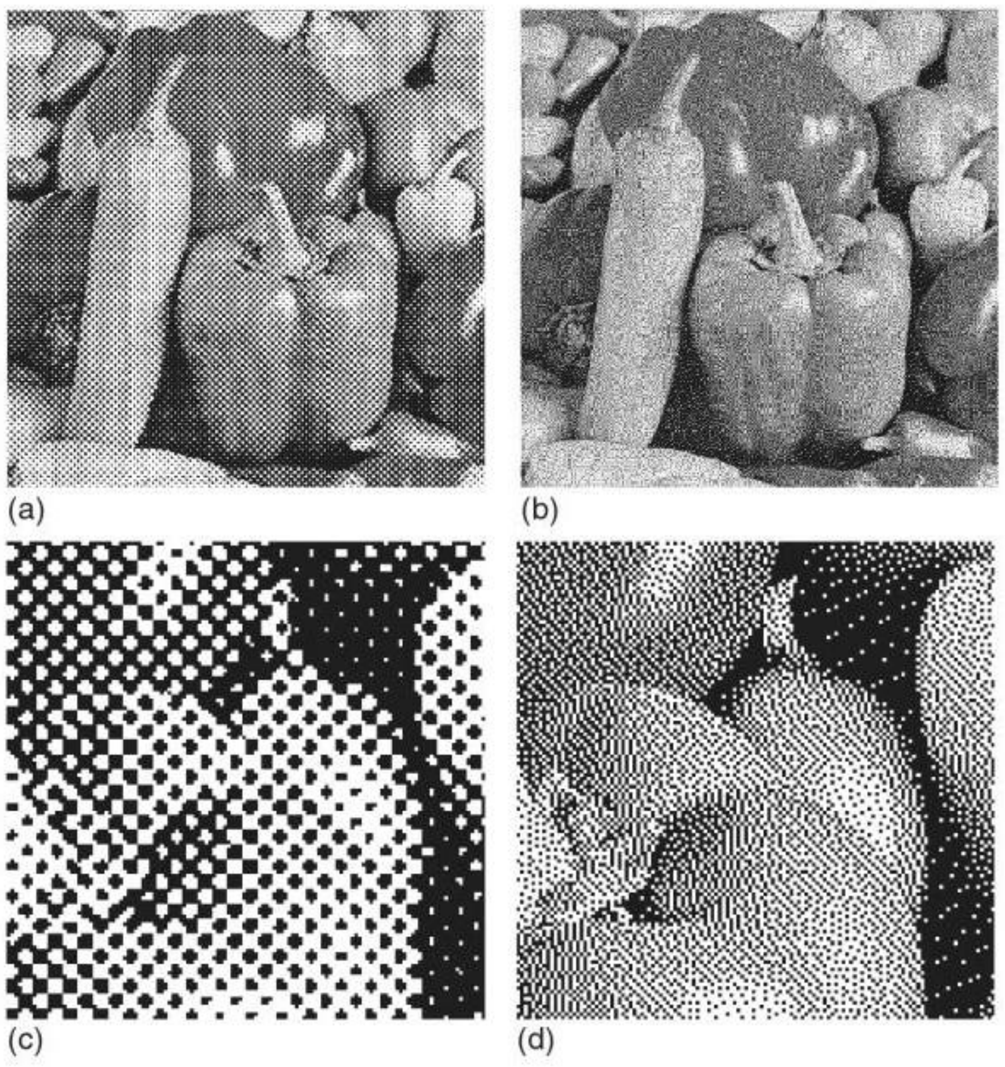


Figure 1: examples of AM and FM halftoning result images. (a) AM halftone. (b) FM halftone. (c), (d) enlargements of a portion of the images shown in (a) and (b), respectively.

Printer Types

In addition to wide academic research, Digital halftoning is under extensive development in the printing industry. There are many types of devices, from desktop inkjet printers to laser and color laser printers and commercial digital presses, which use these algorithms in order to make the appropriate optimizations to gain the best image quality. In this work, we focus on printers were images are simulated by applying dyed substance on a paper or any other substrate.

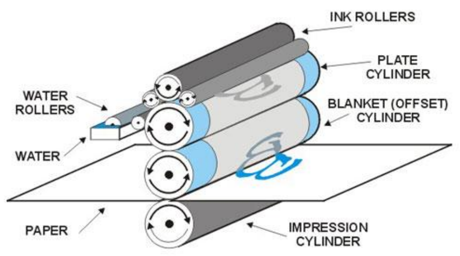
The three most common printers are:

- Offset presses
- Inkjet printers
- Laser printers

Each technology has a set of common or characteristic types of print defects or sources of distortions that may affect the print quality.

Offset Presses

Offset presses use metal plates to press the ink onto the blanket, which then presses the ink onto a substrate. As such, the relative placement of dots within a single color plane tends to be precise. Multiple colors are printed using multiple plates, each plate printed in a separate “pass”. Between color planes, the changes tend to be small global mis-displacement (rotations and translations) of the halftoned images as a whole, rather than varying across the page. Offset presses may have *banding* issues due to small fluctuations in dot size across the page, likely due to variations in pressure of the plates on the substrate during the print process. Offset presses may also have problems transferring (printing) small isolated dots (this is one reason they traditionally used AM rather than FM screens).



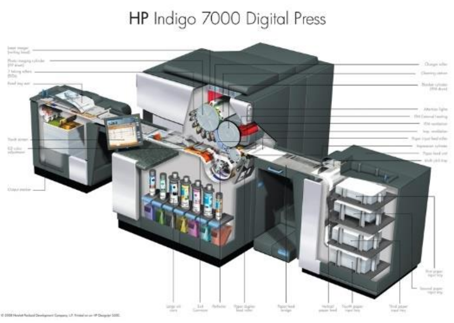


Figure 2a: schematic illustration of offset printer

(from www.offsetprintingtechnology.com)

Figure 2b: Example of digital offset printer

(from www.offsetprintingtechnology.com)

Inkjet Printers

Inkjet printers often have multiple print heads rigidly connected, so a single pass will print dots for few lines and for all colors. This means that the relative placement of dots within a single pass tends to be precise, but due to variations in paper transport speeds, the relative placement of dots along the print direction can vary significantly between different passes. Generally, inkjet print heads can only print a single size dot, but recent generations of write-heads may support two dot sizes.

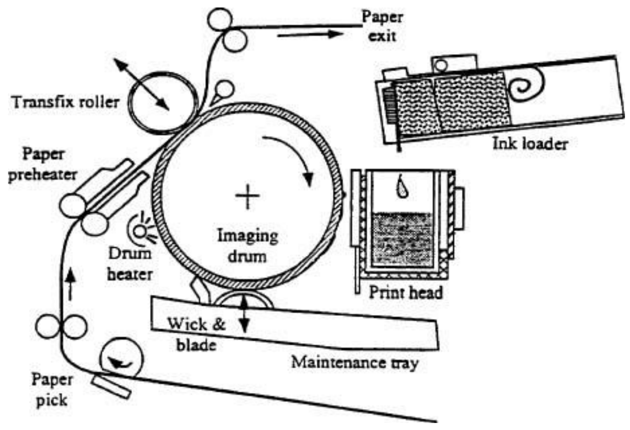


Figure 3a: The basic configuration of the Tektronix's Phaser 350 offset drum transfer ink-jet printer.

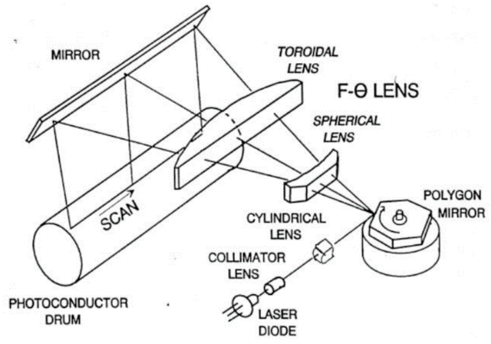
(from www.Imaging.org)

Figure 3b: Example of inkjet printer

(from www.epson.com)

Laser Printers

Laser printers work by charging a photosensitive plate, and then selectively discharging the plate with a laser beam. The beam is scanned across the plate in the cross-print direction using a polygonal mirror rotating at high speed. The plate is typically on a rotating drum, which then collects either toner (dry powder) or ink (liquid) before transferring it to a blanket, which is then brought into contact with the substrate. Color laser printers can only print one color at a time, similar to offset presses, so color plane misalignment due to slight changes in substrate placement between passes is a similar or identical problem to offset presses. In addition, variations in the rotational speed of either the polygonal mirror or the photosensitive plate can result in slight variations in the dot density in the x- and y- directions respectively. In addition, most laser printers have an array of lasers, so a single pass prints multiple rows or lines of dots, somewhat analogous to the inkjet write heads. So, write head banding artifacts can be produced due to changes in dot spacing between bands of dots (each band is a single pass of the write head). One additional complexity with laser printers is that since the laser is digitally controlled, the control electronics may sometimes vary the intensity of the power at a pixel, or vary the width of the pixel. This can be used for much finer control over edge placement, but generally cannot reduce the size of a single isolated pixel. For example, on a diagonal edge the laser wave form may be controlled so that the actual edge of a region is smooth, even if the edge passes through a portion of a pixel (e.g. the middle). Some laser printers may also have problems with small dot transfers similar to offset presses.



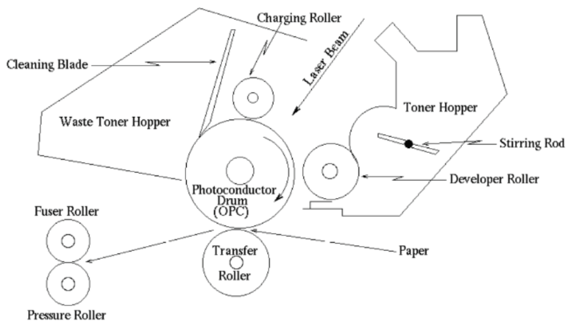


Figure 4a: Laser scan mechanism for electrophotographic printer

Figure 4b: Electrophotographic imaging process

Each halftoning algorithm may be sensitive to an artifact, which is inherent in the printing technology and directly related to the printer's physical limitations. Artifacts like write-head banding, color plane misregistration and dot size variation could lead to poor image quality in the final printed images. The visual severity of the printed image depends on the halftoning algorithm design and the basic assumptions of the designer.

When designing a halftoning algorithm it could be problematic to assume a perfect printer, which has no artifacts. This is the main reason why we need an objective tool, which will allow us to evaluate the algorithm during the development phase and to obtain a detailed information about the sensitivity of the algorithm to any printing technology or printing artifact. Another goal is to allow us to perform comparison between existing algorithms, which might help developing improved halftoning algorithms with reduced visual artifacts.

Types of Distortion

The simulator introduced in this work includes some simple models of common degradations and transformations that occur during the printing process. These are naturally related to the physical process and printing technology. In this work, we focus on some common and important distortions:

1. Dot Size Variation
2. Color plane misregistration
3. Write-head banding

In the following, we elaborate and explain each type of distortion.

Dot size variation

The dot size variation artifact is happening when the printing dots are not equal during the print process. This phenomenon is more common on offset presses and laser printers, while inkjet printers generally have a more constant dot size. Many factors affect the size of the dots, including ink amount variations, voltage variations, laser power variations, dust on the paper, and gluiness of the ink or toner powder. However, the most significant factor is variations in the pressure of the rollers. The direct impact of the dot size variation is mainly banding. Banding is a general term for any type of unwanted straight stripes that appear on images whether they are vertical or horizontal (see figure 7c). More precisely, we model the dot variation as:

$$S_d = S_0 + S_1(P)$$

where S_d is the resulting size of a dot, S_0 is the initial designed size, and S_1 is the variation due to the pressure P of the roller. For the sake of simplicity, in this work we choose to model only the roller pressure phenomenon in order to model dot size variation.

Color plane misregistration

Color plane misregistration occurs when one of the color planes slightly shifts with respect to other color planes (see figure 7b). This artifact is most common on offset presses and color laser printers, since each color is printed in a different pass and slight variation in the page feeding could cause this misregistration. This is unlike inkjet printers where all colors are printed simultaneously.

More precisely: Let I be a halftone binary image with 4 color planes (CMYK) ready for printing.

$$I = (I_c, I_m, I_y, I_k)$$

After the transformation in one of the color planes (e.g. I_c), we will have:

$$I' = (\tau(I_c), I_m, I_y, I_k)$$

where τ is the transformation function for this color plane. This transformation function is a combination of translation and rotation (rigid transformation).

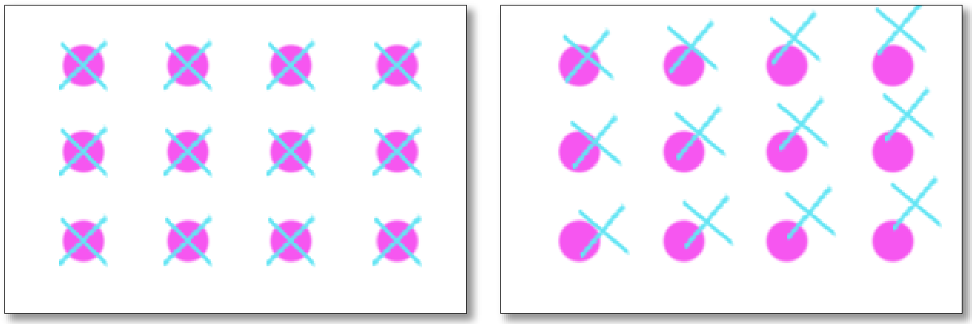


Figure 5: an example of 2-color planes misregistration artifact

Figure 5 shows an illustration of the color plane misregistration for 2 colors: The left figure represents the original image and the correct registration while the right figure represents the same image after a simulation of the color plane misregistration. The circles represent pixels from the Magenta color plane and the X shape represents the pixels from the Cyan color plane. In this example, we introduce both, translation and rotation distortions. The Cyan color plane was translated in both axes and rotated by 5^o.

This illustration demonstrates the rigid transformation of the printing substrate during offset printing. This misregistration may be caused due to a poor control of the substrate placement between color planes.

Write-head Banding

Write-head band or print-head bands are straight lines which occur when there is a slight overlap or a gap between two adjacent passes of the printer's write head (see figure 7a). It happens mostly due to a misalignment of the print cartridges. It is common in inkjet and laser printers.

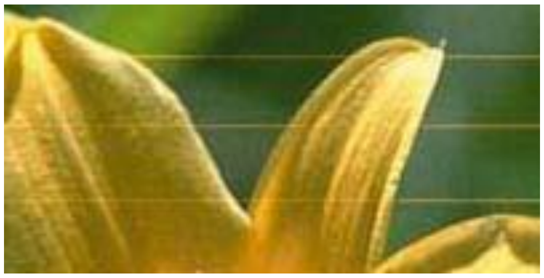


Figure 6: an example of write-head banding artifact

Although the gap distortions is constant along a printed page, in order to test the visible artifacts of this distortion for a wide range of distortion parameters we generate various gap distortions in a single page and tested the resulting print against the ideal print. Let the gap varies between [-k , k] (in unit length along the y-axis), w be the write-head size (in no. of lines) c the fixed y-distance (in length units) between two consecutive lines, and N the maximum number of lines in a printed page.

The gap between consecutive lines i and $i+1$ ($i, i+1$ ∈{0,1,..k}) is generated to be:

$$\Delta y_i = y_i - y_{i-1} = \begin{Bmatrix} c + \varepsilon_i & i \, mod \, w = 0 \\ c & otherwise \end{Bmatrix}$$

where $\epsilon_i = \frac{2ki}{N} - k$

The following images show examples of each print defect:

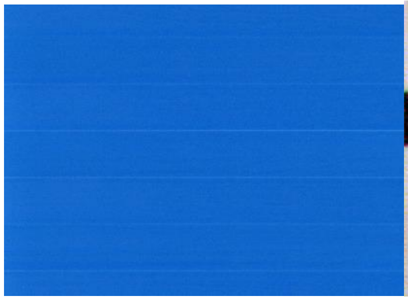


Figure 7a: Write-head banding (from www.hp.com)

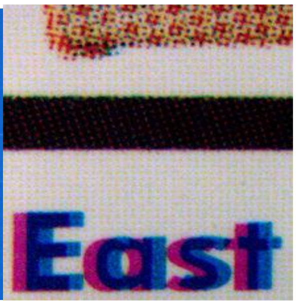


Figure 7b: Color plane misregistration. (from Wikipedia.com/printing registration)

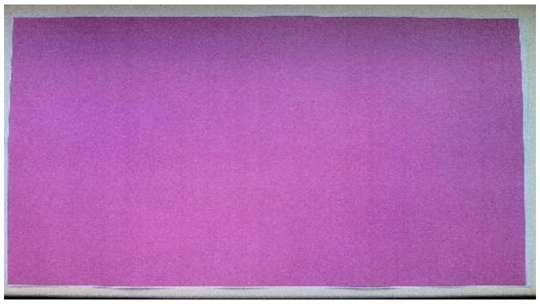


Figure 7c: Dot size variation (from www.colorprintingforum.com)

Finally, the following table summarizes the most common types of printers and their vulnerability to each type of printing defects.

	Offset Press	Inkjet	Laser
Dot size variation	+	-	+
Color plane misregistration	+	-	+
Write-head banding	-	+	+

Figure 8: Summary of printing technologies and its vulnerability to defects

Related work

Designing a halftoning algorithm turns out to be a challenging task. Designing a robust algorithm is even harder and requires a lot of knowledge and experience. Many books, studies and other publications cover the knowledge of the theoretical science behind this craft, explaining the problems and suggesting improvements or methods for analyzing the exact technique.

Baqai et-al [3] describe in their paper the challenges and goals stand in front of the algorithm or halftoning screen designer. This includes the devices model, the Human Visual System, moiré and rosettes artifacts and other topics. They also explain how the types of halftoning screens try to overcome these challenges. In addition, the paper covers some recent trends in this field, such as, iterative approaches and direct binary search (DBS) algorithms. Another throughout review of the color halftoning subject was introduced in [34], which influenced our work.

The design of the screen set used in a halftoning algorithm can be evaluated in order to improve its robustness. For example, in [15], Jin-Young Kim et el. suggested a method for analyzing mathematically the sensitivity of clustered-dot screens to misregistration issues . This approach defines the “cost” and “risk” when designing a screen set. The cost is defined as the impact of any amount of registration error. The risk is defined as the summary assessment of the overall impact of misregistration, taking into account all possible displacements between the two colorant planes.

Some works took into account the specific characteristic of the designated printed model. The first example is the “*Model Based digital halftoning*” [22] by Pappas et-al, which describes a various printer models which are being used by the halftoning algorithms in order to exploit the printer characteristic. It describes the general model, The Sampled Grayscale Printer Model, which is the basis model for other specific models, such as ink-jet and electrophotography model. In addition, this work also describes the usage of these models in screening algorithms, error diffusion algorithms as well as Direct Binary search (DBS). Other works [17, 18, 19] provide additional details and more advanced techniques.

Another similar technique introduced in [35]. This work explains the issues of color printing in electro-photographic printers, such as periodic moiré, and suggests a new technique to address it. In this technique a printer model is utilized which leads to more predictable results. The printer model aims at estimating the absorption of the printed pixel within a 5x5 window. The experimental results show homogenous results with reduced graininess.

In spite of all these great techniques that aim to improve the halftoning algorithms, the goal of our simulation tool is to measure the actual printed results after using a ready-to-use halftoning algorithm while providing benchmarks.

One of the challenges in our work is to evaluate the printed results. In this work, we focused on general similarity, graininess and banding measures. We were inspired by a previous work, which researched both graininess and mottle [15], but finally implemented the ISO 13660 standard [28] as our measurement technique for the sense of simplicity.

The study around similarity, image fidelity and Image quality metrics (IQM) is also fertile. Many studies were performed in order to effectively compare between images and mimic the Human Visual systems (HVS). One of the most famous is the Structural Similarity Measure (SSIM) measure for the image quality assessment [1]. Further research extends this technique to an even more robust one [8] using Multi-scale structural similarity for image quality assessment. While both of these methods focus on grayscale images, other works extended this approaches to color images [6, 7]. The one we choose in this work is the Sparse Feature Fidelity (SFF) [7], which demonstrates impressive results with comparing a reference image with related distorted image.

In banding measurement field, we were inspired by the work of Hila Nachlieli et-al. [2]. This work introduced a tool (MBM) for measuring banding artifacts in printed images, during which, the assessment results were compared to human observation and in average showed same conclusions. The tool uses the second derivative of Gaussian and takes advantages of the multiscale scheme. However, this work is focused on singular mechanical bands and does not accurately fit our needs for banding meter. Finally, we implemented a simplified model for measuring bands, which measures the energy of frequency in a specified range working in the frequency domain as described later in this work.

During our simulations, we use the "*Halftoning toolbox for MATLAB*" [4], which enabled us to use classic algorithms, such as Floyd-steinberg, Stucki and Blue Noise. These algorithms were the first subjects for testing and demonstration. This toolbox directed us to both error diffusion implementations and screening halftoning methods.

Other books, which provided inspiration and knowledge source, are "R. Ulichney: *Digital halftoning"* [26], and "H. Kang : *Digital Color Halftoning"* [27].

The Simulator

The purpose of the simulator is to provide an effective benchmark for the sensitivity of the examined halftoning algorithm to distortions that could happen during the printing process. In fact, the simulator establishes the foundation of many types of simulations that could be conducted to evaluate halftoning algorithms. In this chapter, we will describe how this tool is designed and how it can be used. We used MATLAB to implement the simulator which might be the most convenient way to test halftoning algorithms.

