Automatic visual inspection and defect detection on Variable Data Prints

Marie Vans, Sagi Schein, Carl Staelin, Pavel Kisilev, Steven Simske, Ram Dagan, Shlomo Harush

HP Laboratories

HPL-2008-163R1

Keyword(s):

Variable data printing, high-speed inspection, print defect detection, scanning, GPU

Abstract:

We propose a system for automatic, on-line visual inspection and print defect detection for Variable -Data Printing (VDP). This system can be used to automatically stop the printing process and alert the operator to problems. We present the components required for constructing a vision-based inspection system and show that our approach is novel for the high-speed detection of defects on variable data. When implemented in a high-speed digital printing press, the system will allow a single skilled operator to monitor and maintain several presses, reducing the number of operators required to run a shop floor of presses as well as reduce wasted consumables when a defect goes undetected.

External Posting Date: June 21, 2010 [Fulltext] Approved for External Publication
Internal Posting Date: June 21, 2010 [Fulltext]

Automatic visual inspection and defect detection on Variable Data Prints

Marie Vans[a], Sagi Schein[b], Carl Staelin[b], Pavel Kisilev[b], Steven Simske[a],Ram Dagan[c], and Shlomo Harush[c]

[a]Hewlett-Packard Labs, 3404 East Harmony Rd, MailStop 85, Fort Collins, CO, USA, 80528;

[b]Hewlett-Packard Labs, Technion City, Haifa, Israel, 32000;

[c]Hewlett-Packard Company, Einstein 10, Kiryat Weizmann , Ness Ziona, Rehovot, Israel,76101

Email: marie.vans@hp.com

We propose a system for automatic, on-line visual inspection and print defect detection for Variable Data Printing (VDP). This system can be used to automatically stop the printing process and alert the operator to problems. We present the components required for constructing a vision-based inspection system and show that our approach is novel for the high-speed detection of defects on variable data. When added to a high-speed digital printing press, the system allows a single skilled operator to monitor and maintain several presses, reducing the number of operators required to run a shop floor of presses as well as reduce wasted consumables when a defect goes undetected.

Keywords: Variable data printing, high-speed inspection, print defect detection, scanning, GPU

Introduction

Detecting defects on variable data prints currently requires a skilled operator to manually inspect prints. Example defects that may cause an operator to halt the machine include scratches, spots, missing dot clusters, streaks, and banding. There are two different methods the operator may use to check for defects. Prints may be inspected as they come off the press which implies that the operator is fully occupied with a single machine. Alternatively, the operator may collect stacks of prints for inspection. If a defect is present, paper is wasted. We discuss a system that can monitor the output of a digital printing press to observe the printed pages, automatically detect print defects, and alert the operator if necessary.

Automatic inspection of variable data products is virtually non-existent. Solutions exist for inspection of printed products such as labels and packaging; however, these do not usually change and once a perfect *reference* is decided upon, the inspection system will continually search items coming off the press using the same reference copy. Our solution for variable data requires the ability to change the reference on every print. For example, a customer job may require *personalization* of each print with a different name, address, or other information. A major requirement for this work is that each print be inspected and this means that defect detection for each page must be completed within a second, as the press prints at a speed of 2 meters per second. Therefore, the system must acquire both the reference and printed image, perform various image pre-processing activities, and determine whether a defect exists, all in real-time.

This paper describes a system designed to automatically detect defects on variable data prints. We describe each of the components needed to operate in real-time. An important contribution of this work is the development of an *image dis-similarity* measure for detecting defects. Image dis-similarity is actually related to *image fidelity*[1]. Image fidelity in this context refers to how close one image is to another *perfect* or *reference* image. We want a measure that can report the probability that the printed image contains defects visible to the human eye, i.e. can be detected. We show the results of a large-scale, proof-of-concept experiment using two print defect detection algorithms on 454 defective prints. Our proof-of-concept approach used a scan of the print and compared it with a scan of a *good* reference print. In this paper, we demonstrate that our algorithms meet two important requirements of the project: detection of most of the visible defects and a low false alarm rate. Although our motivation is to develop a product for digital presses, the system is applicable for all problems where finding defects or other differences between variable data images is necessary.

Prior Work

Newman and Jain[2] present a thorough survey of automated industrial inspection systems through 1993, primarily for defect detection. According to Newman & Jain[2], most inspection systems are either developed by machine vision companies or developed in-house by manufacturers for specific applications, with the vast majority going unreported in the literature.

A comprehensive literature search shows that while many more inspection systems have been developed and reported on, an inspection system for variable print data does not exist. We looked at offset print inspection systems, textile inspection systems (both patterned and non-patterned), generic web inspection systems, and Printed Circuit Board inspection (PCB). While none of the systems addressed the same problem with variable data, many of the system components are applicable to our situation and gave us insight into how to solve our specific issues.

For defect detection, most visual inspection systems fall into one of three categories depending on the defect detection approach: image reference (or Template-Matching), design-rule, or some combination (hybrid approach) of both[3,4,5,6]. In the simplest image reference approach, a reference exists that allows a direct comparison to potentially defective products. Inspection of 100% of the potentially defective unit is typical for this approach. A more elaborate referential approach involves recognizing features of potentially defective items in test images and comparing those features with a set of idealized or perfect features. Inspection coverage on potentially defective items can vary and does not need to be 100%. In the design-rule approach, a set of rules that describe properties of products exist which can be statistically verified for a product. In this case, as little as 10% of a product can be inspected before generating the appropriate statistics and determining whether a defect exists. In general, the image reference approach is a much more reliable approach.

Table 3, Table 4, and Table 5 show a representative sample of inspection systems for print, textile, printed circuit board, and texture inspection. Each row represents an implemented and tested system that has been reported in the literature. The 2nd column shows the image data type worked on by the defect detection algorithm, i.e. color, grey-scale, binary, etc. The next column demonstrates whether the algorithms are *defect specific* or general purpose, i.e. able to detect all defects without knowledge of specific types of defects. The *Detection Method* column specifies whether the approach falls into one of the three main categories: Template-Matching, Rule-Based, or Hybrid. The *% Inspected* column indicates the area inspected by the visual system. If the defect detection and false alarms rates are reported, they are included in the next two columns and a description of the main technology used to accelerate computational processing is listed in the last column.

As an example, automatic visual inspection is very common in the textile industry. While textile inspection and inspection of printed products share some characteristics, the main difference is that textiles generally contain repeated patterns which can be characterized and statistically analyzed. In addition, defects in textiles fall into well-defined categories[7] which make it less difficult to develop defect-specific detection algorithms. A recent survey of fabric defect detection looked only at uniform textured materials (non-printed textures) and focused on the difficulty and myriad approaches to finding differences even when there are not random or patterned textures[8].

There are some interesting results to be seen from these tables. First of all most of the Offset Print systems use a template-matching approach while the textile and general Web-Inspection systems tend to use the Rules-Based approach. The exception is Lace Inspection. Lace typically contains more complex patterns and defects, so the Rules based approach is probably more difficult in terms of discovering and generating the appropriate statistics. PCB inspection systems tend to favor template-matching for inspection of patterned wafers. These patterns can be repetitive, for example, memory cells, or more random patterns for logic circuits[6]. In the latter case, a template-matching scheme makes more sense because the logic patterns on a printed circuit board are usually complex.

Many different algorithms are used in the defect detection component of each system. The most popular appear to be morphology for noise reduction or defect enhancement and thresholding based on experiments or some other a priori knowledge about the system. The defect detection rates reported are generally good (somewhere between 80% and 100%). The false alarm rate is seldom reported, making it difficult to draw conclusions on the effectiveness of these systems. Those papers that did report a false alarm rate, reported relatively small numbers, but did not indicate how the rate was calculated and whether it was within specification or tolerance levels for the system.

Table 6 is included to demonstrate the types of systems that are in development, but not yet demonstrated on real systems. The table contains a series of papers that describe algorithms designed specifically for inspection purposes and is similar to the preceding tables on complete systems. From the table we can see similar patterns of detection methods for inspection item: most of the textile inspection algorithms are rule-based and depend on statistical properties of repeated patterns.