Figure 9 describes the high-level design of the simulator and its steps:

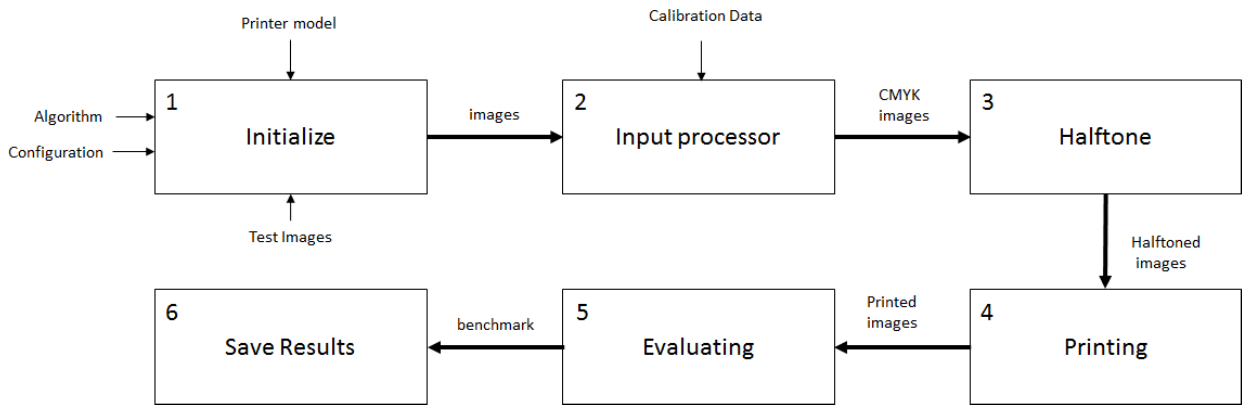


Figure 9: the simulator steps

Step 1: Initialize

In this stage, the user can configure the exact parameters that will be used during the process. It consists the following 5 sub-steps:

1.1 The Algorithms list

First, the process defines the list of halftoning algorithms to examine. An algorithm is defined as an object that holds a pointer to the algorithm function and the type of the algorithm. Currently, there are 3 types of algorithms: Generic, RGB and CMYK. Generic algorithm performs the halftoning function to each color plane in the same way. The number of color planes is extracted in runtime from the given image. RGB algorithm gets an RGB image as an input. CMYK algorithm gets a CMYK image as an input and generally performs different operation to each color plane. For example, some AM algorithms might want to define different angles for each color plane. During the process, each algorithm in the defined list is going through the same pipeline of operations.

1.2 The Configuration

Second, a configuration object is defined to control the exact parameters, which are being used along the process. The following table describes the parameters:

#	Parameter name	Description	Example value
1	dpi	The Dot Per Inch (DPI) which is being used during the printing process	300
2	print_size_inches	The actual print (paper) size (in inches) in both axis.	X=5 , y=5
3	write_head_size	Defines the printer head size (in pixels) which is being used during the printing process	20
4	max_gap	This parameter defines the maximal number of pixels which are being used for gap (or overlap) when simulating write-head banding artifact	4
5	max_dot_size_variation	This parameter defines the maximal increase of the dot size when simulating dot size variation artifact	0.2
6	banding_frequency	This parameter defines the frequency of the bands when simulating the dot-size-variation distortion. It controls the number of bands that will appear in the printed image.	0.2
7	max_shifting	This parameter defines the maximal pixels shifting when simulating color plane misregistration	32
8	icc	This parameter defines the ICC profile to be used during color conversions (see more information in section 2.1).	'bin/SWOP2013C5. icc'

Figure 10: the configuration parameters

More information and the exact usage of these parameters will be explained in details in the following chapters.

1.3 The Printer Model

Third, the printer model is defined. The user can determine whether to use an ideal printer or a printer that includes some artifacts. The ideal printer model performs a simulation of the printing process without any distortion. The other type of printer can simulate the same printing process, but in addition can include some artifacts according to the defined configuration. In other words, this printer can be configured to introduce real-life artifacts during the printing process.

1.4 The Dataset

Another important process within the initialization step is the image dataset setup. The user can choose the dataset of images to be used during the printing process. The images are divided into two categories: natural images and single color images (figure 11),

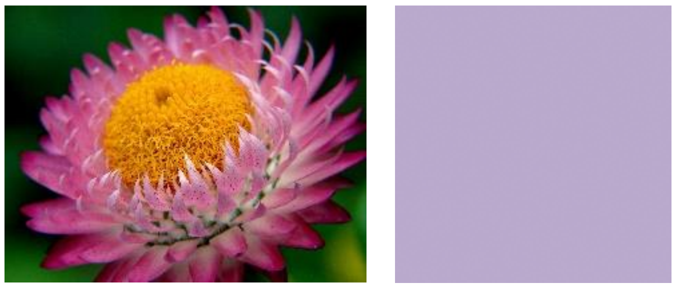


Figure 11: The left image is a natural image and the right image is a single color plain image.

The dataset can be changed easily by replacing the test images in the relevant directories. Both types of images help us evaluate different artifacts.

1.5 The Calibration

Finally, the last step performs the calibration operation.

The Calibration is a process we perform once every first time we use an algorithm. This process allows us to create a dot gain compensation. The dot gain is a well-known phenomenon which causes the printed image to look darker than the original source image (see figure 12). This generally happens during the printing process when the ink interacts with the paper (or other media) and each dot of ink is growing. The effect of larger dots is a darker image.

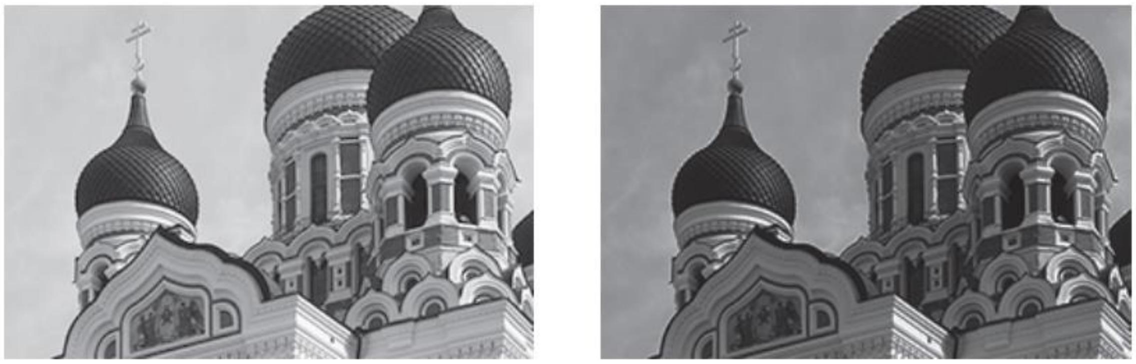


Figure 12: An image (left) and the simulated darker result of the printed image (right) caused by dot gain [34]

We encounter this phenomenon since we print circle dots, while the input is a binary digital image. Each circle dot must cover the whole pixel. Thus, it also covers a portion of its neighbor pixels, and the result is a darker image.

There are many techniques to overcome this unwanted behavior and to compensate in such a way that the printed image will look similar to the original as much as possible in terms of brightness. For this purpose, we create flat grayscale patch images in CMYK format; each image has gray level of 0 to 255. Then, we perform the halftoning algorithm over each patch image. After applying the halftoning, we use a perfect printer in order to produce a printed image. Then, we save the mean level of all pixels in the image. In this case, for each gray level of original image, we have the mean gray level of the resulting image, which is expected to be darker. Figure 13a describes the connection between the original and resulting values. Note that the images are CMYK images, therefore unlike RGB or grayscale image, a higher value means darker result. After we have a vector of all 256 resulting values,

$$f(v_{in}) = v_{out}$$

We calculate the "inverse" vector for these values:

$$v_{in} = f^{-1}(v_{out})$$

While inverting the original vector we preserve a weak monotonic function to make sure the target image will keep the same brightness order between pixels.   This inverted vector will be our calibration data. The calibration data is saved to a file for the next time we use the same algorithm.

Later in the process, we will apply the inverted vector on the source image, which will cause the image to become lighter than the original. Eventually, the dot gain will cause the lighter image to be similar to the original image.

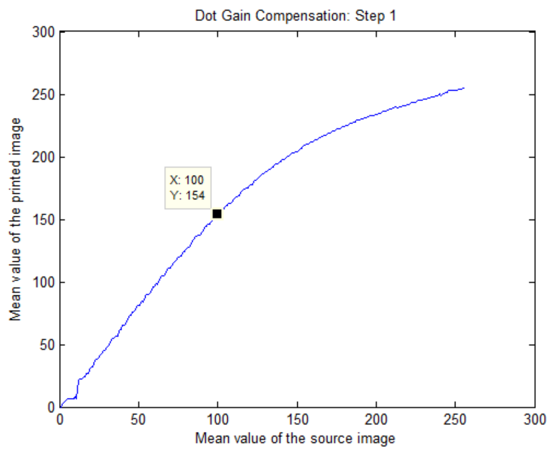


Figure 13a: The plot of the mean values for blue noise algorithm. In this algorithm, when the source image has a graylevel of 100 the darker printed image has a value of 154.

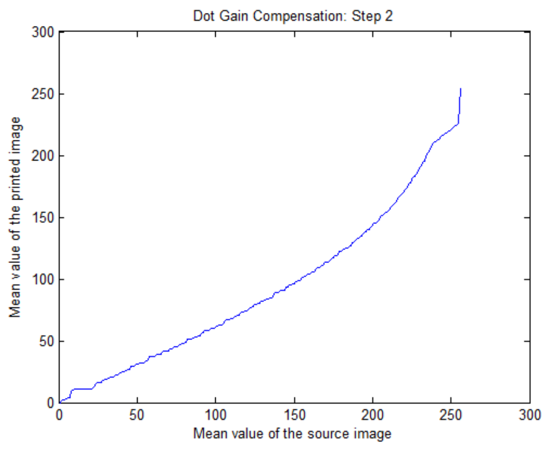


Figure 13b: The plot of the calibrated data after the inversion. This data will be used as the Claibration data.

Step 2: Input processor

Generally, color images are kept in the disk in RGB format; therefore, we need to prepare the loaded images for printing. As a preliminary action, we process the images in order to bring them to the right printing format. Within this process, we perform three operations: Color Conversion, dot gain compensation, resize. In the following, we elaborate each operation we perform:

2.1 Color Conversion

This sub-step is used to convert the RGB image into CMYK image. The conversion is performed using MATLAB's built-in functions and using the input and output ICC profiles. International Color Consortium (ICC) profiles contain information about the color space of a device. Such profiles hold the color characteristics of the device and enable us to map between the input color space and output color space. A profile dataset is generally converted into an intermediate color space (Profile connection space) which in practice mostly uses CIELAB or CIEXYZ color spaces. Therefore, in order to convert from RGB to CMYK we need two ICC profiles: input profile and output profile. The exact profiles can be set in the configuration object as explained in section 1.2.

The reason we have this configuration is to allow any developer to examine his own halftone algorithm, which might be optimized to a specific device. In this case, the ICC profile of the device represents the inherent color attributes of the device. By default, we use SWOP2013C5.icc and sRGB ICC profiles.

2.2 Dot Gain Compensation

Within this sub-step, we apply the tone calibration to the CMYK planes of the image. The Calibration is performed in order to overcome the dot gain phenomenon as described in section 1.5. The tone calibration is applied to the image by converting the integer values of each pixel in the image using a lookup table. The lookup table is the calibration data.

2.3 Resize the image

In this last sub-step, the image is resized to meet the configured print size (in inches) and the printer's Dot-per-inch (DPI). For example, in case the configured print size is 10" by 10" and the printer resolution is 600 DPI, the actual output print size will be:

R = 600 * 10 = 6000

C =600 * 10 = 6000

The output image size is therefore 6000x6000 pixels.

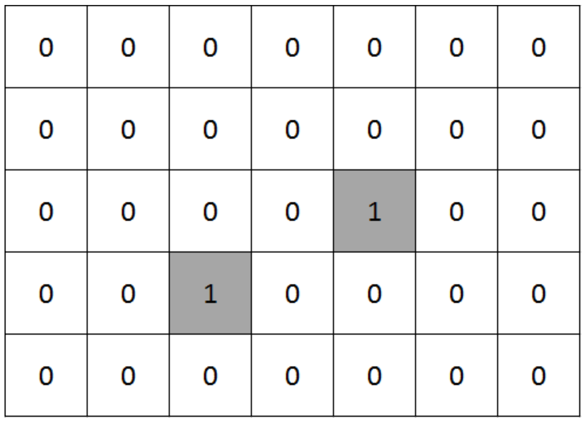
In addition, since the input image size is unknown, we preserve the aspect ratio of the original image.

Step 3: Halftone

After all preparations are made in the previous steps, we send the processed image as an input to the current halftone algorithm under examination. This function receives as inputs the algorithm object and the CMYK image. This step performs the algorithm function according to the algorithm type. For example, if the algorithm expects to get each color plane separately, we iterate over all color planes and perform the halftoning function independently. The output of this stage is a binary image for each CMYK plane, which is the result of the halftoning algorithm.

Step 4: Printing

In this step, we pass the halftoned image to a printer model, which simulates the printing process. In addition to the general printing process, the printer can simulate a real-life printing artifacts such as write-head banding, or color plane misregistration. The purpose of the printing model is to mimic the printer's process according to given parameters. It receives as input the halftoned image, and returns an image, which is considered "printed".



0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	1	0	0
0	0	1	0	0	0	0
0	0	0	0	0	0	0

Figure 14a: The halftoned image

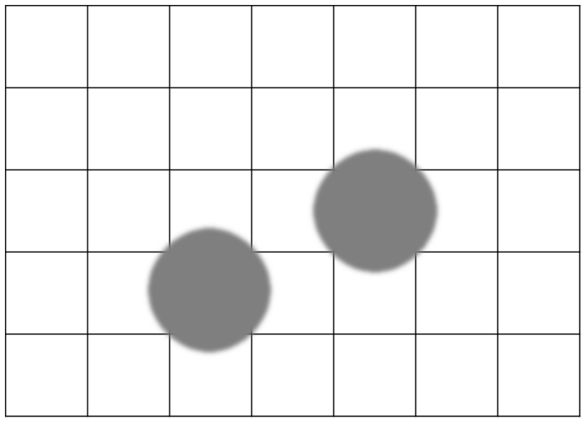
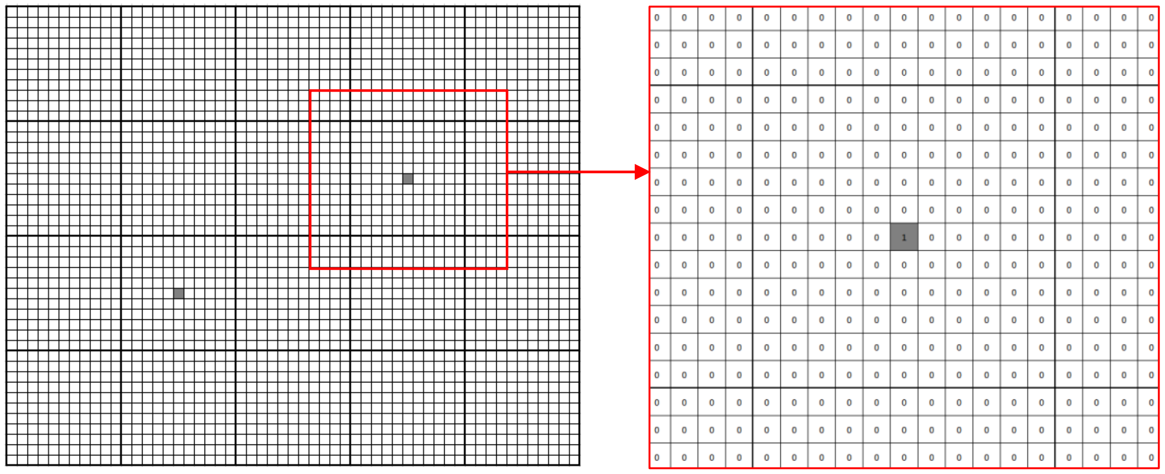


Figure 14b: The printed image

The following sections describe the main operations performed during the printing process:

4.1 Up-sampling

In order to simulate the printing process, we up-sample the halftoned image so we can simulate the ink dot size and manipulate the image in the sub-pixel accuracy. The up-sampling ratio value is a parameter. In all our simulations we used up-sampling level of s=11.



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15:The image from figure 14a after up-sampling

One issue we encountered during the printing process is the image resolution. The input to the printer is typically a 300-1600 DPI halftoned image. As mentioned, this image is up-sampled by 11 in order to simulate printing resolution. For printing size of 10”x10” the image size is up to 176000x176000 pixels per color plane. (1600 * 11 * 10 = 176000)

This enormous image size is impossible to keep in the application memory. This is why all processing is performed in blocks, which are combined together to the final resulting image. The printer model is responsible to maintain this complexity.

4.2 Printing Dots

The halftoned image is a binary image where each pixel has a value of 0 or 1, which states whether an ink dot should be placed at this location. Hence, during the printing process we place a virtual ink drop at every pixel that has value of 1. To simulate the ink drop, we use a disk shape sample with a smooth edge. This sample is used as the convolution kernel to be applied on the halftoned image. Since the halftone image is a binary image, this convolution is equivalent to a convolution with delta kernel. Hence, each pixel neighborhood with a center pixel value of 1 will be replaced by the disk shape. The size of the disk must be correlated with the up-sampling level of the entire image to prevent an oversized dots that might overlap other dots, or undersized dots that will create unwanted gaps.

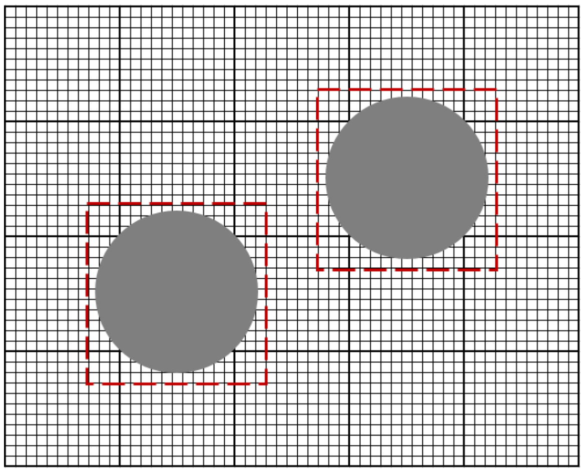


Figure 16: an illustration of two disk shapes convolved with the image from figure 15

While the ink drop has a circle shape, the sample image is square. The exact size of the square is calculated as follows (also described in figure 17). We want an ink drop to be placed at the center of the original pixel. Since the image is up-sampled by 11, an original pixel is represented as an 11x11 pixels in the up-sampled image. In order for an ink-drop to fully cover a pixel square, the ink should be a disk with radius of:

$$R = \lceil 5.5\sqrt{2} \rceil = 8$$

Together with the center pixel and padding, we get a disk size of 19x19 pixels.

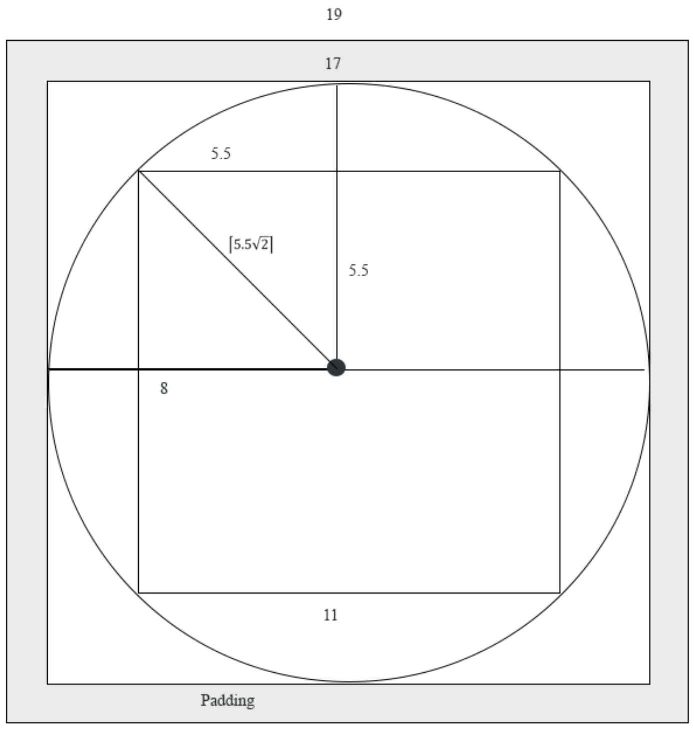


Figure 17: Geometrical calculation of the ink drop.
The square pixel width is 11. Half of it is 5.5. Hence, the radius of the circle is 5.5√2, rounded up to 8.
The diameter including the center pixel is 8+8+1=17. Together with the padding on each side, the outcome is 19.

In order to simulate a real ink dot, the disk's edges needed to be smoothed. For this purpose, we scaled up the disk sample by a factor of 10, and scaled down again by factor of 0.1, both of them use bilinear interpolation. The immediate effect, as can be seen in figure 18, is smooth edges.

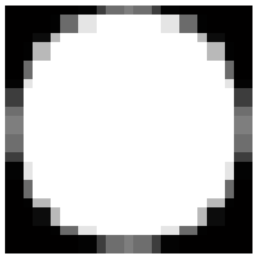


Figure 18a: Creating a disk shape with radius of 8 pixels

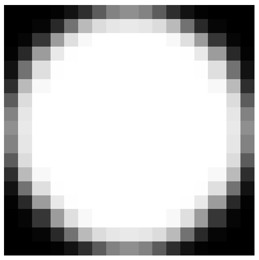


Figure 18b: Smoother disk

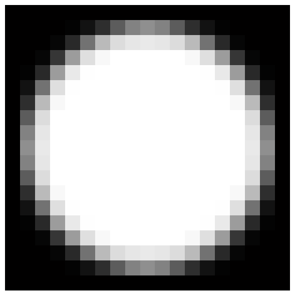


Figure 18c: Smooth disk shape with padding

Finally, we convolve the disk sample with the up-sampled halftoned image which leads to a simulation of ink drops (see figure 19).

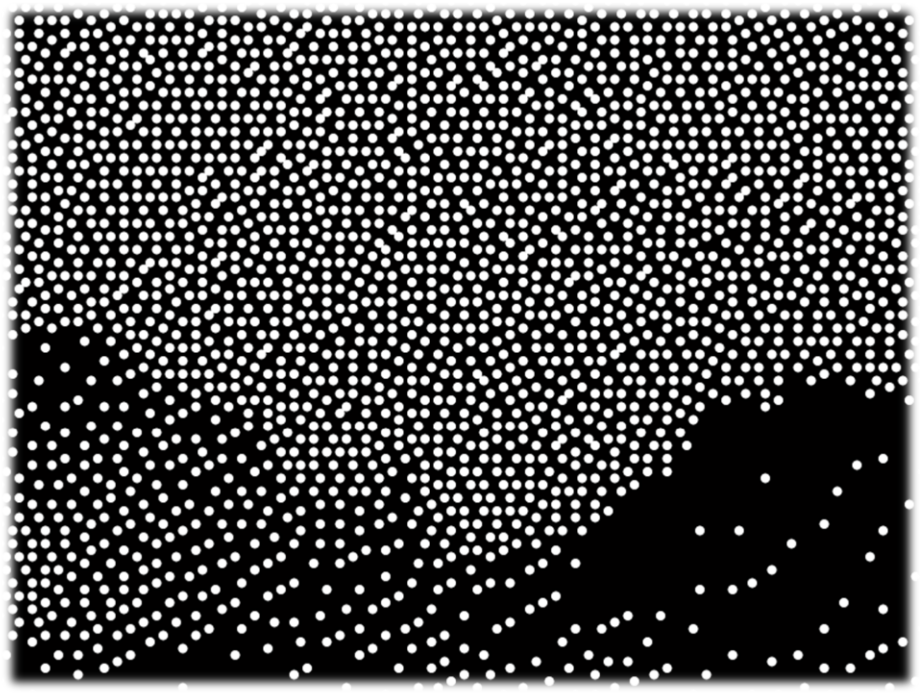


Figure 19: Zoom-in of a printed image

4.3 Adding Printing Distortions

Real life printing process contains variations that sometimes leads to unexpected results, which can mostly be visible as printing defects. In order to demonstrate such behaviors, we add distortions to the printing model that mimics these printing defects. Halftoning algorithms might be sensitive to these distortions. The printer model is getting as a parameter the distortion that needs to be added to the target image. We have implemented the following main distortions:

4.3.1 Dot Size Variation

Dot size variation distortion changes the size of the virtual ink dot, along the page rows. More precisely, it grows and shrinks using a continuous function. We used the sine function to simulate this behavior. The dot size is calculated according to this formula:

$$Size(i) = s_0 + f_0 \sin\left(\frac{2\pi ik}{n}\right)$$

Where i is the row number in the image. s_0 Represents the original size, f_0 the maximum distortion allowed and k is the desired frequency (where n is the total number of rows). The method performs horizontal bands in the image as can be seen in figure 20. Two parameters control this effect, the maximal change of the dot size and the frequency of the bands. Both can be easily configured in every test execution as described in section 1.2. Figure 20 describes four examples of printed images with the same parameters except the banding-frequency parameter, which controls the number of bands appears on the images.

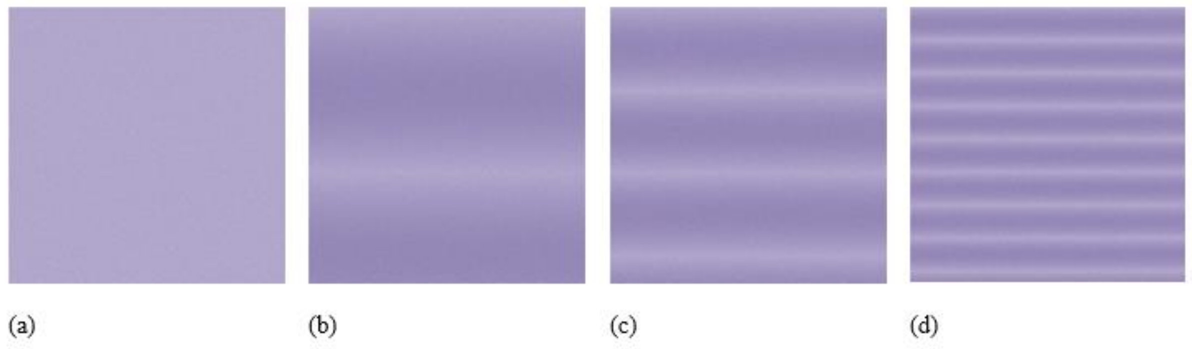


Figure 20: Examples of dot size variation for the Floyd algorithms with the following banding frequencies (left to right): 0, 0.2, 0.4, and 1.0

Figure 21 describes four examples of printed images with the same parameters except for the maximal dot size parameter, which controls the size of each printed dot, and by that affects the magnitude of each band on the image.

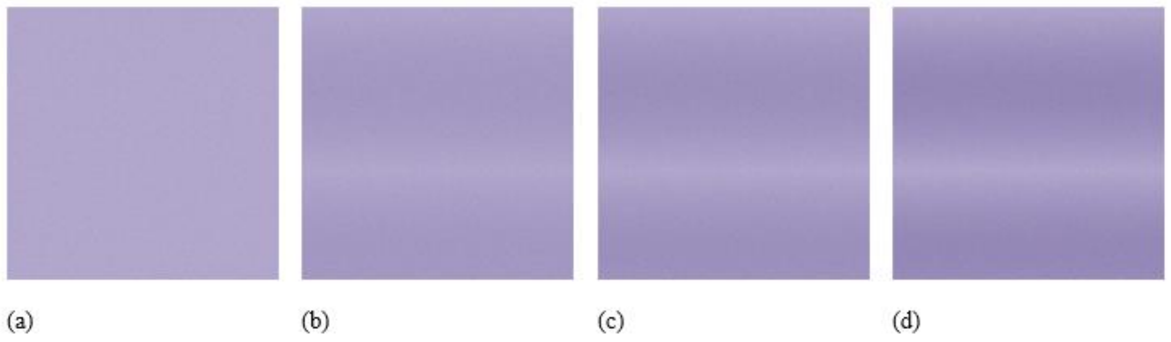


Figure 21: Examples of dot size variation for the Floyd algorithms with the following maximal increase in dot size (left to right): 0, 0.1, 0.15, and 0.2

4.3.2 Color Plane Misregistration

Color plane misregistration distortion shifts one of the color planes at sub-pixel level, and simulates the misalignment between color planes. For this purpose, we shift the first color plane (generally Cyan). In spite of the simple definition of color plane misregistration described in the introduction chapter, we defined the simulation of this artifact by applying multiple color plane misregistration shifts at once. Therefore, we change the shift linearly in every row, yielding a ramp shape of shifting. The parameters of this process is the maximal shifting allowed. The default value is 32 pixels (in simulation resolution). The shift is defined by the following formula:

$$shift(i) = \left(\frac{2m}{n}\right)i - m$$

Where *i* is the row number, *n* is the total number of rows and *m* is the maximal shift.

Figure 22 demonstrates two printed images halftoned by the Stucki halftoning algorithm [36]. The first image does not have any misregistration. The second image has misregistration as described with maximal shift of 50 pixels. Color bands are visible in the second image, especially on the sky part of the image.

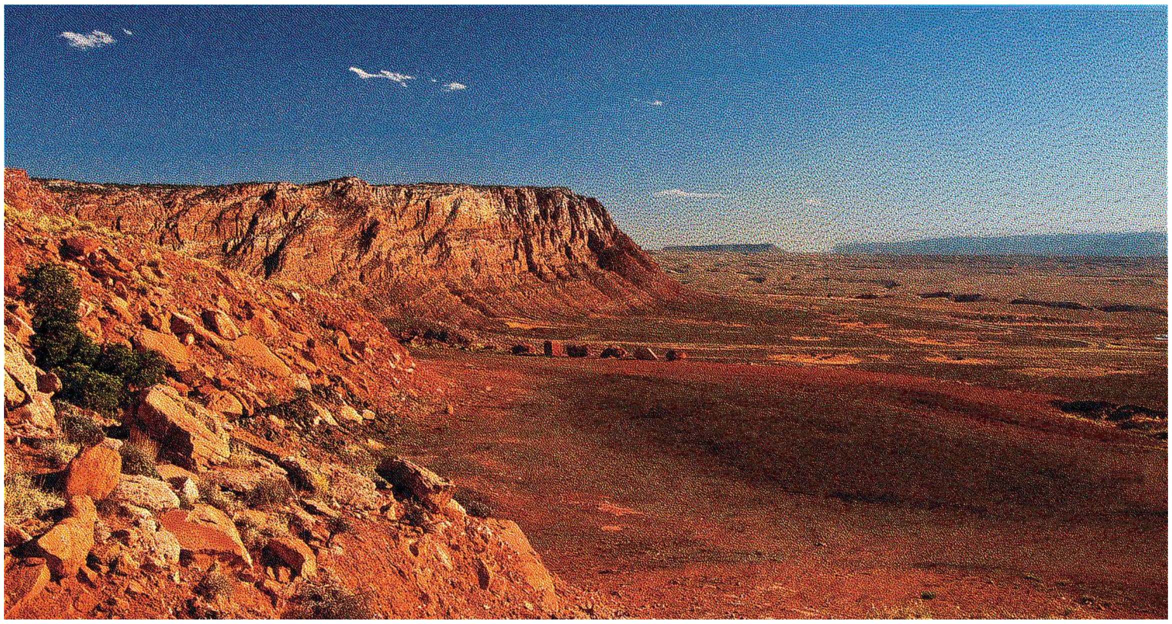


Figure 22a: The reference image halftoned by stucki halftoning, printed without misregistration

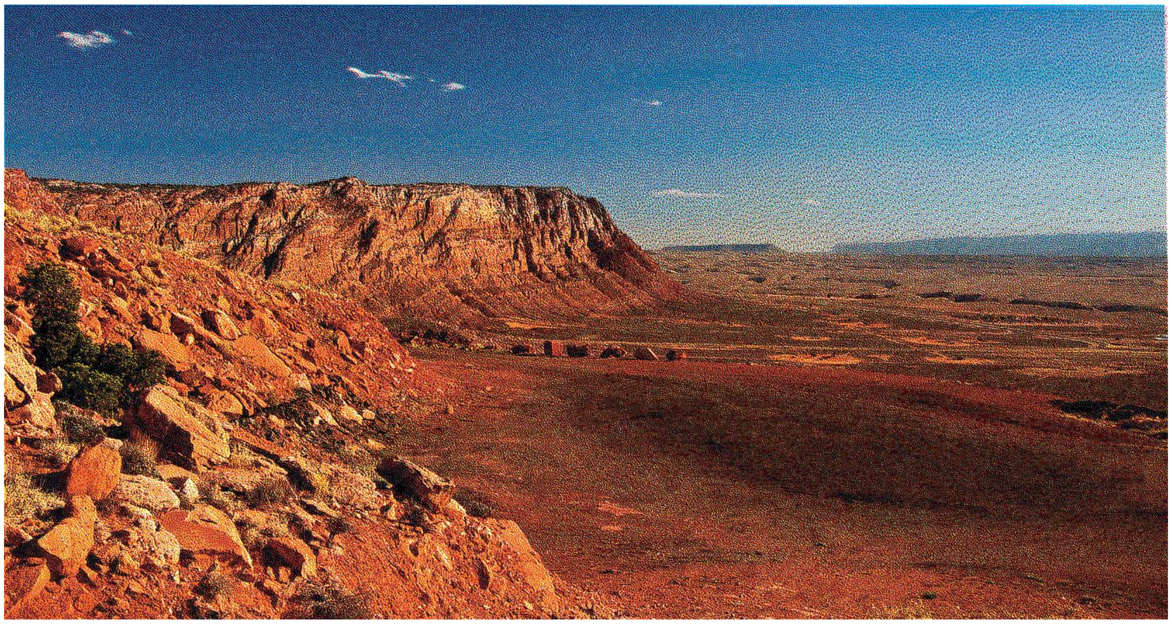


Figure 22b: The reference image halftoned by stucki halftoning and printed with maximal shifting of 50 pixels

4.3.3 Write-head Banding

Write-head banding distortion simulates write-head bands which happens when the write head skips rows of printing or overlaps printed rows due to misalignment during the write-head passes. It will produce thin bands all over the page. The parameters used in this distortion are the write-head size, which is typically 20 pixels and the maximal gap (or overlap) which is typically 4 pixels. The write head banding is defined by the following formula:

$$\Delta y_i = y_i - y_{i-1} = \left\{ \begin{matrix} c + \varepsilon_i & i \, mod \, w = 0 \\ c & otherwise \end{matrix} \right\}$$

Where w is the write-head size in lines, c is the fixed y-distance between two consecutive lines $\epsilon_i = k - \frac{2ki}{n}$ and k is the maximal gap.

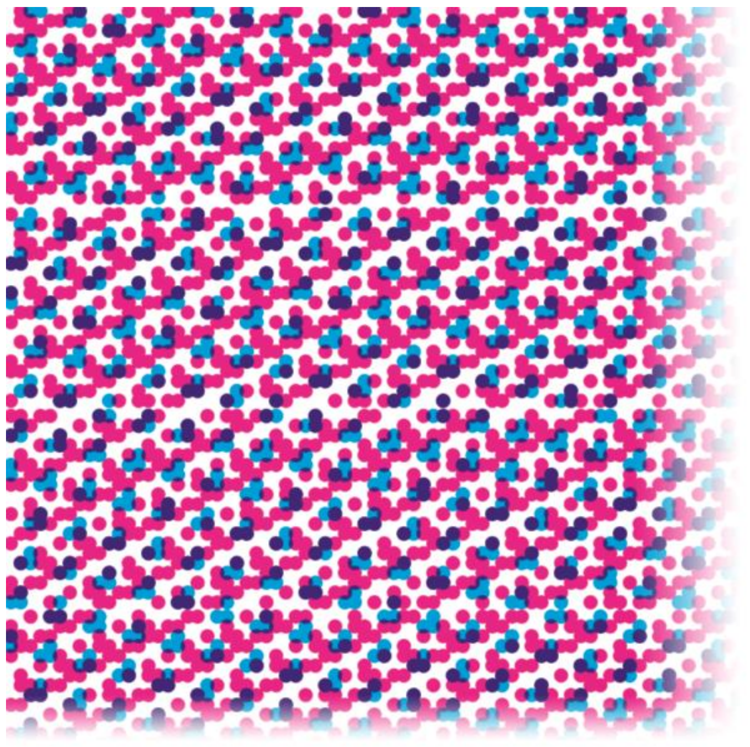


Figure 23: Write head banding on high resolution. Horizontal bands can be seen.

4.4 Downscale

The last operation of the printing process is resizing the image back to the input resolution. The printer model can be treated as a black box that get an input halftoned image, and return the resulting printed image; therefore, it is expected to keep the same image resolution. For that purpose, we resize the up-sampled image back to its original size using a box shaped interpolation kernel with antialiasing.

Step 5: Evaluation Assessment

In order to evaluate the halftone algorithm we need to provide objective measurements correlated with the visual quality of this algorithm in real-life scenarios. Moreover, we would like to assess how the distortions' magnitude affect the visual artifacts. This way we can learn the inherent sensitivity of the algorithm to specific distortions.

The evaluation is performed on the simulated "printed" image as soon as the printing process completes. As mentioned, the image resolution is the same as the input resolution, which is given by the DPI and the desired print size. Out of various measurements, we chose to focus on the following ones:

1. Similarity
2. Graininess
3. Banding benchmark

Similarity

Similarity is a benchmark obtained by measuring how much the source image and the printed image looks alike to a human observer. For that purpose we need to use an Image Quality Metric (IQM) which takes into account the characteristics of the Human Visual System (HVS). In this work we choose the Sparse Feature Fidelity (SFF) function [7] which allows us to easily compare the two images and get a value between 0 and 1. This value represents the benchmark of the similarity between the two images, a reference image and a distorted image. The amount of similarity is given by the SFF function. SFF reflects the chromatic properties of the HVS, and it is found very effective for color image quality assessment. It is based on all of the RGB color components, and it can detect the color distortion in a perceptual way. For that purpose, we convert the printed image from CMYK to RGB before calling SFF the function.

SFF has two main components: “luminance correlation component” called SFF_m and “feature similarity component” of SFF called SFF_f.

The overall score is given by:

$$SFF = \lambda \cdot SFF_m + (1 - \lambda) \cdot SFF_f.$$

Where $0 < \lambda < 1$ is a parameter for adjusting the relative importance of the two components, its default value is 0.8. Figure 24 describes the process of calculation of SFF [7]. We apply this measurement for the natural images only.

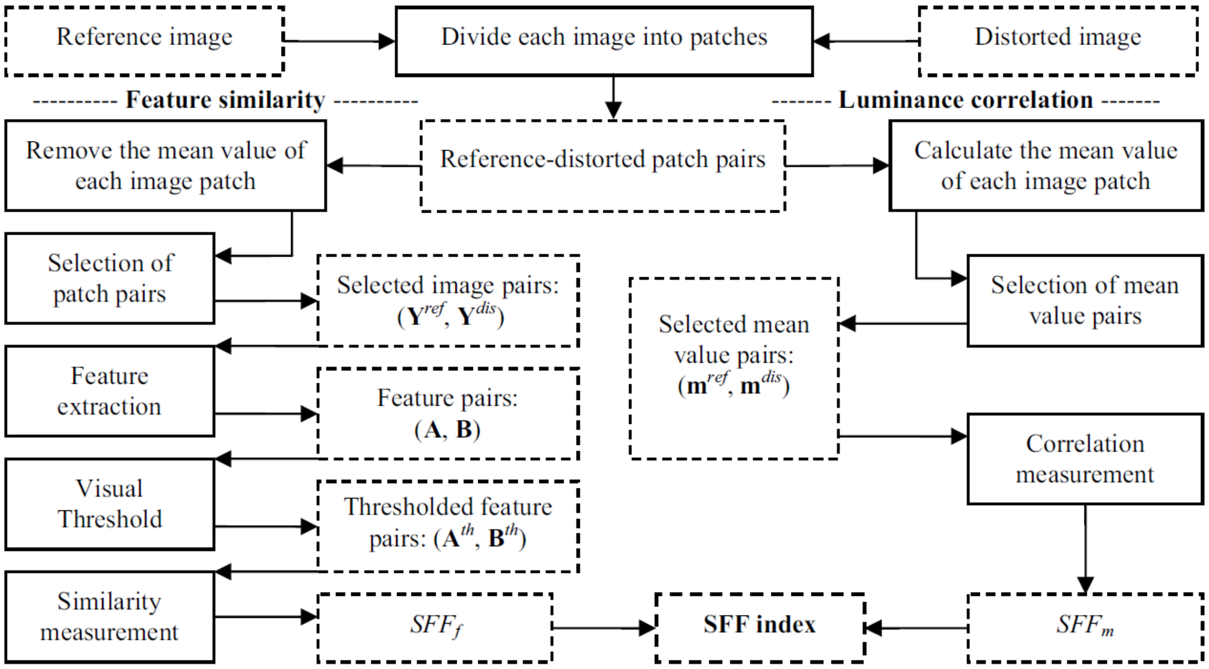

Reference image
Divide each image into patches
Distorted image
---------- Feature similarity ----------
------- Luminance correlation -------
Remove the mean value of each image patch
Reference-distorted patch pairs
Calculate the mean value of each image patch
Selection of patch pairs
Selected image pairs: $(\mathbf{Y}^{ref}, \mathbf{Y}^{dis})$
Selection of mean value pairs
Selected mean value pairs: $(\mathbf{m}^{ref}, \mathbf{m}^{dis})$
Feature extraction
Feature pairs: $(\mathbf{A}, \mathbf{B})$
Correlation measurement
Visual Threshold
Thresholded feature pairs: $(\mathbf{A}^{th}, \mathbf{B}^{th})$
Similarity measurement
SFF_f
SFF index
SFF_m

Figure 24: Block diagram for the fidelity computation of SFF. SFF has two components, SFF_f is the result of feature similarity, and SFF_m is the result of luminance correlation.

Graininess

The Graininess measurement evaluates how much the result image is grainy. Graininess is a subjective perception of a mottled random pattern apparent to a viewer who sees small local density variations in an area of overall uniform density. According to ISO 13660 [28] the graininess is periodic fluctuations of density at a spatial frequency greater than 0.4 cy/mm in all directions. Since the color of the image is irrelevant for graininess measurement, we convert the image to grayscale as a preliminary step of the assessment.

For that measurement, we divided the image into blocks of 1.27x1.27mm, as illustrated in figure 25.

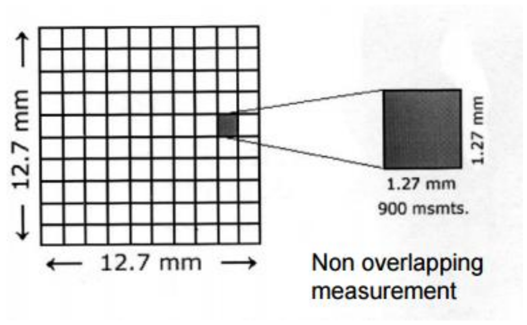


Figure 25: A region of interest divided into tiles [28]

The variance is calculated separately for each block. The graininess is calculated as the mean of the variance :

$$GI = \frac{\sqrt{\sum_{i=1}^{n} \sigma_i^2}}{n}$$

We apply the graininess evaluation only on plain single color test images in order to ensure objective measurement. Natural images were not included in this evaluation.