Our system, as designed, uses a template-matching approach for both general purpose and defect specific defect detection. In comparing our solution to existing systems, ours is most similar to offset print or lace inspection systems, although none of them deal with variable data. Finally, many of the systems we looked at operate off-line and therefore do not have the strict real-time requirements as our VDP inspection system.

System Advantages

The two most important benefits of adding an automatic inspection system for defects to a variable data printing press include a decrease in the amount of wasted consumables and allowing a single operator to monitor more than one press at a time.

In addition to these benefits, there are other benefits to using an automated defect detection system. Tests have shown that manual (human) inspection is about 80% effective[9], and only when the inspection process is highly structured and repeatable. Depending on the customer requirements, 100% visual inspection may be necessary. For example, the pharmaceutical industry has a requirement of 100% inspection rate for all medicinal labels, as an error in labeling could have fatal consequences.

System Solution

Our solution is to use hardware and software that automatically analyzes prints and alerts the press operator when a defect is detected. Our solution compares the digital image of the job to each print at line speed. A line scanner, designed to handle 320mm x 464mm sheets, is placed at the output of the press for fast acquisition of an image of each print. This scanner has been designed using off-the-shelf technologies. There are six main components to the detection system:

- an off-the-shelf line scanner for acquiring an image of each printed sheet as it comes off the press
- acquisition of reference and scanned images
- alignment of the reference image to the image of the scanned printed image
- defect detection
- image processing hardware and software for real-time processing of image data
- decision function and operator alert in the event a defect is found

The process is as follows: A printer raster-generating device receives the digital job at a resolution of 812.8 DPI (dots per inch or 320 dots/cm) for imaging onto a photo imaging plate (PIP). At the same time a series of downscaled image files containing CMYK color separations are sent to the image processing module of the defect detection system for conversion to RGB image format. Meanwhile, the image is sent to press for printing. Once the print exits the press, an on-line scanner captures the entire print and sends the image data through a specialized image processing board and then onto the main defect detection system where it is registered with the reference image.

Figure 1 contains a block diagram of the system software. The digital image used for the print job is input into the defect detection system concurrent with the actual print job. As the print comes off the press it is scanned and the scanned image is sent to the registration module of the print defect detection system. Once registered, the defect detection algorithms are applied to the images and the results are sent to a decision function that determines the probability that a defect exists. Finally, the results can be forwarded to defect diagnostics, which will help in determining what caused the defect.

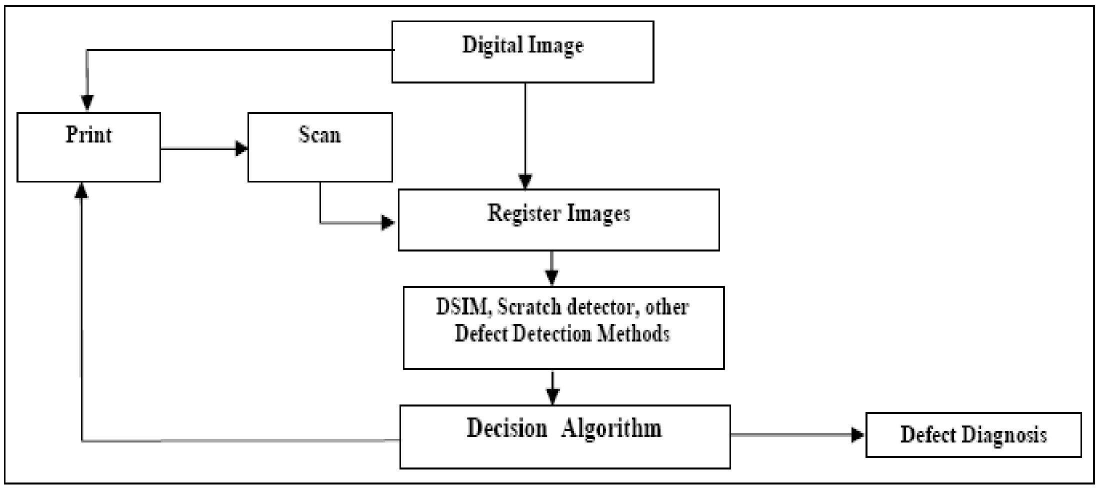

Figure 1 - Print Defect Detection Solution-Software

Image Acquisition

An important piece of the automated VDP inspection system is image acquisition and there are several requirements any solution investigated must meet. A major requirement for the inspection system is that additional costs should be kept to a minimum. In addition, acquisition must be fast enough to capture a print at press-speed at a resolution that does not obscure defects. Finally, the system must be able to capture 100% of the print in both the process (paper travel) and cross-process directions.

During the investigation phase, we determined that a line scanner was the most economical solution. A prototype scanner was built using a scan unit from HP's All-In-One product in an effort to keep down costs. This scanner has a scan width of 310mm which almost covers the entire print format. The line CCD maximal resolution is 600 DPI, however, to handle a 320mm x 464mm each second, scanning resolution is limited to 52 DPI in paper travel direction.

Digital Representation of the Reference Image

The automated defect detection system acquires a reference image for comparison to the print going to the press. Each print is potentially different; therefore a reference image must be generated for each. We compare raster images to avoid layout/format related problems. The simplest and most efficient approach is to use the same image sent to the mechanism that actually prints the job. This alleviates the need to deal with different formats such as PDF.

The raster image processor is responsible for generating the print images from files which are subsequently used by the writing head on the press. Typically this is an 812 DPI image; however, the press software also has the ability to generate downscaled versions of the image. Our algorithms, from registration to defect detection rely on 150 x 52 DPI images for fast processing.   The software can be configured to generate the downscaled images which are stored on disk. Each print consists of four color separations and therefore a separate file for each of cyan, magenta, yellow, and black is generated. The inspection software then picks up those files, performs a simple CMYK to RGB conversion and generates a reference image for use in the rest of the inspection system. This is an asynchronous process that depends only on the production of the four color separation files.

Image Registration

Once a reference image is available and the potentially defective image is acquired, the next step in detecting defects is to register the reference image to the potentially defective image. Our registration algorithm uses a modified form of the dual-Gaussian fitting thresholding technique originally proposed by Kittler and Illingworth [10] to create binary maps while allowing us to quickly generate regions in the image. Regions are connected components generated from the map of the sub-threshold (black) pixels (after smearing to eliminate the effects of specular noise). Solid regions are larger (>1 square inch) regions typical of photos, tables, and graphics; while non-solid regions are the smaller regions typical of text. The edges of the solid regions, together with the centroids of the non-solid regions, allow us to de-skew and align images very quickly, a critical requirement of the entire system.

Rather than finding the skew of both images, we find the differential skew. Skew values are determined by finding the angles along the edges of solid regions and between the centroids of the non-solid regions for both the reference and defective images. By looking at all region combinations (i,j) from 0 to the number of non-solid regions, we find the angle between the centroids of the regions:

```
current_angle = atan( ydelta / xdelta );
//Note: atan 0/x, x/0, or 0/0 cases handled separately
current_angle *= inverse_of_angle_increment; //increments per degree = 5
                                             // or 0.2 accuracy per degree
current_index = 225 + RoundedValue ( current_angle );
```

225 above is 45 x 5 because we used angle accuracy to the nearest 0.2 degrees. 225 is the index for 0 degrees (indexes go from 0 to 450, where index 0 represents -45 degrees and index 450 is +45 degrees.) Every 0.2 degrees in the range [-45, 45] degrees is checked.

Skew values for the solid regions are added to those for the non-solid regions. Instead of using the centroids, however, each boundary pixel for the solid regions is used for angle calculation.

The two arrays of 451 angle increments for the reference and potentially defective image are then compared by finding the maximum number of matched bins (which corresponds to the best correlation) between the two angle histograms, denoted pAngles1 and pAngles2 in the following code fragment.

```
      for( n = 0; n< nAngles; n++)
      {
         marker = n - midpoint;
         for( p = marker; p < (marker+nAngles); p++)
         {
           if( (p>=0) && (p<nAngles) && ( (p-marker)>=0) && ((p-
      marker)<nAngles) )
           {
              min = pAngles1[p-marker];
              if( pAngles2[p] < min )
                min = pAngles2[p];
              matchedBins[n] += min;
           }
         }
      }
```

The maximum value of matchedBins is offset from -225 to 225 (we assume they are within 45 degrees of each other) from the midpoint (225). From it we can directly calculate the differential skew.

For example, consider the following two histograms:

```
pAngles1[0,0,4,5,2,3,0]

pAngles2[3,6,1,3,1,0,0]
```

When pAngles2 is offset from pAngles1 by -3 pixels we get matched bins from the following alignment:

```
pAngles1[0,0,0,0,0,4,5,2,3,0]
pAngles2[3,6,1,3,1,0,0,0,0,0]
```

In this case, there are no matched bins as there is a 0 at each position between them. However, when pAngles2 is offset from pAngles1 by -2 pixels, we get the following alignment:

```
pAngles1[0,0,0,0,4,5,2,3,0]
pAngles2[3,6,1,3,1,0,0,0,0]
```

Now we see that in the sixth position we have matched bins and the sum is 1. Continuing in this manner for -1, 0,+1, +2, and +3 pixels gives the sums 3+1=4, 1+3+1=5, 4+1+2+1=9, 3+5+1+3=12, and 3+2+1=6. Therefore, when pAngles2 is offset by +2 pixels, we get the best alignment (matched bins = 12).

$$\underset{\substack{\text{shift } S \\ |S| < \text{length of array}}}{\text{argmax}} \left[\sum_{0}^{allbins} \{ pAngles_1(bin), pAngles_2(bin+S) \} \right]$$

Equation 1: Bin Alignment

Each step is done for all of the color planes and this allows us to more precisely define the angle difference. If the differential skew angle is nonzero, we rotate the second (defective) image to match the first image using standard 3-shear rotation. Both images are then aligned in the x and y directions using the same algorithm above for determining the maximum matched bins. Four aligned and de-skewed images are created for use by the defect detection algorithms. These are the R, G, and B grayscale images together with the intensity channel.

The image registration algorithm also generates several error maps that we may be able to use in the final decision function. Intensity and color difference maps are created by subtracting the fully-aligned defective images from the reference images. Hue approximates are used rather than raw color channels.

The hue approximates are the subtractive colors of R-G, G-B, and B-R for sRGB. Using these subtractive colors allows us to find defects of any hue and are less targeted to the RGB primaries. The pixels in the defective image are then back-projected to the reference image and a search is done in a local neighborhood of the back-projected point for the minimum difference in intensity, R-G, G-B, or B-R. The benefits realized by this include:

- The number of false alarms are reduced as slight mis-registrations are overlooked
- Only defects which stand out relative to their near surroundings are targeted
- The effects of Grey-level differences in edges are largely overcome

Finally, we add the differences in intensity, R-G, G-B, and B-R to the same defect map and then threshold the summed defects. A binary map and a file containing defect regions as XML class objects are created for use by downstream defect detection systems to disambiguate real defects from false alarms.

Defect Detection

Given a reference image, the problem of detecting defects becomes a problem of finding differences between the two images. A naive approach is to subtract one image from the other. However, a couple of factors make this approach impractical. First, the scanner may introduce artifactual differences between the images. Second, registration is frequently imperfect and can introduce errors.

One approach we use for our experiments is based loosely on the Structural Similarity Information Measure (SSIM[11]). This general-purpose measure of image quality takes into account the Human Visual System (HVS). An additional, complementary detection method looks for specific defects, namely scratches, which have been characterized as common on the press. Scratches are sometimes difficult to detect with general purpose detection methods because of the typically low-contrast nature of these types of defects.

Once the reference and potentially defective images are registered and pre-processed (e.g. noise removal, blurring, etc.), defect detection is performed using the two images. The subsections below describe two algorithms we currently use: structural dis-similarity and the scratch detector.

Structural Dis-similarity

The structural dis-similarity measure is based on the idea that every region in the print image should have a similar region nearby in the reference image, unless it contains a defect. The most prevalent similarity measures, such as difference or sum squared error, are easy to understand and use, but they do not correspond well to perceived visual quality[12,13]. Our defect detection algorithm uses ideas from SSIM [11] which assigns a similarity value to two images (Equation 2). SSIM has three components: a luminance measure (Equation 3), which compares the mean values of two regions; a contrast measure (Equation 4), which compares the standard deviation of the two regions, and a structural measure, which compares the correlation of the regions (Equation 5).

$$SSIM(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma}$$

Equation 2

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

Equation 3

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

Equation 4

$$s(x,y) = \frac{2\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

Equation 5

Where:

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$$

Equation 6

Our algorithm performs an additional local registration for each pixel by looking for the best match between images using the SSIM measure. Searching for the best local match is related to techniques from optical flow[14]. Once we have the best match between the digital reference and scanned image, we then measure the dis-similarity between the images. Our measure, Structural Dis-Similarity Index Measure (DSIM) uses only the contrast and structure measures to determine if two pixels are sufficiently different from each other to signal a defect. During experimentation we discovered that while luminance is helpful for finding the closest pixel match, it is actually a hindrance when trying to determine whether a defect is present or not. Ignoring mean differences when looking for true defects reduces luminance fluctuations that may show up as defects but were actually introduced during the scan process. The basic algorithm is as follows:

```
For each pixel p in input image:
      k x k neighborhood: x = x(p), centered at p:
            1. Find best matching k x k pixel neighborhood y =
               y(p)in reference image within window of size W x W
            2. Compute DSIM:
```

$$\left|(1-c(x,y))\cdot s(x,y)\right|$$

Equation 7

The potential results of this computation need to be explained. The contrast measure can vary between 0 and 1, with 0 meaning the two points in the images are different and 1 indicating similarity. When a defect is present the pixels containing the defect would have deviation measurements that are different from those on the other image. The structure measure may vary between -1 and 1, with -1 indicating a negative correlation and 1 indicating similarity. Any value close to the zero, whether positive or negative is an important indicator of dis-similarity.

Table 1 shows the expected results of the DSIM algorithm based on the values of the structure and contrast similarity measures. These measures are rarely exactly -1, 0, or 1, but a value between them. It does not seem intuitive that if either the structure or contrast variables contain 1 (meaning they are very similar) and the other variable contains 0; the answer should be “Maybe”. In fact, if a defect exists, we expect both variables to be highly dissimilar, even though they measure different attributes of the image. The most obvious case where this situation might occur is when the defect contrast is low with respect to the background on which it sits. The contrast measure would be similar on both images, but the structure measure should show low correlation between them.

Table 1: Defect Detection Result Map

Structure Val	**Contrast Val**	**(1-Contrast)**	**|(1-Contrast)*Structure|**	**Defect?**
-1	1	0	0	Maybe
-1	0	1	1	Yes
0	1	0	0	Maybe
0	0	1	0	Maybe
1	1	0	0	No
1	0	1	1	Maybe

The result of applying this measure is a kind of difference image. This image is then thresholded to obtain a binary error map. The structural dissimilarity measure is particularly adapted to tune out noise and respond to mismatched edges. For example, Figure 2, part (a) illustrates a typical print in RGB colorspace. Figure 2, part (b) contains the portion of the print containing a defect in the green channel. Figure 2, part (c) shows the error map generated by DSIM for the error. The small defect can clearly be seen and easily detected as a defect.