Banding benchmark

Banding is a phenomenon of straight lines, thick or thin, which appear on the printed image vertically or horizontally. The banding benchmark measures the magnitude of the banding artifact across the image. This reflects how the banding artifact is noticeable to the human observer.

$$I_{band}(x,y) = I_{orig}(x,y) + f_0 \sin(\frac{2\pi ky}{N} + \phi)$$

The banding meter estimates the amount of banding artifacts using analysis in the frequency domain. We define a range of frequencies the banding may appear at and use two types of scores: The first score measures the difference between the maximum and the minimum of the power spectrum evaluated over the frequency range. The second score accumulates the power spectrum defined over the frequency range. Simulations show that the two scores provide similar qualitative results.

Again, we apply the banding benchmark evaluation only on plain single color test images in order to ensure objective measurement. Natural images were not included in this evaluation.

Results

Our main examination is used to test the sensitivity of halftoning algorithms to specific distortions. For this purpose, we used the Simulator with many different configured values and checked at which level the sensitivity is detectable. These tests also demonstrate the usage of this tool and reveal its powerful capabilities.

Testing Algorithms

Most of the algorithms we used in our tests were taken from the "Halftoning toolbox for MATLAB" [4]. With this framework, we were able to test classic color halftoning methods such as Floyd, Jarvis, Blue Noise and un-clustered dithering screens [24, 29, 37]. In order to extend our list of halftoning algorithms, we implement a new simple AM halftone algorithm that uses a different angle per color plane. It was important to evaluate the results involving such an algorithm that performs the halftoning process to each color plane independently, unlike all other algorithms from the toolbox.

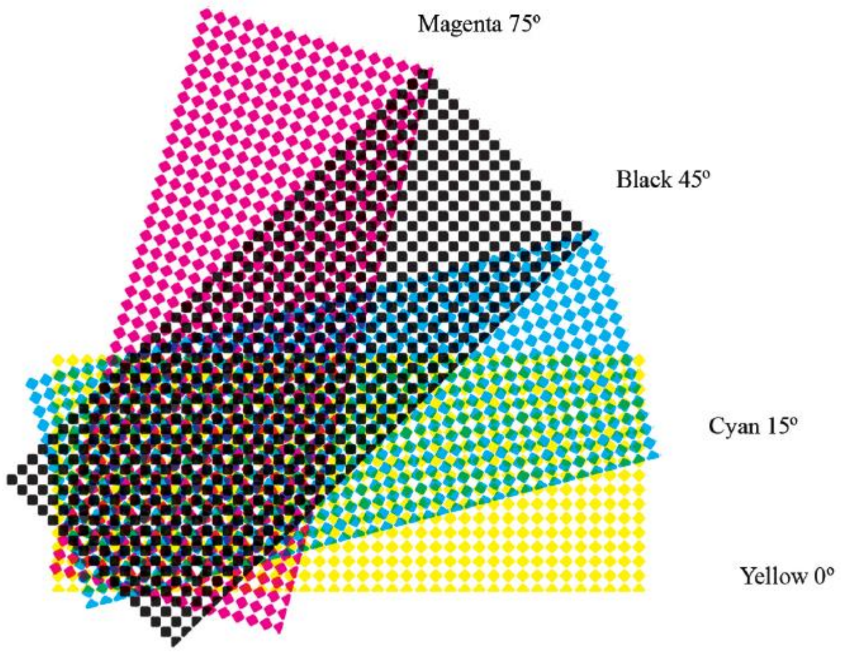


Figure 26: AM color halftoning angles and rosette pattern [34].

The simple AM screen algorithm we developed uses threshold screens. These screens are created at the same size of the source image. A pixel in the source image whose value is bigger than the value of the associated pixel (same coordinates) in the screen has a value of one, otherwise zero. The screens are created using a set of cells, one for each color plane. Each cell defines the angle of its screen. In our case: Cyan 76º, Magenta 14º, Yellow 0º, Black 45º [27].

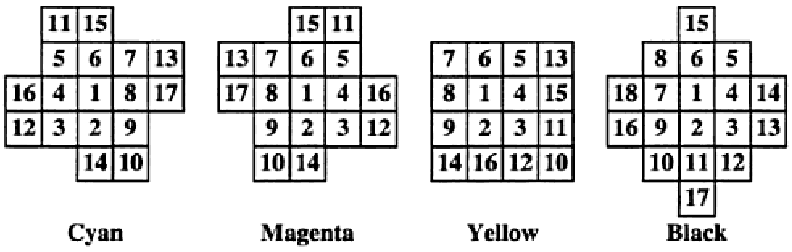


Figure 27: The chosen tiles in our sample AM screening algorithm [27]

As expected, our AM screen algorithm often generates the rosette patterns as can be seen in figure 28.

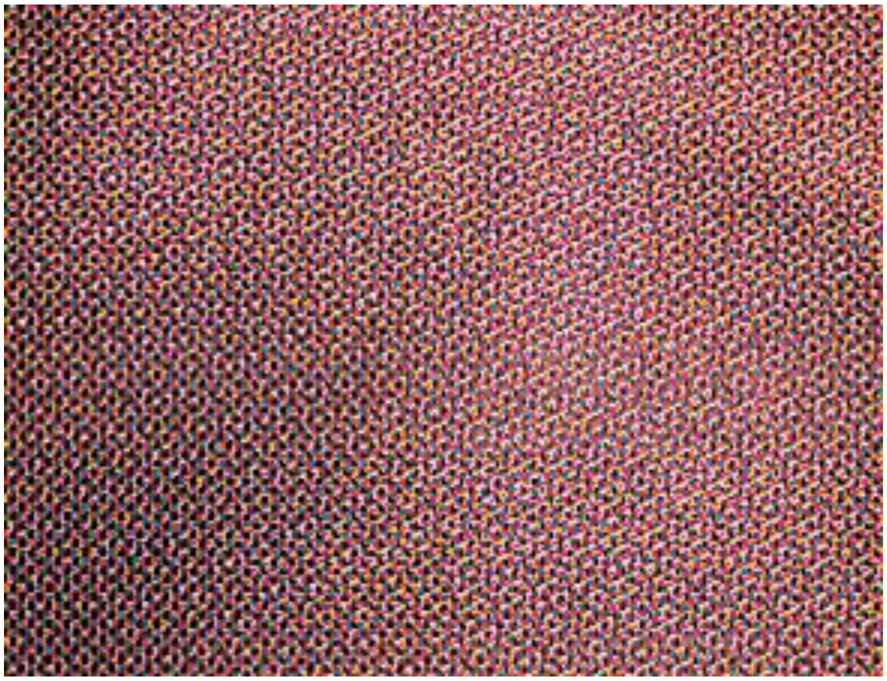


Figure 28: A section of result image printing using our sample AM algorithm.

We used the following halftoning algorithms in our tests:

1. Floyd-Steinberg error diffusion [24] (floyd)

2. Jarvis error diffusion [37] (jarvis)

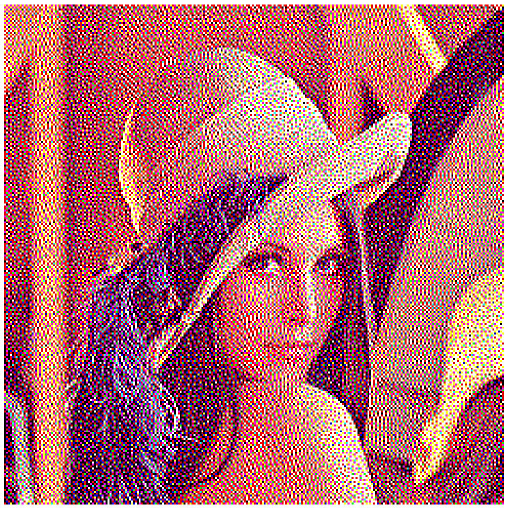
3. Blue Noise [29] (bnoise)

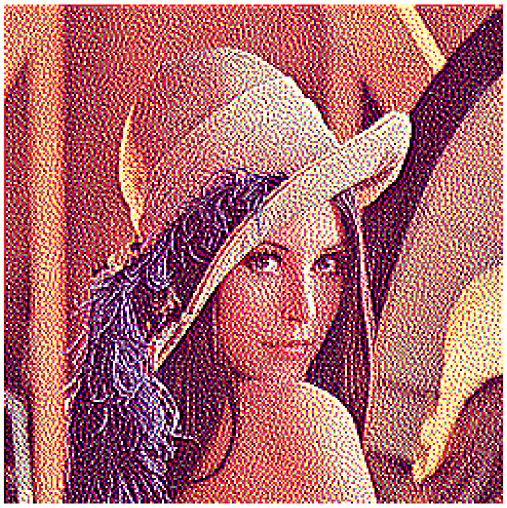
4. Classical screen dithering algorithm, unclustered dots with 16 gray levels (Screen_16u).

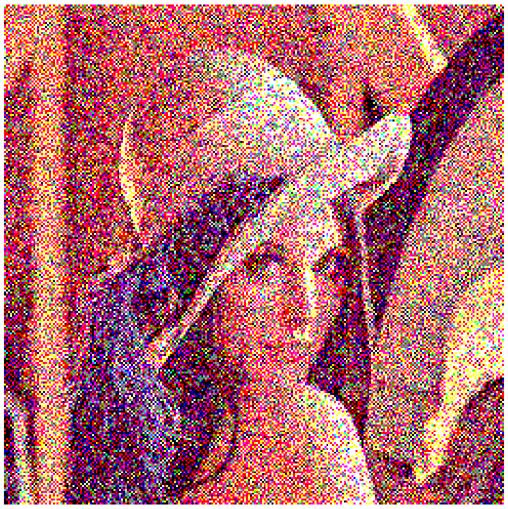
5. Stucki error diffusion [36] (stucki)

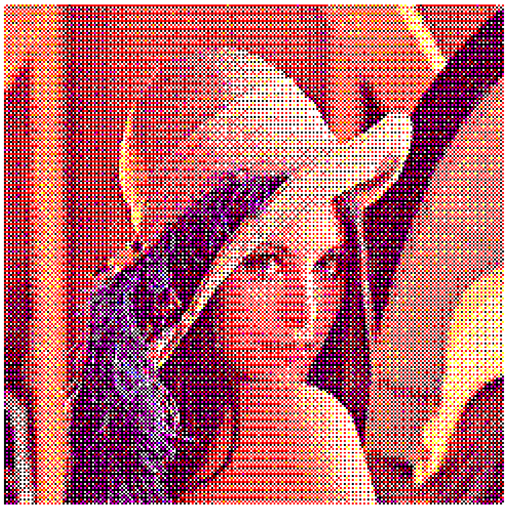
6. Our custom-made halftone algorithm (AM Screen).

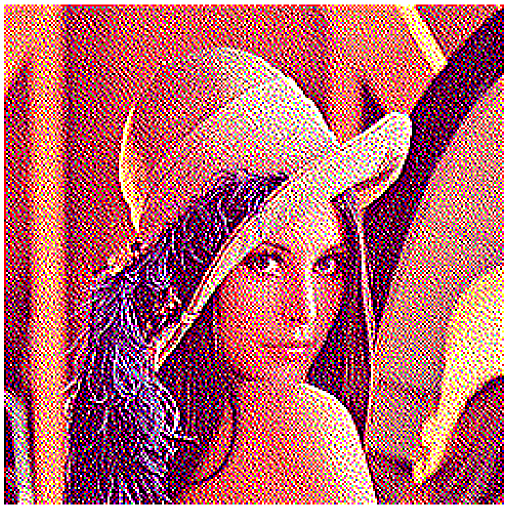
Figure 29 demonstrates sample images produced by each of the algorithms.


Floyd


Jarvis


Blue Noise


Screen 16u


Stucki

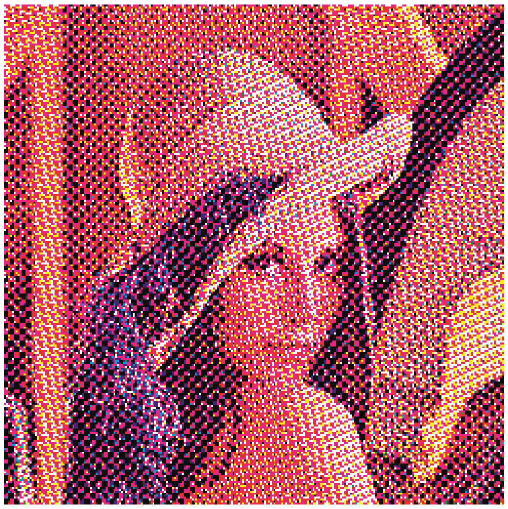
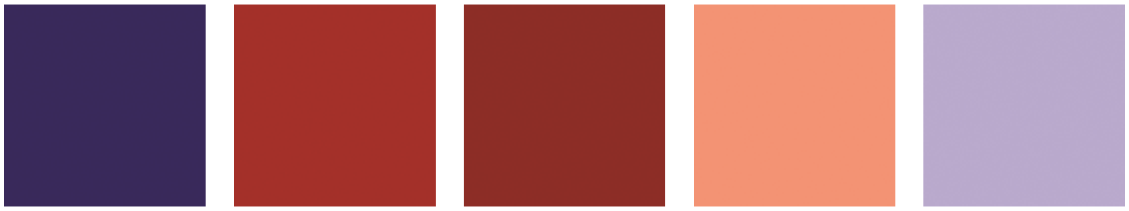

AM Screen

Figure 29: Halftoning algorithm examples

The Dataset

The Simulator allows the user to define the dataset of images that will be used in the simulation by locating them in the Images directory. For the purpose of these tests, we defined a set of images for both groups, natural images in RGB format and plain single color images in CMYK format (See figures 30 and 31). While natural images were used in similarity evaluations, the plain images are used to evaluate graininess and banding.

The following figures show the exact images used in our tests:



C=255
M=250
Y=35
K=44
	C=51
	M=255
	Y=255
	K=35
		C=63
		M=255
		Y=255
		K=64
			C=0
			M=126
			Y=128
			K=0
				C=77
				M=75
				Y=0
				K=0

Figure 30: The single color plain images used in our tests

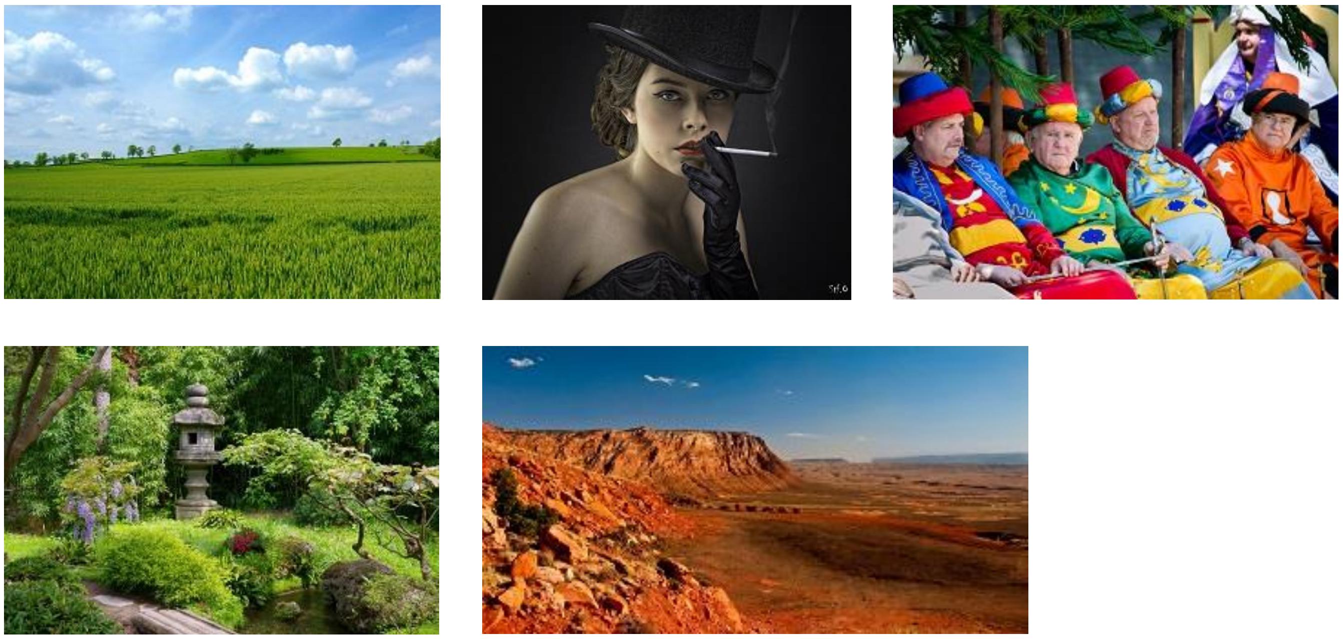


Figure 31: These are the natural images used in our tests. All images taken from Flickr [36]

The Configuration

In all our tests, we used a default set of configuration values, and updated the specific parameter under test. The following table describes the default values:

Parameter	Default Value
dpi	300 DPI
print_size_inches	5”x5”
write_head_size	20 pixels
max_gap	4 pixels
max_dot_size_variation	0.2
banding_frequency	0.2
max_shifting	32 pixels
icc	'bin/SWOP2013C5.icc

Figure 32: The default configuration

Dot size variation

The purpose of this test group is to examine the impact of the dot size variation on each of the algorithms with respect to each one of the evaluation measurements.

Similarity Test

In this test, we examine the influence of the dot-size-variation on the SFF (similarity) measure. We run the simulator on the entire algorithms set, using the images dataset, with various values for the *max_dot_size_variation* parameter in the range of 0 to 0.2 (step of 0.05). We evaluated the result images using our similarity evaluation function (SFF). Eventually, we calculated the average of all results among the natural images. The results are described in the following graph:

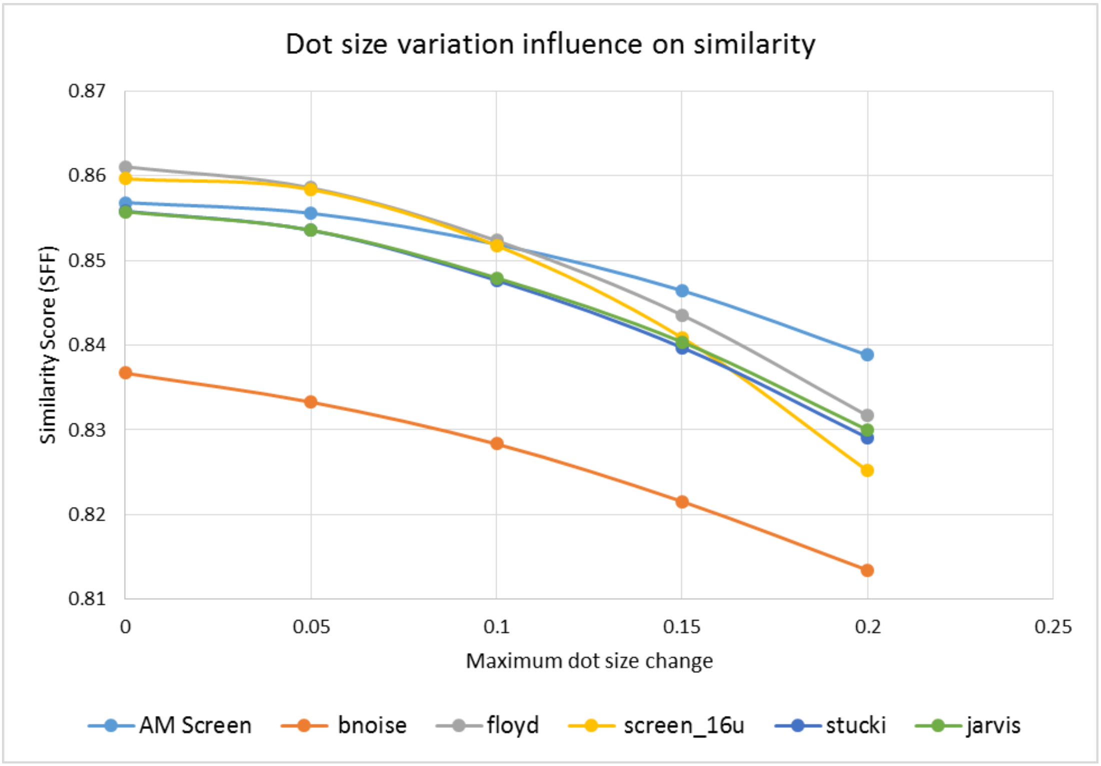


Figure 33: The influence of the dot size on the similarity (SFF) score

Dot size maximal increase	AM Screen	bnoise	floyd	jarvis	screen_16u	stucki
0	0.85683	0.836706	0.861068	0.855762	0.859642	0.85583
0.05	0.85558	0.833292	0.858572	0.853582	0.858386	0.85357
0.1	0.851922	0.828342	0.852354	0.847948	0.851772	0.84766
0.15	0.84645	0.821554	0.843556	0.840376	0.840884	0.839724
0.2	0.838816	0.81344	0.831658	0.829986	0.82516	0.829062

Figure 34: The mean values of the similarity score (SFF) in dot-size-variation test

Dot size maximal increase	AM Screen	bnoise	floyd	jarvis	screen_16u	stucki
0	0.0242087	0.032121	0.022285	0.024985	0.023916724	0.025324058
0.05	0.0244581	0.033518	0.023074	0.025683	0.024005886	0.026067019
0.1	0.0254269	0.034714	0.024608	0.027181	0.025432408	0.027558622
0.15	0.0268251	0.036397	0.026422	0.02872	0.028331135	0.029123105
0.2	0.0289982	0.039001	0.029037	0.031093	0.033405758	0.031505683

Figure 35: The standard deviation values of the similarity score (SFF) in dot-size-variation test

Conclusions from the graph above: The similarity function returns a different result for each algorithm where no variations are introduced (0 in x-axis). This gives us information about the quality of each algorithm without any distortion (perfect printer). For example, Blue noise (Orange dots) has much less similarity than other plots. These results are also consistent with the actual results as can be seen in figure 36. In this figure, we took a block from the source image, the printed result after using the Blue Noise algorithm and the same block from the Floyd algorithm. It is plain to see that the Floyd yields a better perceptual result than the Blue Noise, which is consistent with the graph results above.

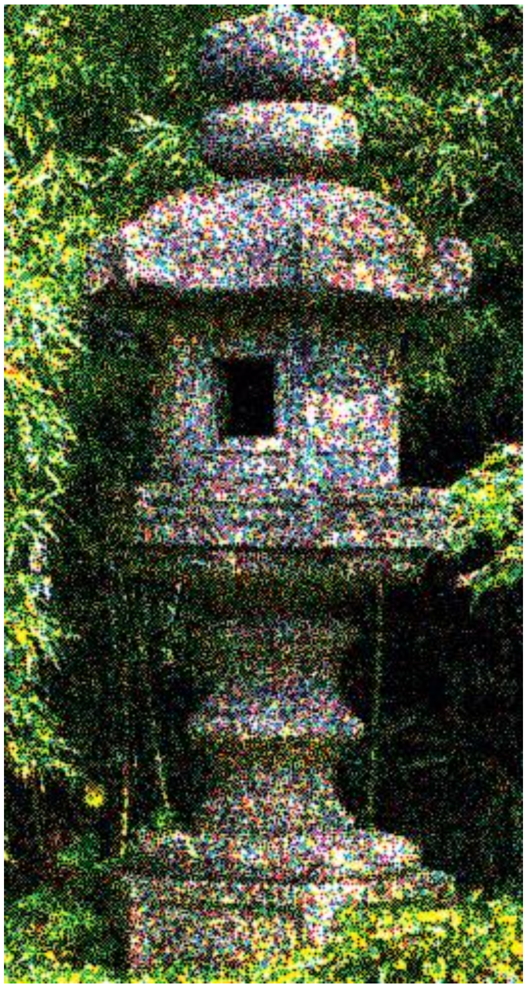


Figure 36a: A sample of result image created using the Blue Noise algorithm

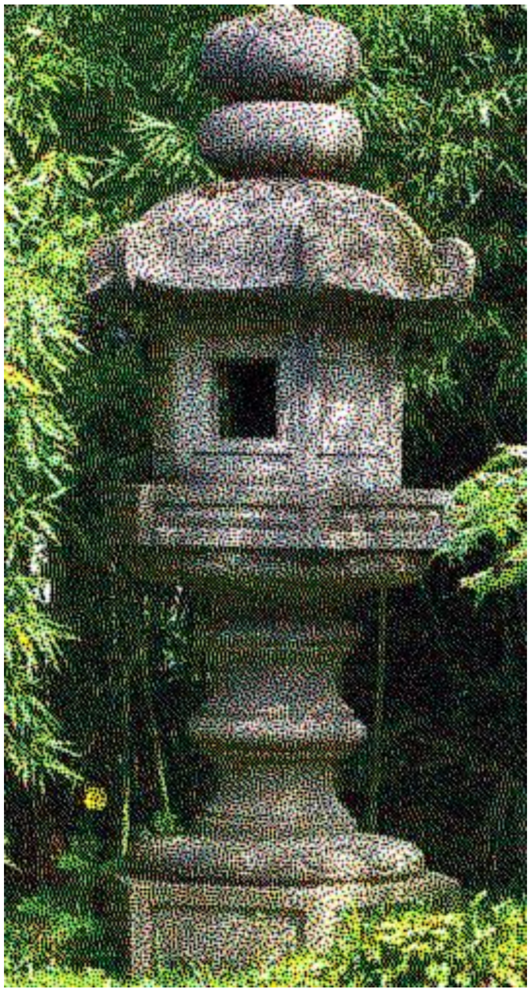


Figure 36b: A sample of result image created using the Floyd algorithm

The rest of the graph shows the trend of each algorithm when raising the maximal dot size. It can be seen that “screen 16 un-clustered” halftoning degrades rapidly when increasing the dot size variations towards 0.2. Another observation is that our sample AM screen algorithm is less sensitive than the other algorithms. Finally, one can locate the exact place where one algorithm is crossing another algorithm, which means the exact value when the inferior algorithm becomes superior.

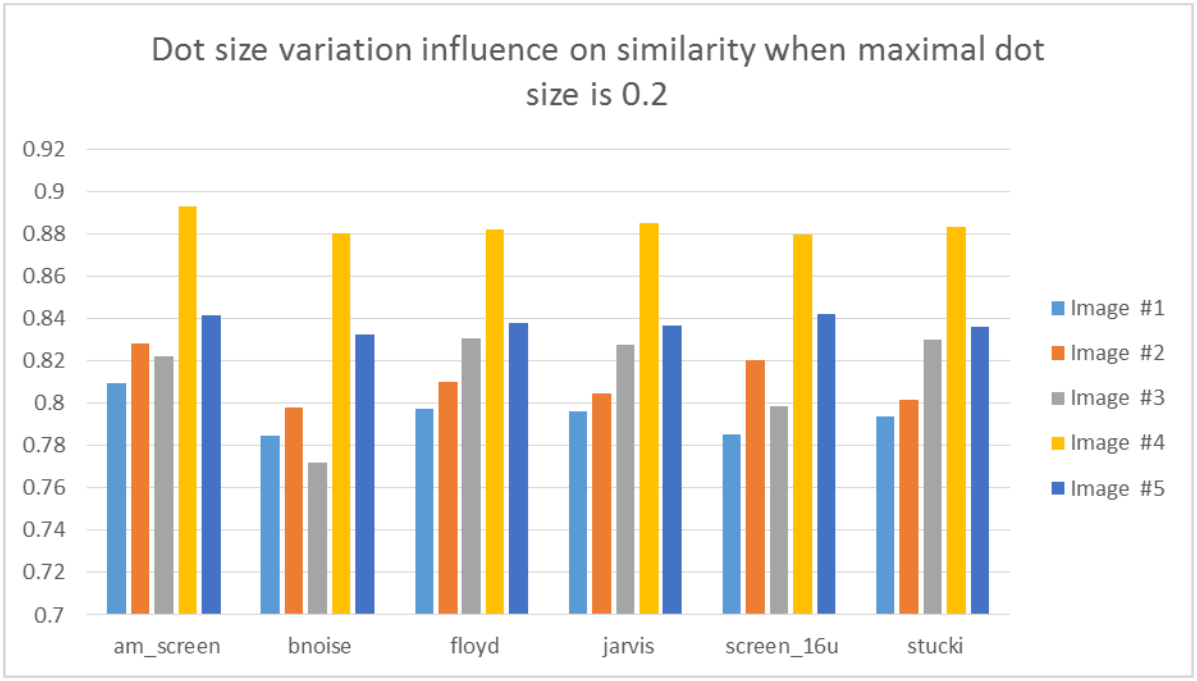


Figure 37: The influence of the dot size on the similarity score in each test. The maximal increase in dot size is 0.2

Graininess Test

This test demonstrates the influence of the dot size variation on the graininess for each algorithm. For that purpose, we repeated the same configuration and settings as in the previous test, except for the evaluation function. In this test, we used the graininess benchmark function. Surprisingly, we did not encounter any exciting effect. However, we can see a slight and linear degradation in all tested algorithms. The following graph describes the results:

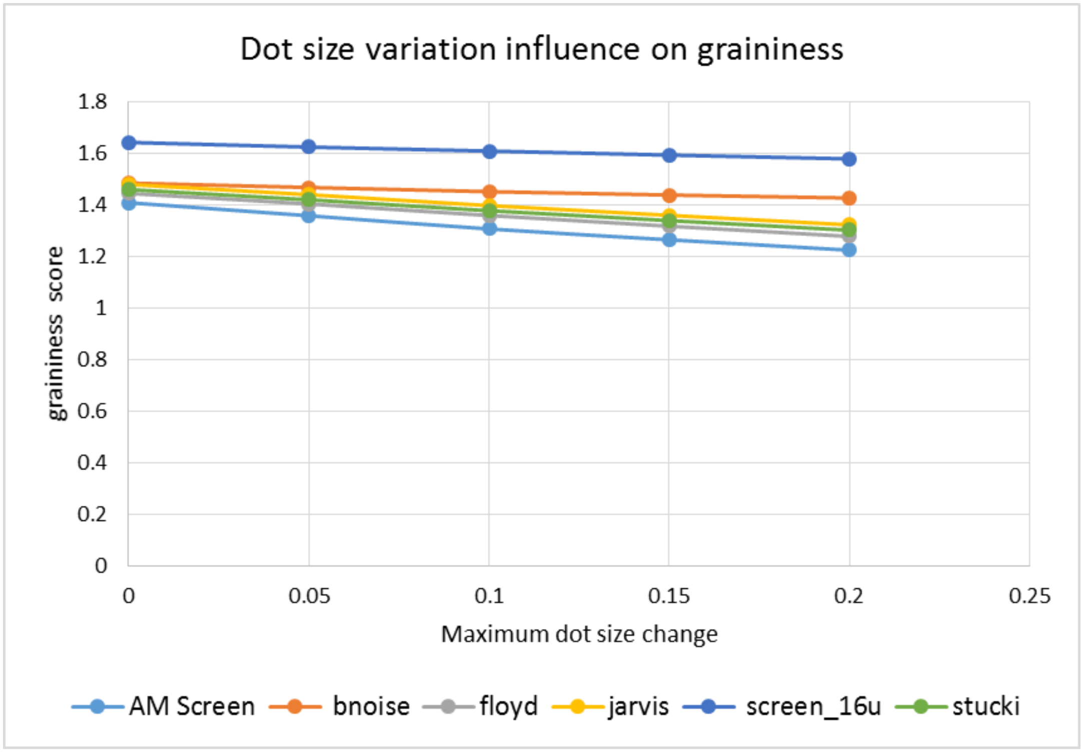


Figure 38: The influence of the dot size on the graininess score

Dot size maximal increase	AM Screen	bnoise	floyd	jarvis	screen_16u	stucki
0	1.408816	1.484862	1.44483	1.47976	1.64266	1.459718
0.05	1.358122	1.467936	1.403896	1.439874	1.624794	1.42021
0.1	1.306912	1.45171	1.359018	1.39773	1.60909	1.377746
0.15	1.264874	1.43839	1.31751	1.359752	1.593714	1.33915
0.2	1.22437	1.426812	1.277974	1.322768	1.578784	1.302082

Figure 39: The mean values of the graininess score in dot-size-variation test

Dot size maximal increase	AM Screen	bnoise	floyd	jarvis	screen_16u	stucki
0	0.5603425	0.543688	0.454166	0.483216	0.643712916	0.459397978
0.05	0.5224205	0.521589	0.421743	0.456998	0.634652474	0.432153142
0.1	0.4838255	0.500204	0.39415	0.431301	0.627621442	0.408006082
0.15	0.448484	0.479432	0.367778	0.407192	0.617178825	0.384993748
0.2	0.4122762	0.459574	0.341638	0.383972	0.605149021	0.362408018

Figure 40: The standard deviation values of the graininess score in dot-size-variation test

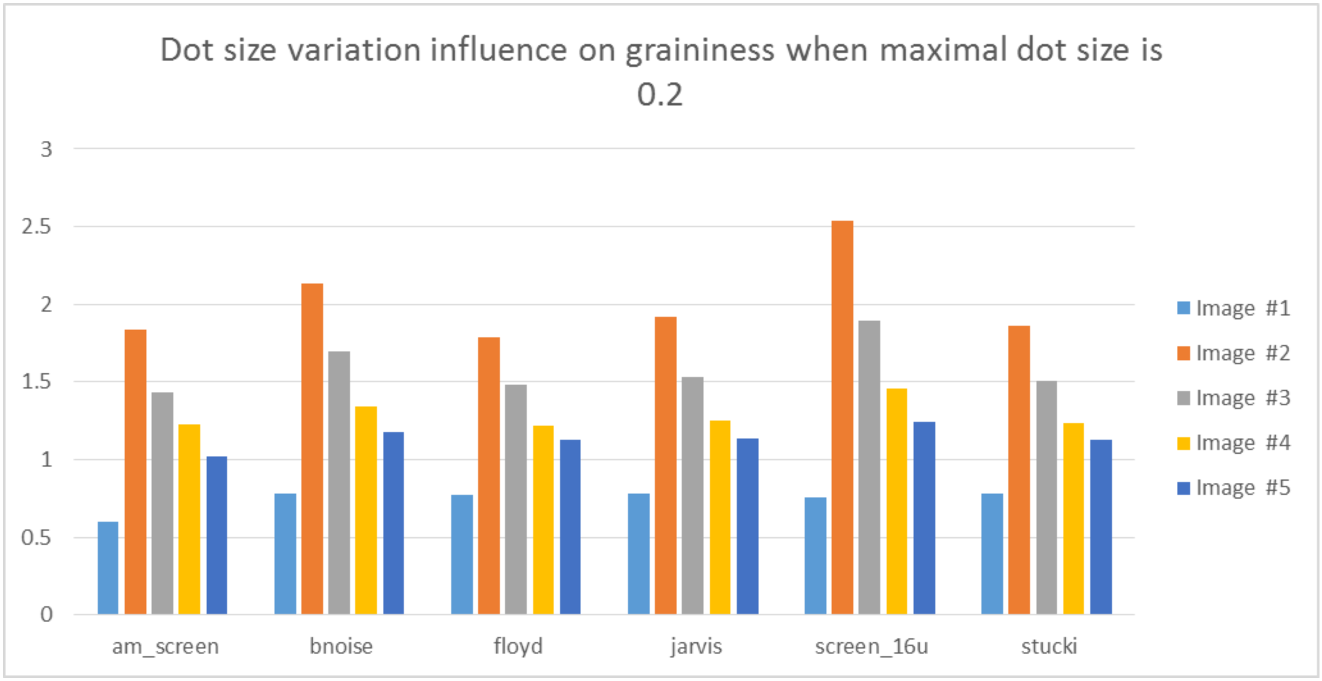


Figure 41: The influence of the dot size on the graininess score in each test. The maximal increase in dot size is 0.2

Banding Test

The purpose of this test is to demonstrate the influence of the dot size variation on banding score. In this test, the evaluation was performed using the banding meter. The results are presented in the following graph:

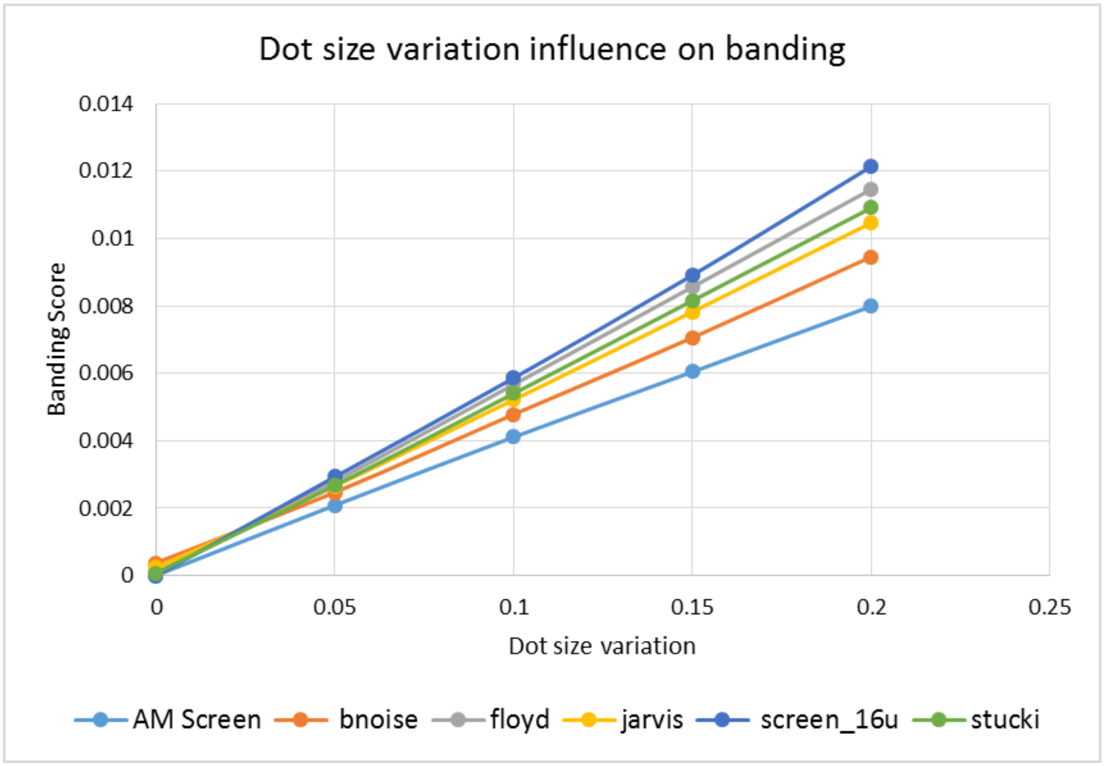


Figure 42: The influence of the dot size variation on banding score

Dot size maximal increase	AM Screen	bnoise	floyd	jarvis	screen_16u	stucki
0	7.036E-06	0.00038	8.8E-05	0.000237	7.42186E-12	7.84646E-05
0.05	0.0020873	0.002471	0.002805	0.002659	0.002929444	0.00267792
0.1	0.0041125	0.004784	0.00567	0.005219	0.00586366	0.00540342
0.15	0.0060444	0.007062	0.008565	0.00783	0.00891462	0.00815946
0.2	0.0079918	0.009463	0.011464	0.010468	0.0121437	0.01091252

Figure 43: The mean values of the banding score in dot-size-variation test

Dot size maximal increase	AM Screen	bnoise	floyd	jarvis	screen_16u	stucki
0	3.65E-06	0.000151	3.67E-05	0.000304	6.87292E-12	3.07848E-05
0.05	0.0013029	0.001524	0.00155	0.001375	0.00150915	0.001396437
0.1	0.0025115	0.002683	0.002903	0.002547	0.002949192	0.002623327
0.15	0.0035672	0.003898	0.004163	0.003601	0.004313027	0.003732398
0.2	0.0046743	0.004962	0.005415	0.004595	0.00573359	0.004796295

Figure 44: The standard deviation values of the banding score in dot-size-variation test

It can be seen that the banding is linearly increasing with all algorithms, and the gap between the methods is wider as we increase the dot size. Nevertheless, if we look at the next figure, we can see that the sensitivity of screen16u is increasing from the first time we add the banding distortion. In addition, The Blue Noise algorithm lowered its dominance compared to other algorithms.

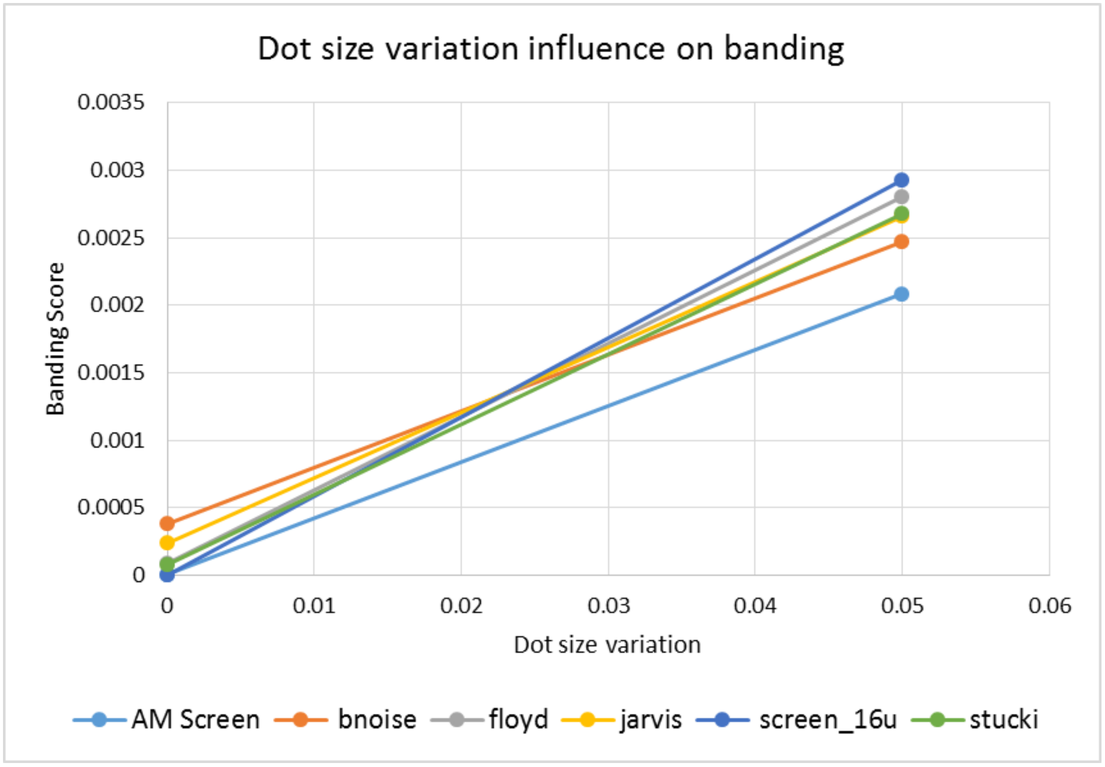


Figure 45: The change in the trend the first time bands are introduced

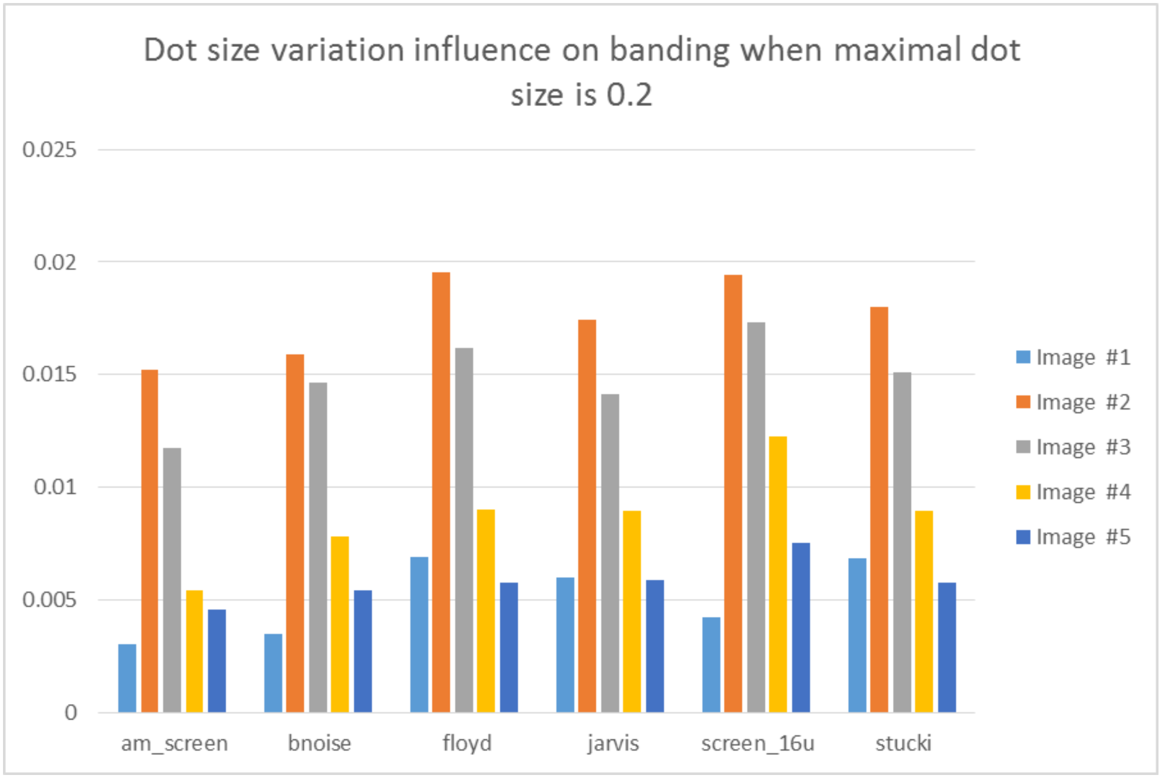


Figure 46: The influence of the dot size on the banding score in each test. The maximal increase in dot size is 0.2

Color plane misregistration

This group of tests was performed to examine the influence of the Color plane misregistration on the image similarity, graininess and banding. We used the same dataset and default configuration as described in the previous section, but we changed the value of *max_shifting* parameter in every run and observed the results. The *max_shifting* ranges from zero to fifty with a step of five pixels.