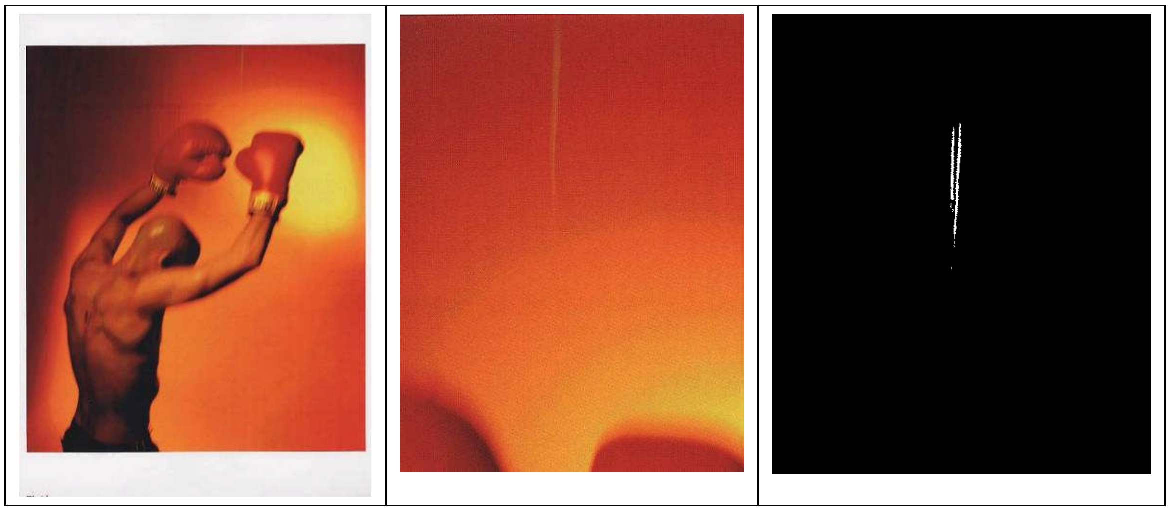

Figure 2: DSIM-- (a) Defective Image; (b) Enlarged image of defect; (c) DSIM Results

Sparse Projection-Based Scratch Detector (SPSD)

DSIM is a general purpose defect detector in that defects of all shapes and sizes are caught by it without any defect-specific knowledge of the defects themselves. As we will see in the Experimental Set-up & Results section, the DSIM algorithm missed 27% of the defects present in 454 prints. The unique characteristics of the majority of the missed defects, called scratches, suggest using a dedicated scratch detector that performs better than a general-purpose defect detector. The majority of these scratch defects have several characteristics in common. They are usually very thin and they may have very light contrast with the surrounding background, which can be textured or noisy. In other words, these defects have a very low signal-to-noise ratio. They are similar to image features such as line edges and they may appear across the entire page or as a local segment. In order to overcome these difficulties, we use the directional coherence of these scratches and the fact that the direction of the scratch is usually known in advance. The main idea of our approach is to improve contrast by projecting (summing) pixels along the scratch.

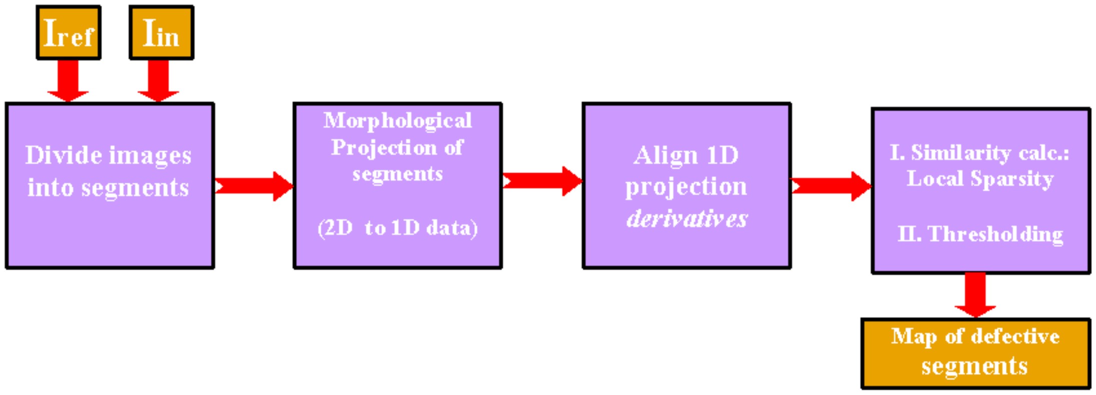
Iref
Iin
Divide images into segments
Morphological Projection of segments
(2D to 1D data)
Align 1D projection derivatives
I. Similarity calc.: Local Sparsity
II. Thresholding
Map of defective segments

Figure 3: Block diagram of scratch detection method

Figure 3 contains a block diagram of the scratch detection algorithm. The input to the algorithm consists of the reference and potentially defective images. The first block of the algorithm divides reference and tested images into overlapping segments. The segment size can either be fixed or, it can be chosen adaptively. The second block is the morphological projection operation; in order to improve delectability of low contrast defects we use morphological operations revealing local maxima (minima) in reference and tested images. Then, each segment of the maps of local maxima is projected (summed) in a particular direction. This effectively represents a 2 dimensional to 1 dimensional data transformation. An alternative, more computationally efficient way to perform morphological projection is to first perform summation, and then apply a 1 dimensional morphological operation that finds local maxima. The third block of the algorithm takes 1 dimensional derivatives of the projected segment data, and aligns the reference and tested projection derivatives. The fourth block calculates a similarity measure between the reference and tested projection derivatives. This similarity measure is sparsity related and reflects the difference in the number of spikes in the corresponding segments. In our experiments we used kurtosis as a sparsity measure. Finally, the similarity measure is subject to a thresholding operation. The threshold value is proportional to the reference segment activity. This effectively reduces false detection of intrinsic image features such as line segments and various edges. Segments wherein the difference is greater than a predefined threshold are marked as defective.

A map of defective segments is generated as output from the process. Figure 4 shows an example of the error map output by the scratch detector. The highlighted blocks show where defects were found in the image.

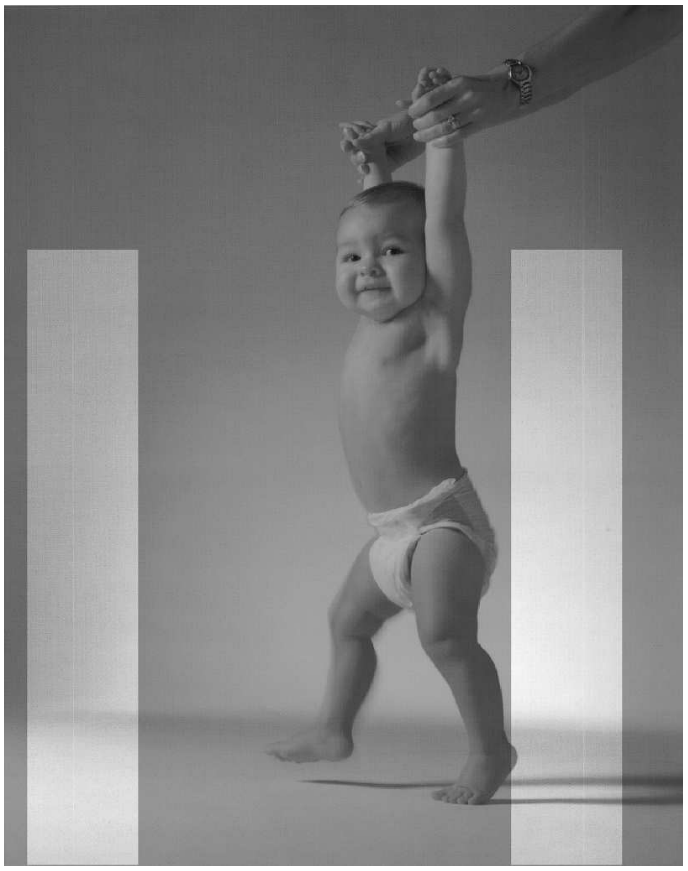

Figure 4: Results--Error map of Scratch Detection method

Real-time defect detection

The defect detection system operates as part of a variable data printing system and therefore needs to operate in real-time. This requirement translates to a processing rate of at least one page per second in current industrial printing systems. We focused on the DSIM computation as this is the most expensive part of the system. The algorithm compares block-similarity and searches for the best matching block. This operation is time consuming since it requires a window of pixels for every pixel in the image to be compared to the corresponding window in the reference image. The key observation in accelerating DSIM is that it is a massively parallel algorithm. Per-pixel decisions only depend on a small number of nearby pixels, and the computation is order independent. Moreover, the algorithm is compute-intensive and not memory bounded. These characteristics make DSIM a perfect candidate for acceleration on Graphics Processing Units (GPUs). Implementing computationally intensive algorithms on GPUs is a trend that becomes more and more prevalent, and this is a good example where the characteristics of an algorithm and a computing environment match.