The following graph shows the results:

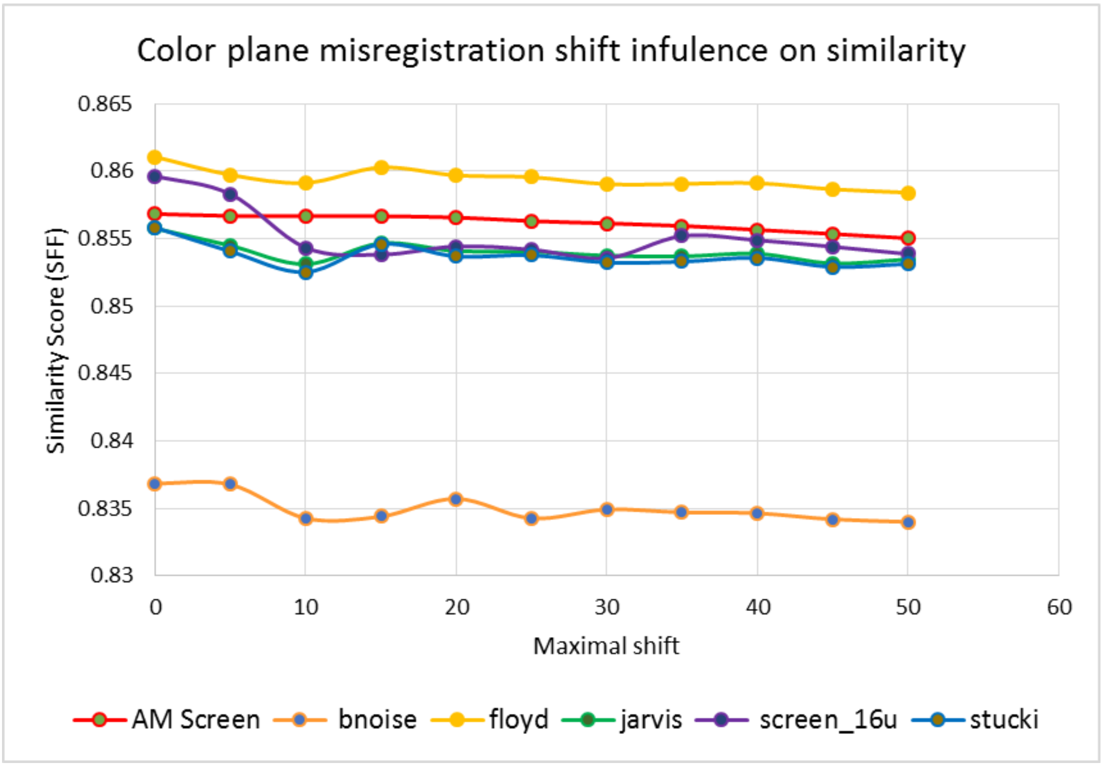


Figure 47: The influence of color plane misregistration on similarity score (SFF)

Maximal Shift	AM Screen	bnoise	floyd	jarvis	screen_16u	stucki
0	0.85683	0.836852	0.861068	0.855762	0.859642	0.85583
5	0.85669	0.836768	0.859736	0.854484	0.85831	0.854056
10	0.856674	0.83427	0.859138	0.853116	0.854332	0.852524
15	0.856664	0.834396	0.860282	0.854656	0.853816	0.854596
20	0.856572	0.835716	0.859692	0.85409	0.854426	0.853686
25	0.856302	0.834264	0.859564	0.854034	0.854188	0.853774
30	0.856134	0.834916	0.859046	0.853738	0.853548	0.853228
35	0.855946	0.834704	0.859062	0.853684	0.855196	0.853296
40	0.855658	0.83464	0.859118	0.853884	0.854886	0.853574
45	0.855352	0.834194	0.858664	0.853166	0.854404	0.85291
50	0.855034	0.833998	0.858418	0.853474	0.853866	0.853128

Figure 48: The mean values of the similarity score (SFF) in color-plane-misregistration test

Maximal Shift	AM Screen	bnoise	floyd	jarvis	screen_16u	stucki
0	0.024208679	0.032322863	0.022284619	0.024984616	0.023916724	0.025324058
5	0.024290447	0.032024997	0.022504087	0.025209841	0.024282231	0.025816921
10	0.024380347	0.034599347	0.022980837	0.025964076	0.025332781	0.026882035
15	0.024368594	0.034130979	0.02249899	0.025402081	0.025042764	0.025781278
20	0.024405356	0.033180112	0.022640378	0.02553367	0.025088702	0.025980482
25	0.024487738	0.034523529	0.022787087	0.025673486	0.025348412	0.026080016
30	0.024537847	0.033392477	0.022951236	0.025771796	0.025596782	0.026278372
35	0.024607455	0.03340859	0.023032288	0.025859892	0.024792209	0.026272514
40	0.02470252	0.033545616	0.022901453	0.025724125	0.02494534	0.026029785
45	0.024812424	0.033785974	0.023024682	0.025835013	0.02516629	0.026186731
50	0.024916841	0.033383251	0.023156806	0.025797532	0.025203286	0.026211924

Figure 49: The standard deviation values of the similarity score (SFF) in color-plane-misregistration test

From this graph, it can be seen that all algorithms degrade when shifting applies, even within the first shift of 5 pixels (in simulation resolution) which are equivalent to a portion of a pixel. Another observation is that the *screen16u* algorithm is much more sensitive after 10 pixels shift.

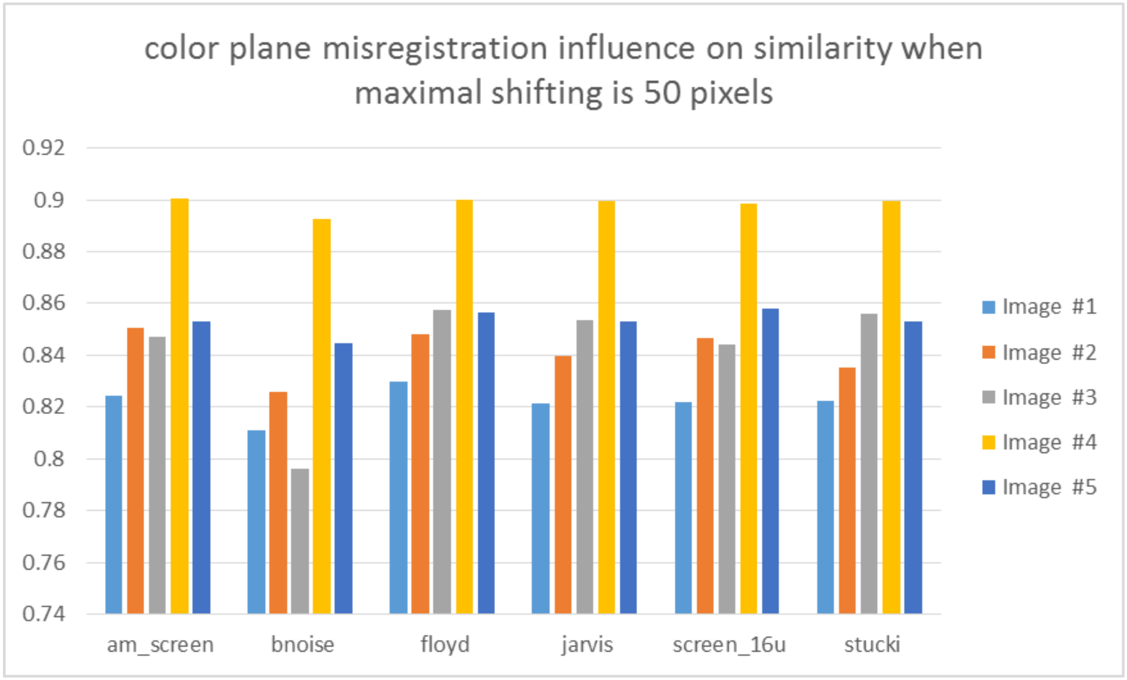


Figure 50: The influence of the color plane misregistration on the similarity score in each test. The maximal shift is 50 pixels

Figure 51 shows the color-plane-misregistration influence on graininess. The results show that the *screen16u* algorithm generates much grainier images than other algorithms. Nevertheless, it has a trend of reduced graininess when shifting increases. It can be explained by the fact that the image becomes more smeared and less grainy at some level of shifting. When reaching a 30 pixels level, the graininess is increasing (Depending on the exact screen used and its own cell size).

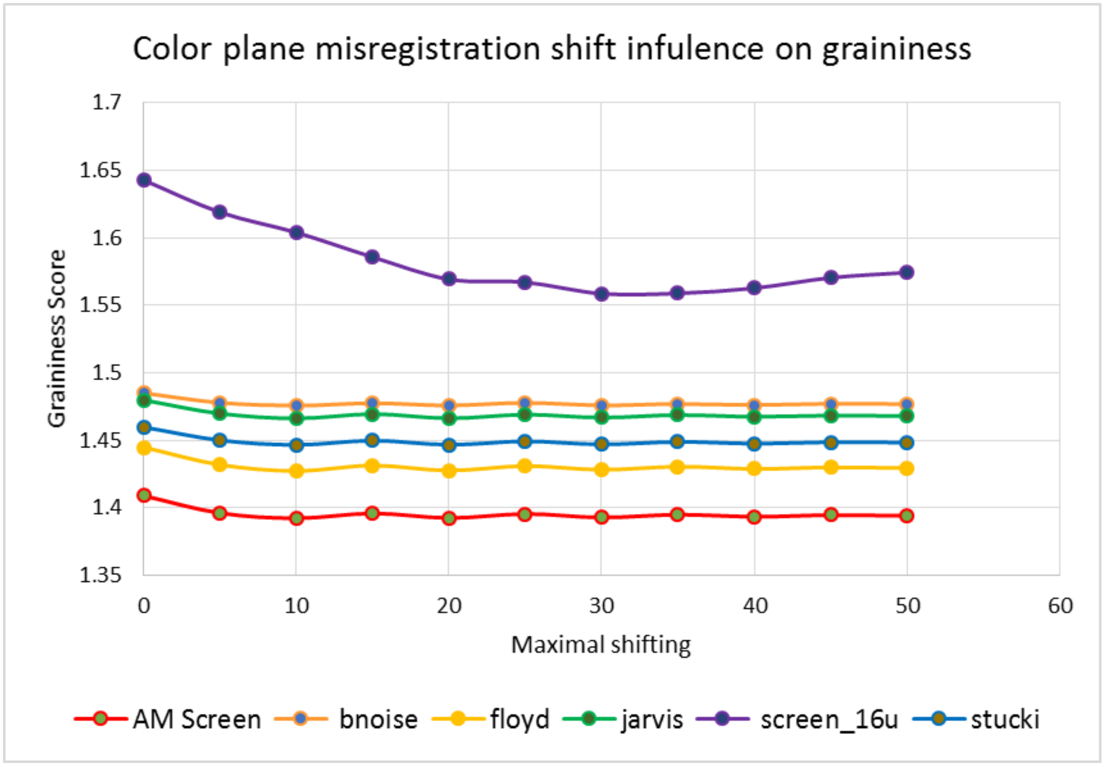


Figure 51: The influence of color plane misregistration on graininess score

Maximal Shift	AM Screen	bnoise	floyd	jarvis	screen_16u	stucki
0	1.408816	1.484972	1.44483	1.47976	1.64266	1.459718
5	1.396064	1.477804	1.431828	1.469786	1.618924	1.449988
10	1.392196	1.475688	1.427296	1.46623	1.60395	1.446584
15	1.395862	1.477478	1.431304	1.46942	1.585718	1.449794
20	1.392402	1.475786	1.427608	1.466296	1.569308	1.446688
25	1.395402	1.47774	1.430948	1.468998	1.567094	1.449228
30	1.392768	1.475882	1.428174	1.466834	1.558656	1.44698
35	1.394868	1.476948	1.430452	1.468688	1.55886	1.448808
40	1.393334	1.476138	1.428746	1.467398	1.5627	1.447586
45	1.39453	1.477112	1.429918	1.46832	1.57042	1.448616
50	1.393966	1.476826	1.429406	1.467922	1.574258	1.4483

Figure 52: The mean values of the graininess score in color-plane-misregistration test

Maximal Shift	AM Screen	bnoise	floyd	jarvis	screen_16u	stucki
0	0.560342453	0.543481577	0.45416628	0.483215789	0.643712916	0.459397978
5	0.560656902	0.544300589	0.450706909	0.48346827	0.607321322	0.460566612
10	0.560000934	0.543522818	0.447912039	0.481941672	0.580220544	0.45934794
15	0.560343953	0.544276948	0.449725145	0.482658784	0.565978144	0.459978298
20	0.559770822	0.543764619	0.44753285	0.48145798	0.560076801	0.458842669
25	0.559978009	0.544136706	0.448936563	0.482235441	0.565369047	0.459184262
30	0.559578598	0.543605296	0.447458446	0.481542149	0.560465228	0.458388453
35	0.559717299	0.543001525	0.448327177	0.482038053	0.557543256	0.458687107
40	0.559448086	0.543224047	0.447302579	0.481508768	0.557060517	0.45821821
45	0.559557344	0.54359604	0.447675336	0.481678625	0.564160308	0.458488719
50	0.55940922	0.542893894	0.447344341	0.48141548	0.566501651	0.45838044

Figure 53: The standard deviation values of the graininess score in color-plane-misregistration test

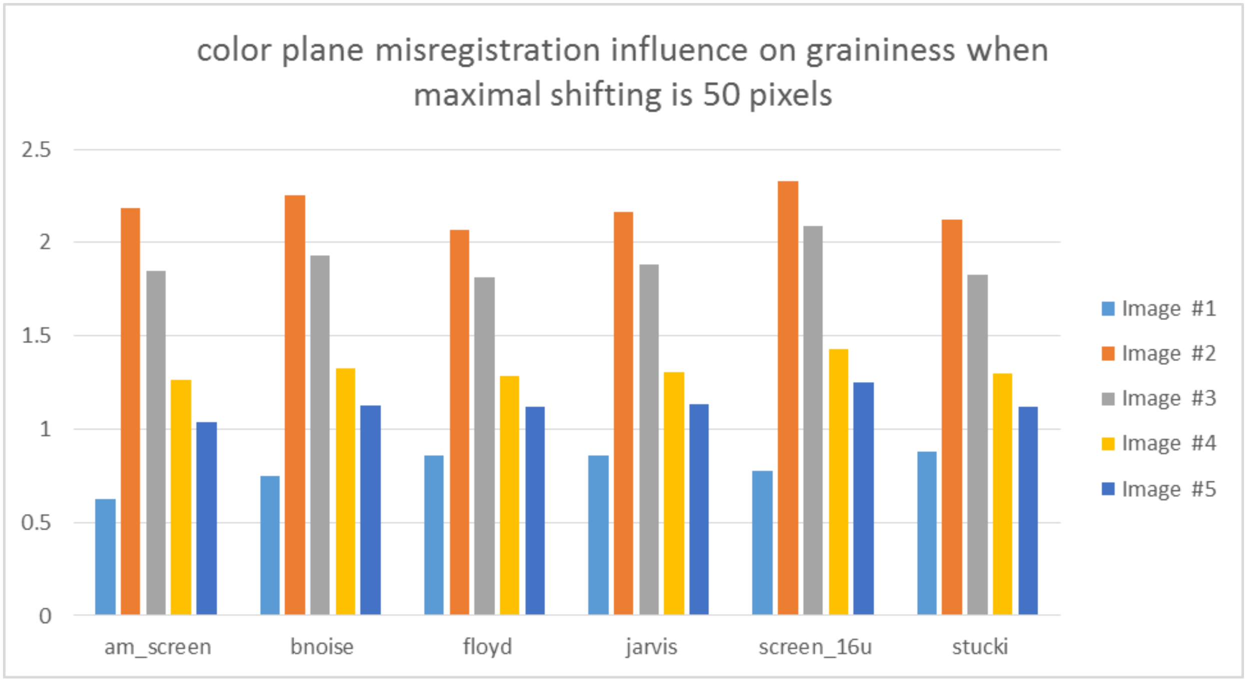


Figure 54: The influence of the color plane misregistration on the graininess score in each test. The maximal shift is 50 pixels

Figure 55 is the last graph for this group and it depicts the color-plane-misregistration influence on the banding. While most algorithms show almost no influence on banding as we increase the amount of shifting, the *screen16u* algorithm dramatically increases starting at the first shift, and even more so in 15 pixels shift.