We implemented the algorithm using the CUDA[15] computing interface from Nvidia. CUDA gives the programmer low level access to the massive computational capacity of the GPU. The implementation follows closely after the algorithm, and achieves parallelism that is dictated by the number of processing units in the GPU. To reduce memory access each image is stored in texture data storage which is cached and optimized for a two-dimensional memory access pattern. Additionally, DSIM for each pixel is computed for every color-channel independently. This significantly improves the memory access time since each pixel is only accessed once. After the R, G and B per-channel DSIM values are computed their values are combined with a logical OR operator to yield the final DSIM decision.

The code was tested on Geforce 8800 GTX and Quadro FX 3700 graphics cards. It is able to achieve 3.75 Mpixel/second data rate. When compared to a serial C implementation that was not optimized and tuned, we see a significant improvement of up to an order of magnitude.

Decision Function

A critical part of the system involves notifying the operator that a defect has occurred and the seriousness of it (press-stopping or not). The decision function takes results generated by the detection algorithm and uses it as input to the decision function. Because the defect map is binary, morphological filtering is applied efficiently to reduce the noise and remove visually imperceptible defects. Morphological filtering is commonly used for reducing noise without destroying important information in the image.

We use a 3x3 square Gaussian filter as the structure element for both the open and close morphological operations. Once the image is filtered, we do a simple projection in both the horizontal and vertical to determine if real defects occur on the page. In our experimental results, if more than two pixels are “on” in a single row or column, we trigger a defect warning. Size and shape of the defect can subsequently be used to determine whether the defect warrants an automatic shut-down of the machine or a warning to the operator without affecting the current print job. Additionally, the location of the defect can be quickly determined from the results of the morphological operations. This is helpful in a closed-loop system that automatically diagnoses the printer problem.

Experimental Set-up and Results

We conducted a large scale experiment to test our detection algorithms. Our experiment consisted of 454 scanned images of 320mm x 464mm size prints from the HP Indigo press. While most contain defects, 22 randomly placed images did not and were included to ensure the algorithm would not find defects on images that contained no defects (i.e. false alarms). Reference images were generated by scanning good prints on the same scanner used for scanning defective prints. Using a printed reference avoids technical difficulties related to the format of the digital data. At the same time, technical challenges related to comparing images, such as registration, are retained.

During registration we find the differential skew, as this decreases the chances for error. This claim is based on a test of 40 differently-skewed pairs of files for which the skew is considered difficult to determine. The test showed that when using differential skew, the correct skew value was determined correctly for 92.5% of the difficult files. When using absolute skew, the skew on files with one skewed image was incorrectly identified for 15% of the cases. Difficult files where skew angle errors differed relatively between two files resulted in a 20% rate of incorrectly identified skew values.

As part of the registration, a set of four images are generated for both the reference and the potentially defective images, corresponding to the red, blue, green, and intensity planes of the image. The DSIM algorithm is then run on each of the color planes separately. The resulting error images are OR'ed together to obtain a final error map. During analysis we found that all four color planes were necessary because different defects show up in different color planes due to the highly variable nature of the prints themselves.

Table 2 summarizes the detection results of the DSIM algorithm on 454 samples prints containing 1653 defects. The table shows that the total number of defects found in 454 prints was 1193. This represents about 73% of the total number of defects. Most of the defects that were missed were low-contrast defects. Low-contrast defects are those defects with intensity values that are very close to the intensity values of the background on which it sits. Many of the low-contrast defects missed are defined as “scratches” and were found by the sparse projection-based scratch detector.

For DSIM, the rate of false alarms is low at 1%. It should also be noted that most of the false alarms were of a single type that were traced to a problem with the scanner used to scan in the images. The Sparse Projection-Based Scratch Detector found the majority of the defects missed by DSIM.

Type	Total Defects	Total Found	Total Missed	Detected	False Alarms
DSIM	1653	1193	460	72%	1%
Scratch Detector	325	217	108	67%	0.036%
Union				86%	

Table 2: DSIM & Scratch Detector Results

While this large-scale experiment is a good test of the algorithms developed so far, it should be kept in mind that it is not a substitute for a test of the system. Obviously, a test of the system would include the prototype scanner and some way of determining the severity of the defect.

Recently, we have implemented the system using an HP Indigo press with an inline scanner attached to a bridge at the output of the press. This was a preliminary test used to determine the feasibility of the algorithms in a real environment. While mechanical issues prevented us from running a large-scale experiment, we were able to confirm that the software components worked well and runtime was very close to that required. The following example illustrates the results we typically observed.

Figure 5 shows a cropped version of both a reference (left) and defective (right) image. The defect is a very small white spot within the letter “i” in the word “Action!”. The defect can be seen in Figure 6 as the small whitish area in the right part of the dot.

Figure 7 shows both the contrast and structure measures for this portion of the image. In these images, higher values (white) indicate higher similarity between the images. The contrast map (right) shows the defect quite clearly. It is harder to see the defect on the structure map; however, the values for the defective pixels are lower than those surrounding them.

Figure 8 shows the entire defect map for the image. We use this map to determine whether a defect exists or not. The small defect on the dot of the “i’’ shows up near the middle and right of the figure. The figure also shows that there are many other differences between the images. The most obvious are the edges of the images on the left and bottom. This is due to a slight scaling issue between the defective and reference images. We use the decision function to remove this type of false alarm and noise to clean up the defect map for reporting true defects.

Figure 9 shows the results of applying morphology on the defect map of Figure 8. The figure is zoomed to show the portion of the image containing the defect. The real defect was found and additionally, a line at the very bottom of the image. This is a real artifact caused by a situation in which the scanned image is actually shorter than the scan bed. It should also be noted, that no false alarms occurred on this image.

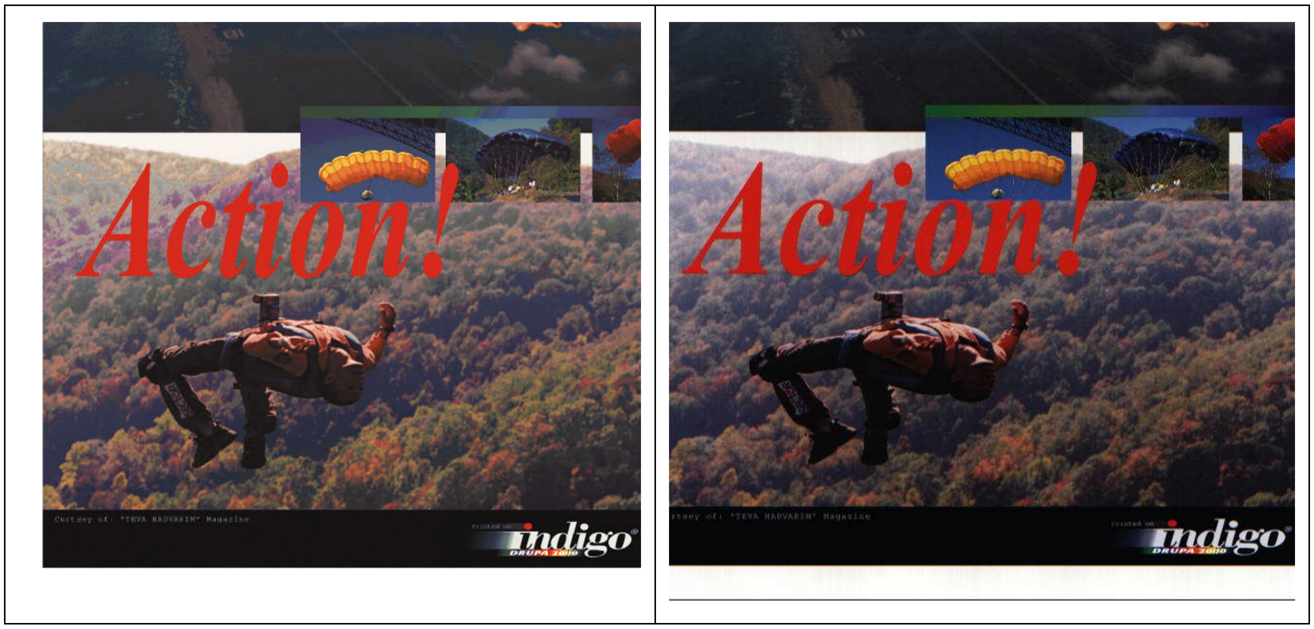

Figure 5: Images-- (a) Reference Image; (b) Defective Image

Figure 6: Zoomed Defect on dot in "i" in Action Word

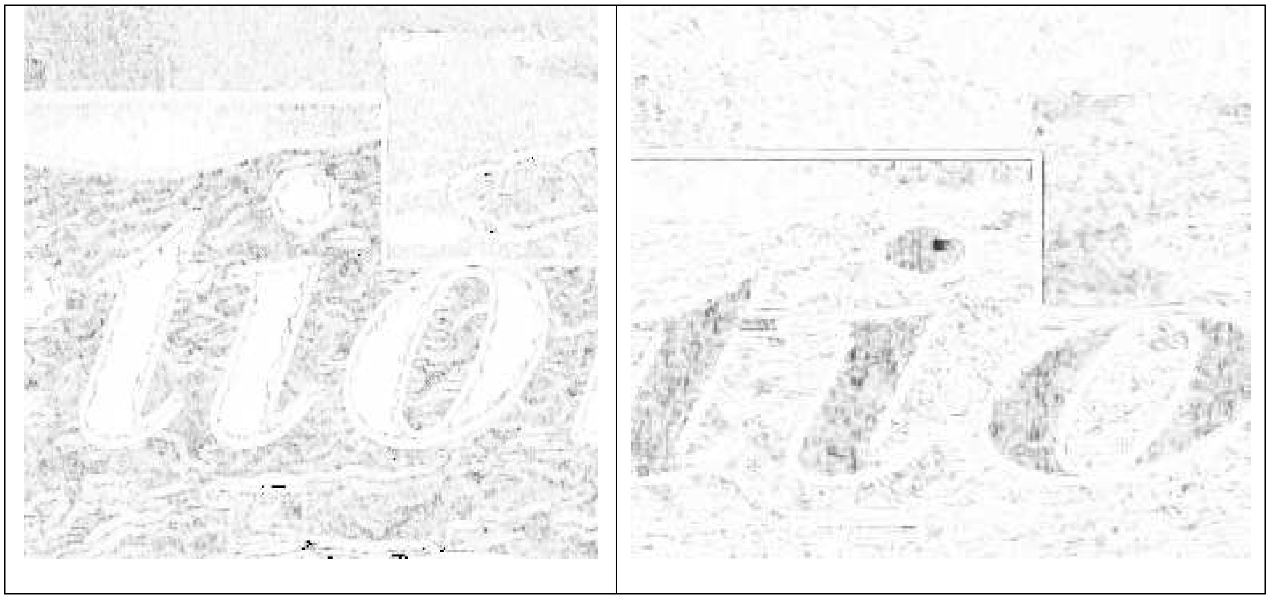
Figure 7: Similarity Measures: a). Structure, b). Contrast

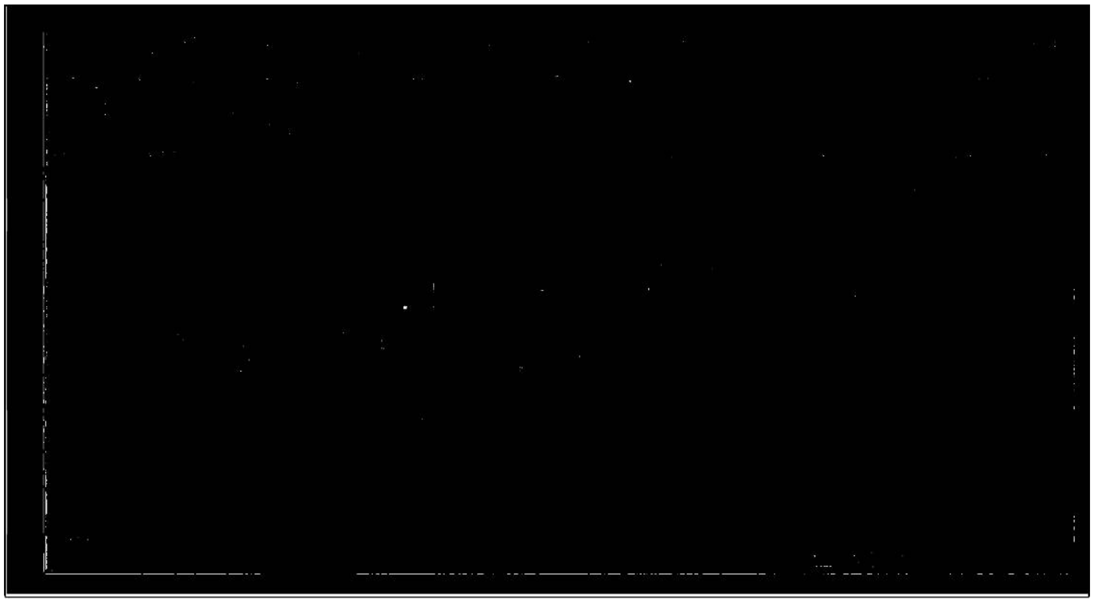

Figure 8: Results DSIM - Whole Image

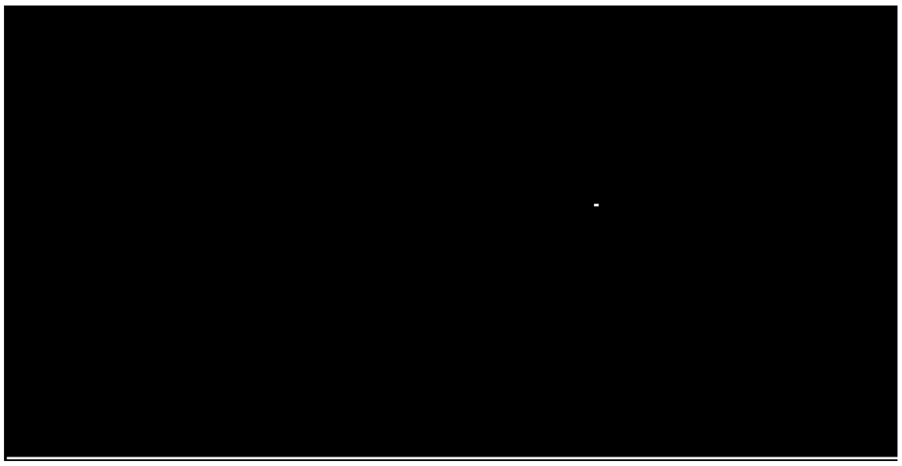

Figure 9: Defect Map after Morphology: Right Half

We were able to run enough samples to show feasibility. Interestingly, most of the software worked very well and timing was very close. The most valuable information we discovered during these tests were new technical challenges that need to be addressed in building a fully functional system. One problem involved the prototype paper handling system which caused the paper to frequently jam prior to going under the scan head. This resulted in severe skewing problems from which our registration algorithm was unable to recover. There are also various parameters which are user-settable on the press, for example scale, which are not necessarily communicated to the detection algorithm. This causes problems with registration later because the scale of one image may be very different from the other. This test was instrumental in developing our path forward as we discovered issues that could not be predetermined using simulations.