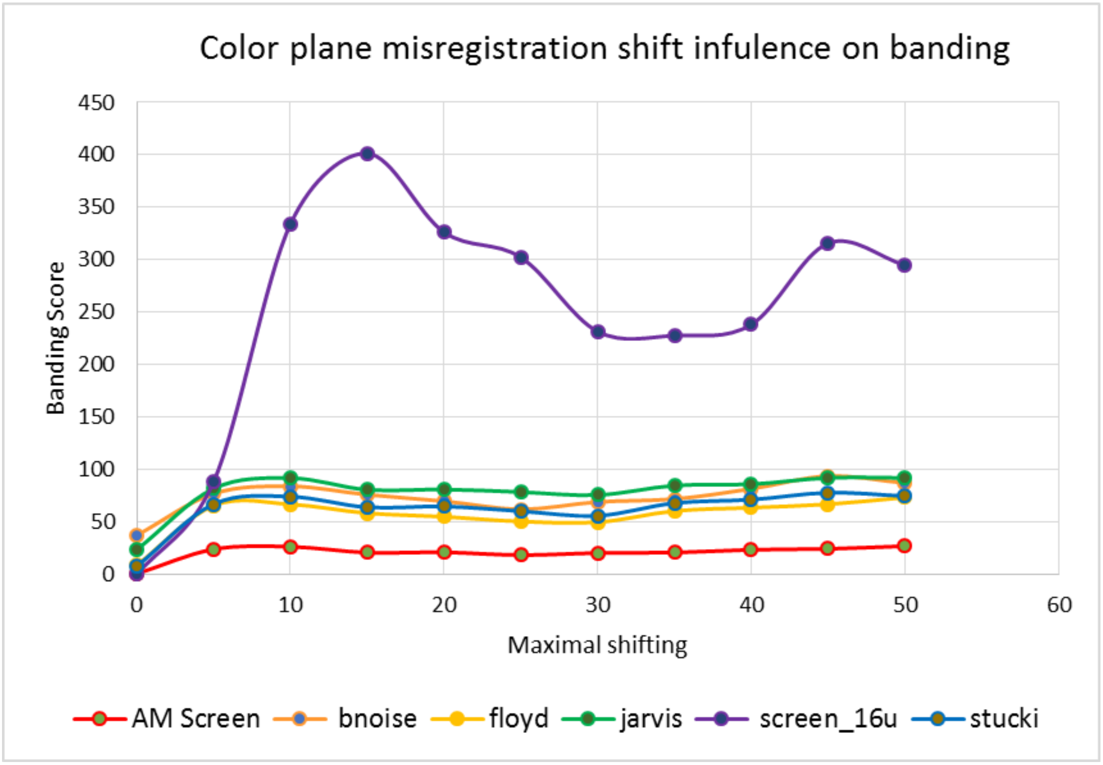


Figure 55: The influence of color plane misregistration on banding score

Maximal Shift	AM Screen	bnoise	floyd	jarvis	screen_16u	stucki
0	7.03576E-06	0.000374366	0.000088002	0.00023672	7.42186E-12	7.84646E-05
5	0.000240684	0.000770796	0.000657533	0.000818806	0.000885281	0.000671322
10	0.000264676	0.00084095	0.000669425	0.000919199	0.003341965	0.000742012
15	0.000208568	0.000761128	0.000584479	0.000808318	0.004014446	0.000640942
20	0.000211804	0.000699554	0.000551226	0.000810965	0.003263585	0.000648676
25	0.000185067	0.000622416	0.000506012	0.000786175	0.003023996	0.000603434
30	0.000205016	0.000691614	0.000498536	0.000757624	0.002316774	0.000560527
35	0.000209526	0.000720184	0.000605203	0.000847766	0.002277853	0.000681206
40	0.000236126	0.000816604	0.000637064	0.000861118	0.002382079	0.000713428
45	0.000244402	0.000938334	0.000669772	0.000921274	0.003157422	0.000779476
50	0.000270536	0.00087278	0.000734064	0.000920756	0.002949494	0.00074816

Figure 56: The mean values of the banding score in color-plane-misregistration test

Maximal Shift	AM Screen	bnoise	floyd	jarvis	screen_16u	stucki
0	3.65037E-06	0.000150604	3.6712E-05	0.000304391	6.87292E-12	3.07848E-05
5	0.000147648	0.000964363	0.000822619	0.000810302	0.000726388	0.000863437
10	0.000179791	0.001022909	0.000873724	0.001083872	0.003096052	0.00107514
15	0.000132096	0.000902825	0.000710128	0.000721267	0.004002402	0.000732505
20	0.000141397	0.000896889	0.000700492	0.000871157	0.003158123	0.000881308
25	0.000117486	0.000601765	0.000573895	0.000728249	0.002735569	0.000732763
30	0.000130647	0.000774917	0.00051661	0.000642637	0.002013474	0.000628862
35	0.000136504	0.000868274	0.000805257	0.000930878	0.002010645	0.000956714
40	0.00016247	0.000982957	0.00070481	0.000723431	0.002114157	0.000803403
45	0.000160389	0.001056543	0.000881796	0.00105126	0.002839676	0.001115537
50	0.000176422	0.0010981	0.000836716	0.000876752	0.002921293	0.000885391

Figure 57: The standard deviation values of the banding score in color-plane-misregistration test

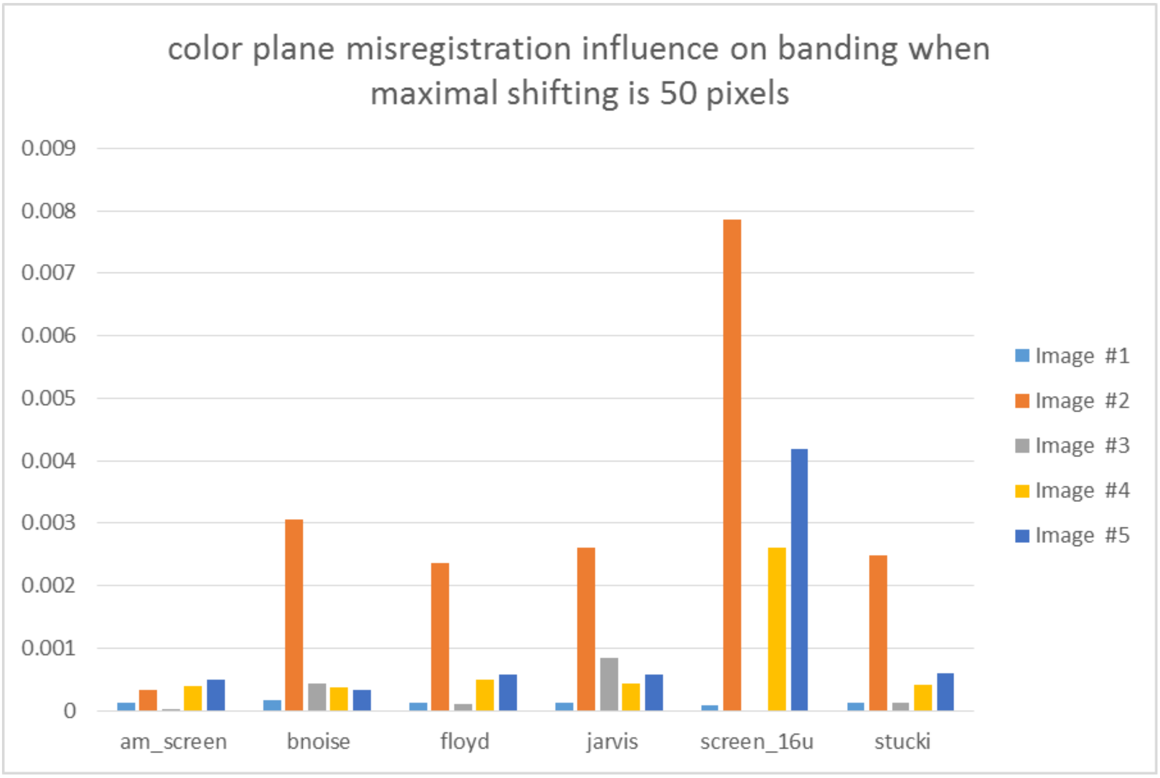


Figure 58: The influence of the color plane misregistration on the banding score in each test. The maximal shift is 50 pixels

While observing the result images, we see that this artifact causes color banding when using the *screen16u* algorithm (see figure 59a). For the sake of comparison, no bands appear when using floyd algorithm (see figure 59b).This phenomenon shows that some algorithms are very sensitive to the color plane misregistration while others are less sensitive. Moreover, this observation shows that while simple and straightforward algorithm produces good result on a “perfect” printer, it might lead to poor results when a very small misalignment occurs, due to its inherent sensitivity.

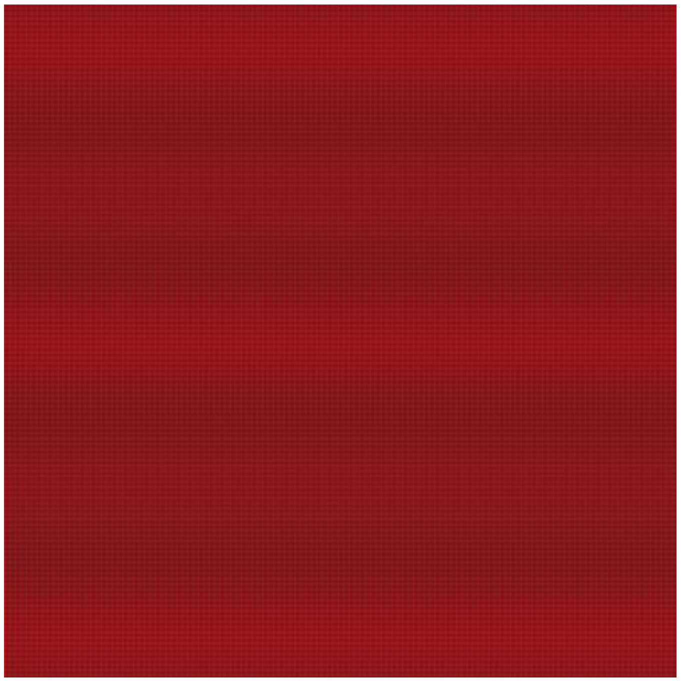


Figure 59a: A result image create by screen16u algorithm. One can notice color bands.

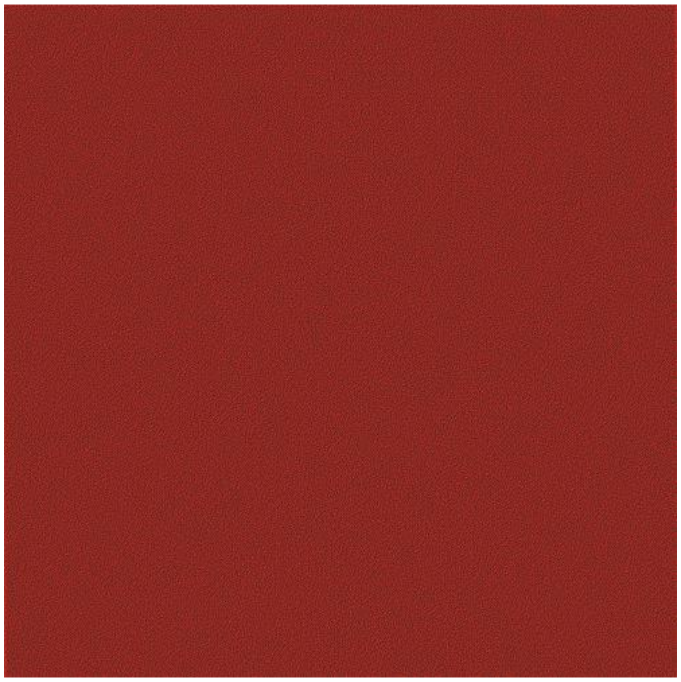


Figure 59b: A result image create by floyd algorithm. Color bands do not exist.

Write-Head Banding

This group of tests examine the impact of the Write-head banding on the algorithms and its influence on similarity, graininess and banding. Once again, we used the default configuration: The write head size is 20 pixels, and the maximal gap is 4 pixels. In these set of tests we compared two result images: first, a result image of a banding distortion and second, a result image without one. We observed which algorithm produces better results in average. The following figures shows a section of the result images:

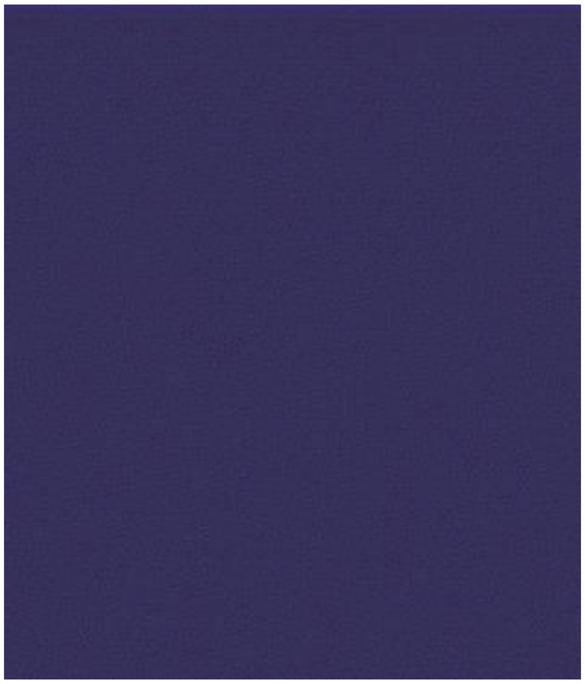


Figure 60a: A section of a printing result using Jarvis algorithm.

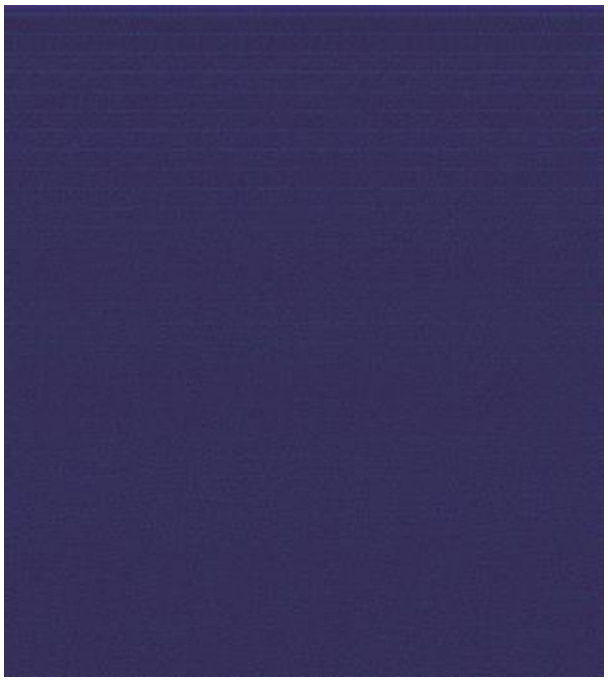


Figure 60b: A section of a printing result using Jarvis algorithm, with write-head banding distortion.

The following diagram describes the impact of the write-head banding distortion on similarity:

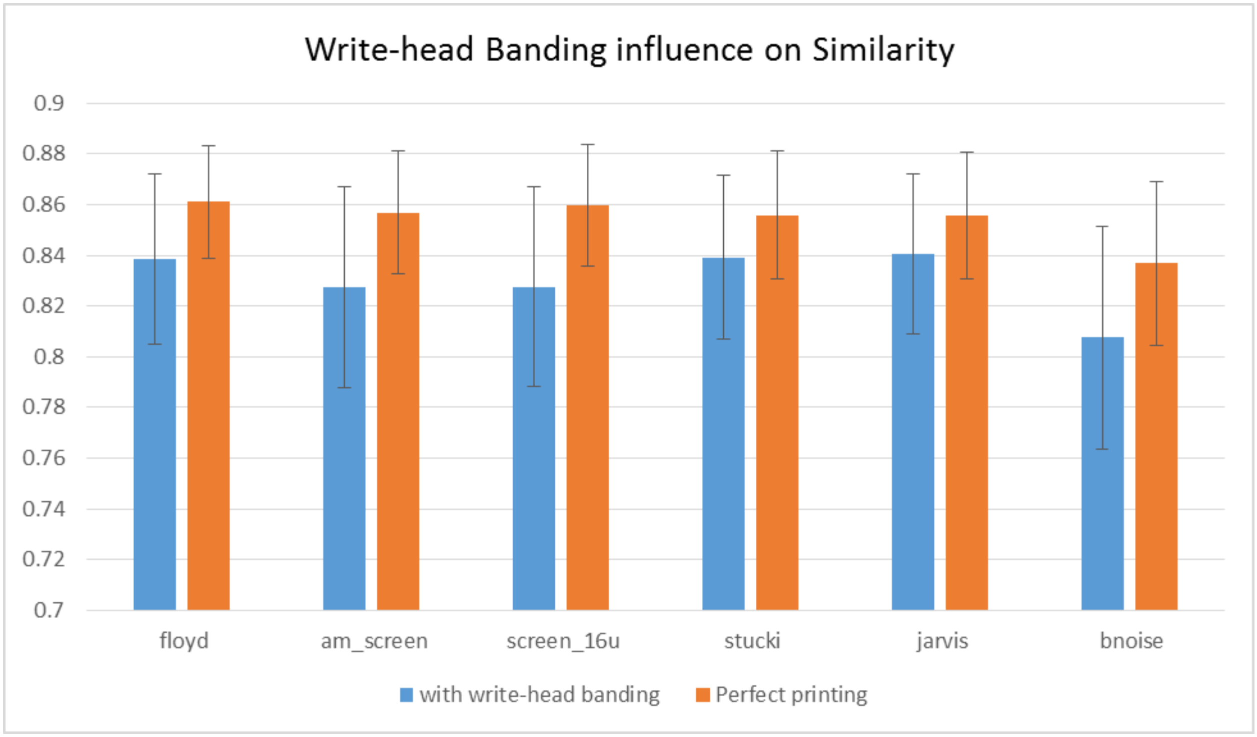


Figure 61: The influence of write-head banding on similarity score (SFF).

Algorithm	mean with write-head banding	SD with write-head banding	Mean with perfect printer	SD with perfect printer	Difference
floyd	0.838464	0.033697613	0.861068	0.022284619	0.022604
am_screen	0.827178	0.039601664	0.85683	0.024208679	0.029652
screen_16u	0.827656	0.039281255	0.859642	0.023916724	0.031986
stucki	0.839264	0.032492158	0.85583	0.025324058	0.016566
jarvis	0.840598	0.031706806	0.855762	0.024984616	0.015164
bnoise	0.807636	0.043994082	0.836852	0.032322863	0.029216

Figure 62: The Standard deviation and mean values of the similarity score (SFF) in write-head banding test

Clearly, Floyd, Jarvis and Stucki have less impact than other algorithms.

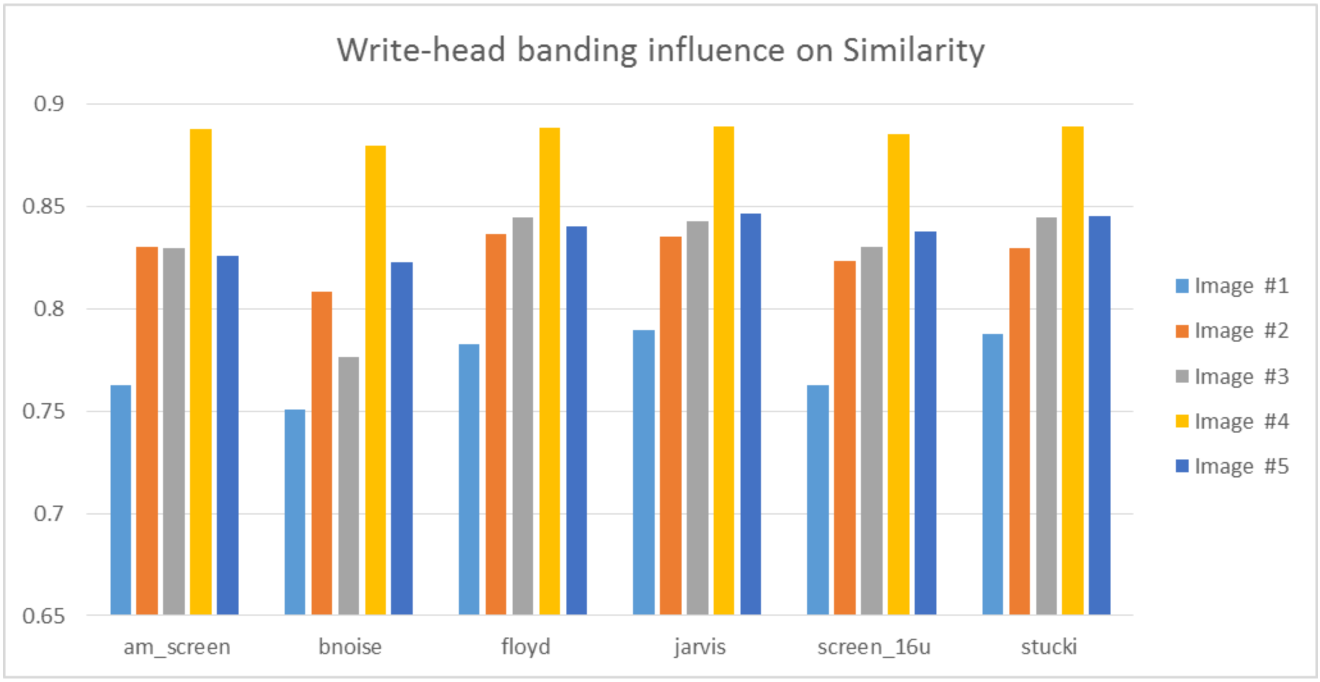


Figure 63: The influence of the write-head banding on the Similarity score in each test.

The next diagram describes the impact of the write-head banding distortion on graininess:

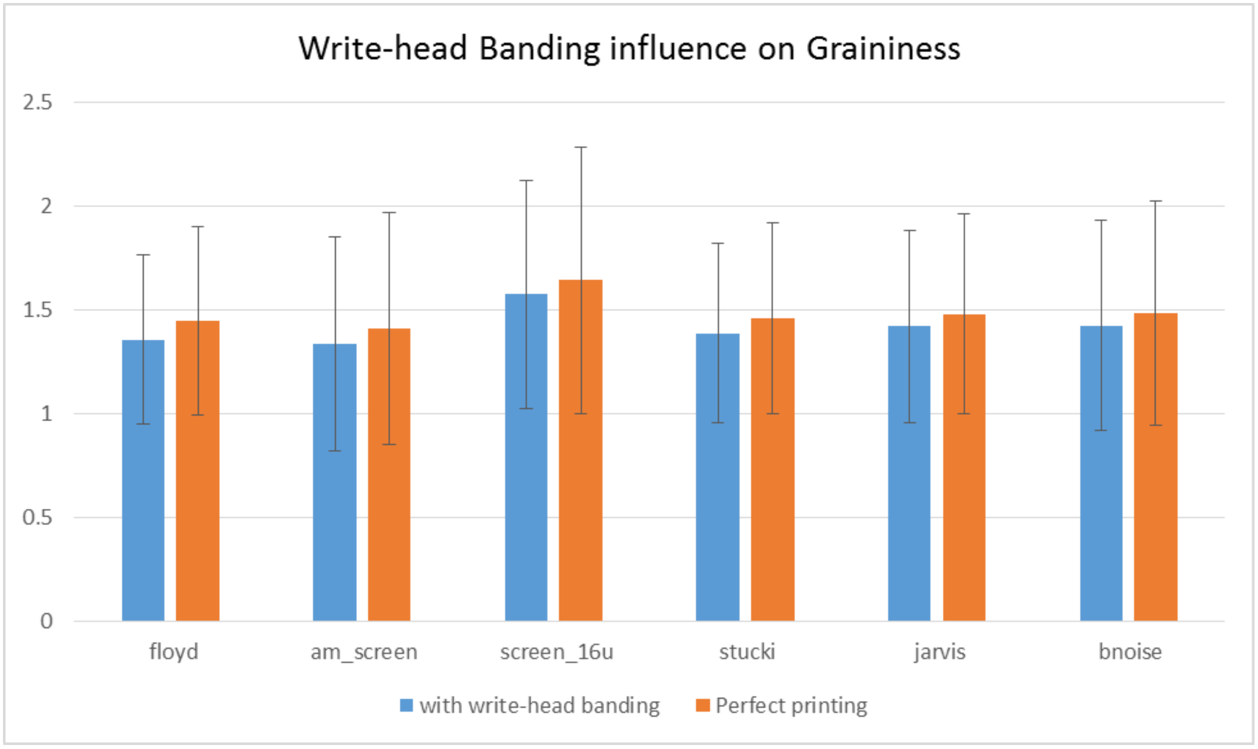


Figure 64: The influence of write-head banding on graininess score.