Conclusions and Future Work

We have described an on-line, automatic defect detection system for VDP prints. We have shown work-to-date and demonstrated that our algorithms consistently detect a variety of defects with a very low false alarm rate. We carefully analyzed a set of results from an experiment using the DSIM and scratch detection algorithms on 454 images. We have also successfully demonstrated feasibility by testing the system on an HP Indigo commercial press.

In the future there are a few important issues that need to be addressed: .

- Tune existing algorithms for best performance in terms of defects identified and low false alarms.
- Develop additional detection algorithms for low-contrast, defects which are still not easily detected.
- Compensation for potential mismatch between digital and printed images:
    - Stretching/Skew from belts, paper, blanket, writing head
    - Vibration on the press
    - Dot Gain
    - Illumination and reflection
- A large-scale experiment using a working commercial press needs to be run in order to determine additional issues that should be resolved before a fully functional defect detection system can be complete.

Acknowledgements

We would gratefully like to acknowledge the help of Avi Malki of HP Indigo in making the testing of our latest version of the system functional. Without his unflagging support and hard work, the machine test would not have been possible. In addition, we would like thank Rodolfo Jodra of Hewlett-Packard Company for his effort in bringing the scanner to fruition. Many other people at HP Indigo worked on the hardware aspects of the system and we would like to thank them as well.

Appendix A – Tables

Table 3: Inspection Systems

System	Image Type	Defects Detected	Detection Method	% Inspected	% Detection Rate	False Alarm Rate	Real Time
Offset Print [16]	4 color	All	Template Match	100%	NR*	NR*	NR*
Offset Print [17]	grey-level map	All	Template Match, Image Subtraction	100%	NR	NR	On-Line, Image Proc. boards using SIMD
Offset Print[18]	RGB converted to CIELAB	dotgain, density, registration	Template Match	Patches	NR	NR	OffLine
Offset Print[19]	grey-level map	All	Template Match Subtraction	100%	NR	NR	CPU with INTEL IPP
Offset Print[20]	grey-level map	All	Template Match Subtraction	100%	NR	NR	NR
Printing[21]	luminance channel	All	Template Match, subtraction, adaptive thresholds	< 100%	95%	0.5%	OnLine
Web-Inspect, General [22]	grey-level map	All	Rule-Based, Auto-regression, multi-level thresholding, defect-free reference	100%	NR	NR	FPGA Video Processor for data acquire
Web-Inspect, Paper[23]	grey-level map	All	Rule-Based, feature detect& segmentation, self-organizing map (SOM)	100%	NR	NR	algorithms implemented in pipelined architecture using VHDL
Textile [24]	grey-level map	All	Rule-based, match with learned characteristics of defect free samples	100%	100%	0.0	OffLine
Textile (Wool)[25]	RGB + Human-Visual Color space	All	Rule based, adaptive thresholds using defect free samples, feature extraction	100%	90%	NR	video acquire & processing board
Textile [26]	grey-level & binary	All	Rule based, adaptive thresholds using defect-free samples, feature extraction	100%	~80%	NR	video acquire & processing board
Textile [27]	grey-level & binary	All	Rule-based,adaptive thresholds(2) using defect-free samples, noise filtering	100%	91%	NR	DSP & FPGA boards
Textile (Lace) [28]	grey-level & binary	All	Template Match,user-settable thresholds, Morphological filtering	100%	NR	NR	OffLine, Algorithmic speed-up

* NR – Not Reported in publication.

Table 4: Inspection Systems - Continued

System	Image Type	Defects Detected	Detection Method	% Inspected	% Detection Rate	False Alarm Rate	Real Time
Textile [29]	Wavelet transform, grey-level, & binary	All	Rule-Based, search for disruptions in patterns, attenuate background, accentuate defects	100%	89%	2.5%	DSP image acquire, dual frame processing
Textile(Pattern) [30]	Grey-level & binary	All	Template Match, adaptive thresholding (2) based on statistical parms of ref image, erosion	100%	86.2%	4.3%	DSP board, FPGA
Textile(plain weave)[31]	Grey-level & binary	All	Hybrid,Image preprocessing (noise reduce,), local variance, median filter, adaptive threshold	100%	91%	7%	Frame grabber, 2 CPUs for parellel processing
Texture(wood)[32]	Grey-level & binary	Defect Specific	Rule-Based, Pyramid linking(multi-resolution) for segmentation, trained Bayes classifier	100%	81%	NR	VAX 11-785, NCUBE sys with 8 processors
PCB[33]	Grey-level & binary	Defect Specific	Hybrid, Design data as reference, image subtraction, morphological operations, statistical inference on defect	100%	95%	0.5%	Specialized image processor
PCB[34]	Grey-level & binary	Defect Specific	Template Match, diffs between 2 parts PCBs using 2 types lighting	NR	100%	5%	Algorithms implemented in hardware
PCB[35]	Grey-level & binary	NR	Template Match, Morphology to create difference maps from defective &reference images not precisely aligned, thresholded	NR	95%	5%	FPGA

Table 5: Inspection Systems - Continued

System	Image Type	Defects Detected	Detection Method	% Inspected	% Detection Rate	False Alarm Rate	Real Time
Misc.(Tiles)[36]	4 color channels	Defect Specific	Hybrid, Template match for patterned tiles, texture analysis based on auto-regression for textured. Statistical estimates from good images	All	NR	NR	NR
Misc.(Color CRT)[37]	Y, PB, PR channels	Defect Specific	Rule-based, white conformity measure based on contrast & luminance and compared with results with human subjects	All	NR	NR	NR
Misc.(flat metal)[38]	grey level & binary	All	Rule-based: Segmentation, morphological ops (noise reduce), expert sys for classification	All	NR	NR	MATROX IP board, DSP processor
Misc.(stampings)[39]	grey level	All	Template Match,Image subtraction, preprocess: blur/Smooth by Sobel edge detect, ID by region connect, threshold, clustering	All	NR	2%	NR

Table 6: Inspection Algorithms

Algorithm	Image Type	Detection Method	% Detection Rate	False Alarm Rate
Web-Inspect, Non-woven[40]	luminance channel	Rule-Based, Statistical estimates	NR	NR
Textile [41]	NR	Rule based, Detect Outliers outside regular features	NR	NR
Textile[42]	grey level image	Hybrid, Feature vector and co-variance matrix extracted. Z^2 for each window thresholded for significance level	NR	NR
Textile [43]	grey level image	Rule-Based, Trained NN, truth hand-generated	NR	NR
Textile [44]	grey level image	Rule-Based, Statistical parameters determined from training on samples	100%	0
Textile[45]	grey level image	Rule-Based, Linear NN & NN feature vectors with PCA (Principal Component Analysis) to reduce vector size	NR	NR
Textile, patterned[46]	grey level image	Template Match, image subtraction of wavelet transform, thresholding, noise filtering	97.7%	NR
Textile [7]	grey level image	Rule-Based, feature extraction based Auto-correlation function, SOM classification	NR	NR
Web (Paper)[47]	grey level image	Rule-Based, feature extraction:local binary patterns, feature reduction, SOM classification	88.7% - 99.8%	NR
Texture (Granite)[48]	RGB color and CIELAB transform	Rule-Based, Probability based on color and blob inspection	NR	NR
PCB[49]	binary images	Rule-Based, Segmentation using mathematical morphology	NR	NR
Misc.(Leather)[50]	grey level image	Rule-Based, Edge detection to ID edges, accept/reject edge as defect based on pooled variances of areas around defects, 2 thresholds determined apriori	NR	NR
Misc.(Machined Parts)[51]	grey level image	Rule-Based, feature extraction: Morphological ops, recursive adaptive thresholding algorithm	NR	NR

[1] D. A. Silverstein and J. E. Farrell, ICIP'96, IEEE International Conference on Image Processing, SOMEWHERE, 881--884, (1996).

[2] Timothy S. Newman and Anil K. Jain, Computer Vision and Image Understanding **61**, 231--262 (1995).

[3] Shang-Hong Lai and Ming Fang, Real-Time Imaging **5**, 3--14 (1999).