Algorithm	mean with write-head banding	SD with write-head banding	Mean with perfect printer	SD with perfect printer	Difference
floyd	1.35703	0.409170504	1.44483	0.45416628	0.0878
am_screen	1.336968	0.515233338	1.408816	0.560342453	0.071848
screen_16u	1.574434	0.549273268	1.64266	0.643712916	0.068226
stucki	1.387994	0.431063333	1.459718	0.459397978	0.071724
jarvis	1.420458	0.465037413	1.47976	0.483215789	0.059302
bnoise	1.425116	0.506902246	1.484972	0.543481577	0.059856

Figure 65: The Standard deviation and mean values of the graininess score in write-head banding test

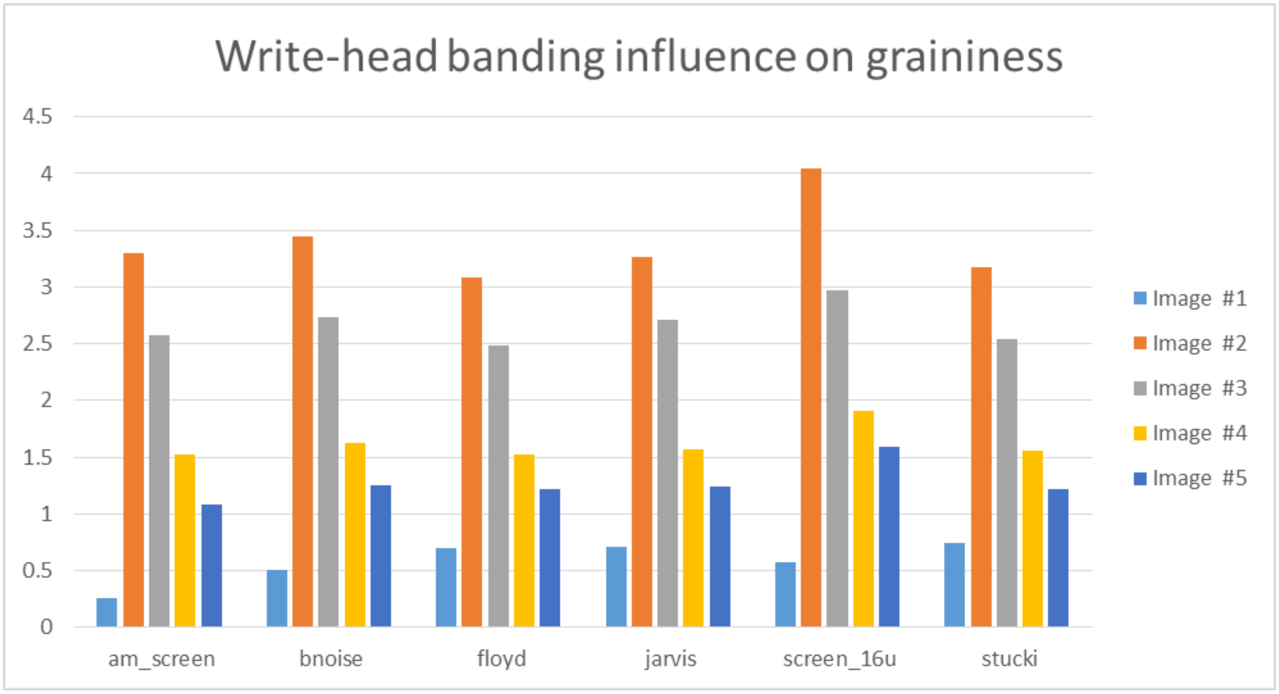


Figure 66: The influence of the write-head banding on the graininess score in each test.

From this diagram, we learn that there is a small degradation in the graininess score, but it is not substantial. In addition, there is no great difference between the examined algorithms.

The last diagram describes the impact of the write-head banding distortion on banding:

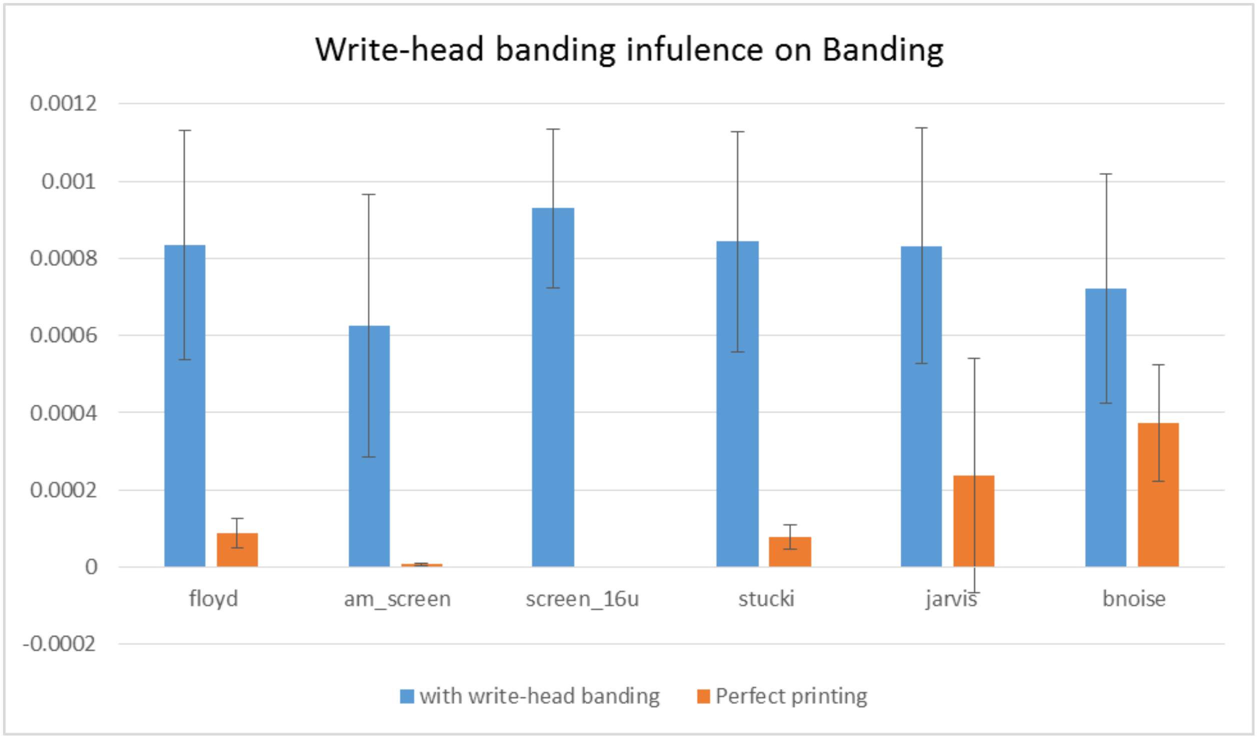


Figure 67: The influence of write-head banding on banding score.

Algorithm	mean with write-head banding	SD with write-head banding	Mean with perfect printer	SD with perfect printer	Difference
floyd	0.000834868	0.000296714	0.000088002	3.6712E-05	0.000746866
am_screen	0.000624596	0.000339866	7.03576E-06	3.65037E-06	0.00061756
screen_16u	0.000929824	0.000206277	7.42186E-12	6.87292E-12	0.000929824
stucki	0.000842972	0.000285296	7.84646E-05	3.07848E-05	0.000764507
jarvis	0.000831412	0.000304744	0.00023672	0.000304391	0.000594692
bnoise	0.000721964	0.000296441	0.000374366	0.000150604	0.000347598

Figure 68: The Standard deviation and mean values of the banding score in write-head banding test

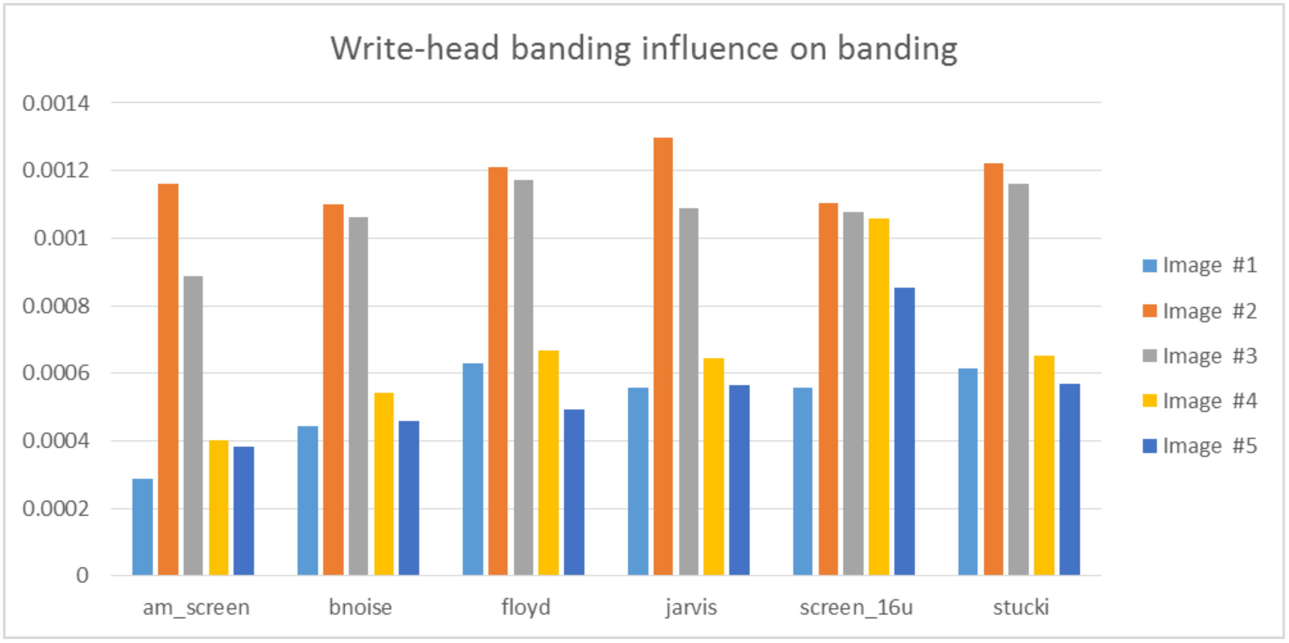


Figure 69: The influence of the write-head banding on the banding score in each test.

Since we added bands (write-head bands) to each image, it was expected that the banding score would increase. Most algorithms have a similar level of bands, but not an identical one. It can be seen that even without our added distortions, most of the algorithms create some small level of banding on their own. It is also easy to see that the contribution of the write-head banding distortion to Blue Noise algorithm's result was much smaller in comparison with other algorithms.

Conclusion

The results shown in the previous section demonstrated the assumption that some algorithms are more sensitive to artifacts that might happen on real life printing scenarios. These results can be seen only on printed images. When using this simulation tool it is easy to predict these errors, understand the algorithm sensitivity and even match it to the most suitable printing technology. Therefore, the necessity of using this evaluation on halftoning algorithms is obvious. When a new algorithm is introduced, it is not enough to assume a perfect printer on laboratory environment. Instead, testing it on real-life printer simulator is essential. Prior to this work, the only way to examine a new halftoning algorithm was by actual printing on various printer models. These printing tests might include many iterations during development and debugging process. By using this simulator, which modeled the process mathematically, the evaluation can be made without actual printing, thus saves time, supplies and efforts.

The tests described in the previous section examine only six classic algorithms, and show a demonstration of the tool. However, the expectation is that both the academia and the industry fields, which produce halftoning algorithms, will use this tool. This evaluation tool will allow those developers to evaluate their own intellectual property algorithms in real life scenarios. The evaluation result could shed a light on the effectiveness of new sophisticated techniques.
This list of algorithms is configurable and by that can host any proprietary algorithm. Using this tool could result better halftoning algorithms, or a better match between exiting algorithms and specific printing technology.

For future research, additional artifacts might be added to this tool in order to assess other sensitivities such as jitter and ghosting. Moreover, other scoring methods can be used in addition to banding, graininess and similarity.

References

[1] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity", IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004.

[2] Hila Nachlieli, Doron Shaked, Shai Druckman, Maya Shalev, and Yona Yaniv, “Automatic mechanical band perceptual evaluation”, in Proceedings IS&T International Conference Digital Printing Technology Digital Fabrication (NIP25), September 2009, pp. 495-498.

[3] Farhan A. Baqai et-al, “Digital Color Halftoning [Problems, algorithms, and recent trends]”, IEEE SIGNAL PROCESSING MAGAZINE [87] JANUARY 2005

[4] Vishal Monga, Niranjan Damera-Venkata, Hamood Rehman, and Brian L. Evans, “Halftoning toolbox for MATLAB”, version 1.2.

[5] Hila Nachlieli and Doron Shaked, “Measuring the quality of quality measures”, IEEE Transactions on Image Processing, vol. 20, no. 1, January 2011.

[6] Amir Kolaman and Orly Yadid-Pecht, “Quaternion structural similarity: A new quality index for color images”, IEEE Transactions on Image Processing, vol. 21, no. 4, April 2012.

[7] Hua-Wen Chang et-al, “Sparse Feature Fidelity for Perceptual Image Quality Assessment”, IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 22, NO. 10, OCTOBER 2013

[8] Zhou Wang et-al, “Multi-Scale structural similarity for image quality assessment” , Proceedings of the 37th IEEE Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, Nov. 9-12, 2003.

[9] Y. Ju, T. Kashti, T. Frank, D. Kella, D. Shaked, M. Fischer, R. Ulichney, and J. Allebach, “Black-Box Models for Laser Electrophotographic Printers – Recent Progress”, IS&T NIP29 (29th Int. Conf. on Digital Printing Technologies), Seattle, WA, Sep 2013.

[10] Jason Youn, Jian Sun, Yanling Ju, Tamar Kashti, Tal Frank, Dror Kella, Mani Fischer, Robert Ulichney, Guy Adams, and Jan Allebach, “Effect of image capture device on the accuracy of black-box printer models”, Proc. SPIE 9015, Color Imaging XIX: Displaying, Processing, Hardcopy, and Applications, 90150T (January 8, 2014).

[11] Weibao Wang, Peter Bauer, Jerry Wagner, and Jan P. Allebach, “MFP scanner diagnostics using a self-printed target to measure the modulation transfer function”, Proc. SPIE 9016, Image Quality and System Performance XI, 90160A (January 7, 2014); doi:10.1117/12.2037959.

[12] Alastair Reed, David Berfanger, Yang Bai, and Kristyn Falkenstern, “Full-color visibility model using CSF which varies spatially with local luminance”, Proc. SPIE 9027, Imaging and Multimedia Analytics in a Web and Mobile World 2014, 902705 (March 3, 2014); doi:10.1117/12.2045408.

[13] Isaac Amidror& Roger D. Hersch, “Analysis of the microstructures (‘‘rosettes’’) in the superposition of periodic layers”, Journal of Electronic Imaging 11(3), 316–337 (July 2002).

[14] Carl-Magunus Fahlcrantz & Per-Ake Johansson, “A comparison of different print mottle evaluation models”, 2004

[15] Jin-Young Kim et-al, “Design of Color screen sets for robustness to color plane misregistration”, IEEE International Conference on Image Processing, 2011

[16] Roy S. Berns, Animesh Bose, and Di-Yuan Tzeng. “The Spectral modeling of large-format ink-jet printers”, RIT Munsell Color Science Laboratory. December 1996.

[17] Thrasyvoulos N. Pappas, Chen-Koung Dong, and David L. Neuhoff, “Measurement of printer parameters for model-based halftoning”, Journal of Electronic Imaging 2(3), pp. 193--204. July 1993.

[18] Thrasyvoulos N. Pappas and David L. Neuhoff, “Least-squares model-based halftoning”, IEEE Transactions on Image Processing 8(8), pp. 1102--1116. August 1999.

[19] Je-Ho Lee and Jan P. Allebach, “Inkjet printer model-based halftoning”, IEEE Transactions on Image Processing 15(5). May 2005.

[20] Yanling Ju, Dhruv Saxena, Tamar Kashti, Dror Kella, Doron Shaked, Mani Fischer, Robert Ulichney, and Jan P. Allebach, “Modeling large-area influence in digital halftoning for electrophotographic printers”, Proceedings SPIE-IS&T Electronic Imaging. January 2012.

[21] Lawrence A. Taplin, “Spectral modeling of a six-color inkjet printer”, Masters Thesis, RIT. December 2001.

[22] Thrasyvoulos N. Pappas, Jan P. Allebach, and David L. Neuhoff, “Model-based digital halftoning”, IEEE Signal Processing Magazine. July 2003.

[23] Farhan A. Baqai and Jan P. Allebach, “Halftoning via direct binary search using analytical and stochastic printer models”, IEEE Transactions on Image Processing 12(1). January 2013.

[24] R.W. Floyd and L. Steinberg, “An adaptive algorithm for spatial grey scale”, in Proc. Society for Information Display, 1976, vol. 17/2, pp. 75-77.

[25] David R. Wyble, “A Critical review of spectral models applied to binary color printing”, RIT. 2000.

[26] Robert Ulichney, “Digital halftoning”, The MIT Press 1993.

[27] Henry R. Kang, “Digital Color Halftoning”, 1999

[28] ISO-13660 “Information technology -- Office equipment -- Measurement of image quality attributes for hardcopy output -- Binary monochrome text and graphic images”, 2001

[29] T. Mitsa and K. Parker, "Digital Halftoning using a blue noise mask", J. Opt. Soc. Am. A, Vol. 9, November 1992, pp. 1920-1929.

[30] R. Eschbach and K. Knox, "Error diffusion algorithm with edge enhancement", J. Opt. Soc. Am. A, Vol. 8, No. 12, December 1991, pp. 1844-1850.

[31] Michael Kriss “Handbook of Digital Imaging”

[32] Gooran, Sasan. 2005, “Hybrid Halftoning, A Useful Method for Flexography”, Journal of Imaging Science and Technology 49(1): 85–95.

[33] Gooran, Sasan. 2006. “A Novel Hybrid AM/FM Halftoning Based on Multi-Level Halftoning”, Journal of Imaging Science And Technology 50 (2): 157–167.

[34] Paula Žitinski Elías, “Halftoning for Multi-Channel Printing”, 2014

[35] Puneet Goyal et-al “Electro-Photographic model based stochastic clustered-dot halftoning with direct binary search”, IEEE International Conference on Image Processing, 2011

[36] P. Stucki, “MECCA—A multiple-error correcting computation algorithm for bilevel image hardcopy reproduction”, IBM Research Laboratory, Zurich, Switzerland, Res. Rep. RZ1060, 1981.

[37] J.F. Jarvis, C.N. Judice, and W.H. Ninke, “A survey of techniques for the display of continuous-tone pictures on bilevel displays”, Comp. Graph. Image Proc., vol. 5, pp. 13-40, 1976.

[38] https://www.flickr.com.

תקציר

שיטת ההפטון (Halftone) הינה טכניקה ידועה המשמשת כדי ליצור אשליה של תמונת עם דרגות אפור רציפות באמצעות סידור חכם של אלמנטים בינאריים. טכניקה זו נמצאת בשימוש נפוץ בתעשיית ההדפסה. גישות רבות הוצעו הן עבור הדפסה צבעונית והן עבור הדפסה בשחור-לבן. לכל גישה יתרונות וחסרונות, אך לא קיים אלגוריתם יחיד המשמש עבור כל המטרות. למעשה, אין כלי מדידה אובייקטיביים להערכת אלגוריתמים מסוג זה. הבעיה המרכזית היא כי ישנם אלגוריתמים שיכולים להיראות מאוד מרשימים עבור המפתחים של האלגוריתם, כיוון שהתוצאות נראות טוב בסביבת המעבדה, אך כאשר נעשה בהם שימוש בזמן אמת, התוצאות עלולות לצאת בעיתיות ולא צפויות.

תופעה זו עלולה לקרות לעתים קרובות עקב העובדה שמפתחים בדרך כלל מניחים שהמדפסת המיועדת הינה מדפסת "אידאלית", אך במציאות למדפסות יש עיוותים רבים. אלגוריתמים שונים יכולים להיות רגישים לעיוותים אלו. מטרתה של עבודה זו הוא לבנות כלי בדיקה אשר נועד לטפל בבעיה זו על ידי מידול והדמיית תהליך ההדפסה הלכה למעשה, כפי שקורה במדפסות אמיתיות, תוך כדי הדמיית אותם עיוותים. הכלי מודד את הרגישות של אלגוריתמי יעד אלו לעיוותים שונים. בעבודה זו אנו מדדנו 6 אלגוריתמים מקטגוריות שונות, הפעלנו את כלי הבדיקה וכמו כן הצגנו את תוצאות הבדיקה.

,

הבינתחומי הרצליה | בית ספר אפי ארזי למדעי המחשב

המרכז הבינתחומי בהרצליה
בית-ספר אפי ארזי למדעי המחשב

כלי להערכת אלגוריתמי הדפסה

סיכום עבודת גמר המוגשת כמילוי חלק מהדרישות לקראת תואר מוסמך במסלול מחקרי במדעי המחשב

על-יד אריאל שמרלינג

העבודה בוצעה בהנחיית פרופ' יעקב הל-אור וד"ר קרל סטאלין

אוקטובר 2015