[4] Madhav Moganti and Fikret Ercal, Computer Vision and Image Understanding **63**, 287--313 (1996).

[5] Roland T. Chin, in *Computer Vision: Theory and Industrial Applications*, edited by Carme Torras, Vol. **1**, p.377--404.

[6] Byron E. Dom and Virginia Brecher, Machine Vision and Applications **8**, 5--19 (1995).

[7] A. S. Tolba and A. N. Abu-Rezeq, Computers In Industry **32**, 319--333 (1997).

[8] Ajay Kumar, IEEE Transactions on Industrial Electronics **55**, 348-363 (2008).

[9] Bill Smith, IEEE Spectrum **30**, 43--47 (1993).

[10] J. Kittler and J. Illingworth, Pattern Recognition **19**, 41--47 (1986.).

[11] Zhou Wang, Alan C. Bovik, Hamid R Sheikh, and Eero P. Simoncelli, IEEE Transactions on Image Processing **13**, 600--612 (2004).

[12] Alan C. Bovik, Zhou Wang, and Ligang Lu, IEEE International Conference on Acoust., Speech, and Signal Processing, Orlando, FL, 3313--3316, (2002), edited by Billene Mercer IEEE, (2003).

[13] Ahmet M. Eskicioglu and Paul S. Fisher, IEEE Transactions on Communications **43**, 2959--2965 (1995).

[14] B.K.P. Horn and B. G. Schunk, Artificial Intelligence **17**, 183--203 (1981).

[15] NVIDIA CUDA Compute Unified Device Architecture. Programming Guide, version 2.0 (2008)

[16] Petra Perner, Machine Vision and Applications **7**, 135--147 (1994).

[17] F. Torres, J. M. Sebastian, L. M. Jimenez, and O. Reinoso, Image and Vision Computing **16**, 947--958 (1998).

[18] Hansjorg Kunzli, Freddy Deppner, Karl Heuberger, and Yufan Jiang, SPIE, Color Imaging: Device-Independent Color, Color Hardcopy, and Graphic Arts III, San Jose, 286--291, (1998).

[19] Hui-Chao Shang, You-Ping Chen, Wen-Yong Yu, and Zu-De Zhou, International Journal of Advanced Manufacturing Technology **33**, 756-765 (2007).

[20] N. G. Shankar, N. Ravi, and Z. W. Zhong, 4th International Conference on Control and Automation, Montreal, Canada, 794-798, (2003).

[21] F. Trucheteta, IEEE International Conference on Industrial Electronics, Control and Instrumentation, Maui, Hawaii, 1882--1887, (1993).

[22] S. Hossain Hajimowlana, Roberto Muscedere, Graham A. Jullien, and James W. Roberts, Real-Time Imaging **5**, 23--34 (1999).

[23] Jukka Iivarinen, Katriina Heikkinen, Juhani Rauhamaa, Petri Vuorimaa, and Ari Visa, International Journal of Pattern Recognition and Artificial Intelligence **14**, 735--755 (2000).

[24] Aura Conci and Claudia Belmiro Proenca, ACM 14th International Conference on Software Engineering and Knowledge Engineering, Ischia, Italy, 707--714, (2002). ACM, New York, (2002).

[25] Liwei Zhang, Abbas Dehghani, Zhenwei Su, Tim King, Barry Greenwood, and Martin Levesley, Real-Time Imaging **11**, 257--269 (2005).

[26] Che-Seung Cho, Byeong-Mook Chung, and Moo-Jin Park, IEEE Transactions on Industrial Electronics **52**, 1073--1079 (2005).

[27] Panagiotis Mitropulos, SPIE-IS\&T, Electronic Imaging, Machine Vision Applications in Industrial Inspection, Machine Vision Applications in Industrial Inspection, VII, San Jose, CA, 59--69, (1999), edited by Kenneth W. Tobin SPIE--The International Society for Optical Engineering, (1999).

[28] H. R. Yazdi and T. G. King, Real-Time Imaging **4**, 317--332 (1998).

[29] Hamed Sari-Sarraf and James S. Goddard, Jr., IEEE Transactions on Industry Applications **35**, 1252--1259 (1999).

[30] Radovan Stojanovic, Panagiotis Mitropulos, Christos Koulamasand, Yorgos Karayiannis, Stavros Kaubias, and George Papadopoulos, Real-Time Imaging **7**, 507--518 (2001).

[31] Ahmed Abouelela, Hazem M. Abbas, Hesham Eldeeb, Abdelmonem A. Wahdan, and Salwad M. Nassar, Pattern Recognition Letters **26**, 1435--1443 (2005).

[32] D. Brzakovic, H. Beck, and N. Sufi, Pattern Recognition **23**, 99--107 (1990).

[33] Haruo Yoda, Yozo Ohuchi, Yuzo Taniguchi, and Masakazu Ejiri, IEEE Transactions on Pattern Analysis and Machine Intelligence **10**, 4--16 (1988).

[34] Yasuhiko Hara, Hideaki Doi, Koichi Karasaki, and Tadashi Iida, IEEE Transactions on Pattern Analysis and Machine Intelligence **10**, 69--78 (1988).

[35] Hiroyuki Onishi, Yasushi Sasa, Kenta Nagai, and Shoji Tatsumi, IECON'02, IEEE 28th International Conference on Industrial Electronics, Control, and Instrumentation, Seville, Spain, 2208--2213, (2002).

[36] G. S. Desoli, S. Fioravanti, R. Fioravanti, and D. Corso, IEEE International Conference on Industrial Electronics, Control, Instrumentation and Automation, Maui, HI, 1871--1876, (1993).

[37] Toshio Asano, Keisuke Kawame, Jun Mochizuki, and Nobuo Fukuhara, IECON'92, IEEE International Conference on Industrial Electronics, Control, Instrumentation and Automation, San Diego, 725--730, (1992).

[38] C. Fernandez, C. Platero, P. Campoy, and R. Aracil, IECON'93, IEEE International Conference on Industrial Electronics, Control, and Instrumentation, Lahaina, Hawaii, 1993, (1854--1859).

[39] Ralf Langenback, Alexander Ohl, Peter Scharf, and Jorg Semmler, SPIE, Machine Vision Applications in Industrial Inspection, IX, San Jose, CA, 9--19, (2001).

[40] D. Brzakovic, N. S. Vujovic, and H. Sari-Sarraf, IEEE International Conference on Robotics and Automation, Albuquerque, NM, 1--8, (1997).

[41] Dmitry Chetverikov, 15th International Conference on Pattern Recognition, Barcelona, Spain, 521--524, (2000).

[42] George Mamic, 15th International Conference on Pattern Recognition, Barcelona, Spain, 767--770, (2000).

[43] Claus Neubauer, 11th International Conference on Pattern Recognition, The Hague, The Netherlands, A688--A691, (1992).

---

[44] Fernand S. Cohen, Zhigang Fan, and Stephane Attali, IEEE Transactions on Pattern Analysis and Machine Intelligence **13**, 803--808 (1991).

[45] Ajay Kumar, Pattern Recognition **36**, 1645--1659 (2003).

[46] Henry Y.T. Ngan, Grantham K.H. Pang, S. P. Yung, and Michael K. Ng, Pattern Recognition **38**, 559--576 (2005).

[47] Topi Maenpaa, Markus Turtinen, and Matti Pietikainen, Real-Time Imaging **9**, 289--296 (2003).

[48] K. Y. Song, J. Kittler, and M. Petrou, Image and Vision Computing **14**, 667-683 (1996).

[49] Seyfullah Halit Oguz and Levent Onural, IEEE International Conference on Robotics and Automation, Sacramento, CA USA, 2696--2701, (1991). IEEE Computer Society Press, (1991).

[50] W. Wen and A. Xia, Pattern Recognition Letters **20**, 315-328 (1999).

[51] Frank Y. Shih and O. Robert Mitchell, IEEE International Conference on Robotics and Automation, Philadelphia, PA, USA, 1764-1766, (1988). IEEE Computer Society Press, (1988).