**Introduction**

The image produced by a video camera pointing at black printing on a white sheet of paper is not truly black and white.

白い紙に黒い印刷を向けるビデオカメラによって生成された画像は、真に白黒ではありません。

This image is grey-scale (or colour) no matter where the camera is pointed.

この画像は、カメラの向きに関係なくグレースケール（またはグレー色）です。

Unless lighting on the desk is carefully controlled, video images of paper lying on the desk are poor representations of the originals.

机の照明を慎重に制御しない限り、机の上に横たわる紙のビデオ画像は、オリジナルをうまく表現できません。

Unlike the inside of a scanner or photocopier, it is very difficult to guarantee even lighting across the surface of a desk.

スキャナーやコピー機の内部とは異なり、机の表面全体の照明を保証することは非常に困難です。

This open space may be exposed to desk lamps, overhead lights, windows or moving shadows, all of which can cause varying brightness across the page.

このオープンなスペースは、デスクランプ、オーバーヘッドライト、窓、または動く影にさらされている可能性があります。これらはすべて、ページ全体で明るさが変化する可能性があります。

Human vision compensates for this, but when the image of paper on the desk is manipulated by machine without taking these variations into account the results can be very bad.

人間の視覚はこれを補いますが、机上の紙の画像がこれらの変動を考慮せずに機械によって操作されると、結果は非常に悪くなる可能性があります。

In Figure 5-1, for example, the background of the right hand “in case of emergency” box does not match the surrounding background, leaving a distinct edge.

たとえば、図5-1では、「緊急の場合」の右側のボックスの背景が周囲の背景と一致せず、明確なエッジが残っています。

The box on the right appears lighter than the box on the left, even though it is an exact copy (fold this page if you want to check).

右側のボックスは、正確なコピーであるにもかかわらず、左側のボックスよりも明るく表示されます（確認する場合は、このページを折ります）。

The only difference is in the brightness of the surrounding background.

唯一の違いは、周囲の背景の明るさです。

It gets darker from left to right because the light was on the left side of the paper from which this image was grabbed.

この画像が取得された紙の左側に光があるため、左から右に暗くなります。

This problem is most noticeable when dealing with high contrast line art or text because the original document is genuinely black and white, yet the camera produces an image of varying levels of grey.

この問題は、元のドキュメントが純粋に白黒であるにもかかわらず、コントラストの高いラインアートやテキストを扱うときに最も顕著ですが、カメラはさまざまなレベルのグレーの画像を生成します。

Many applications prototyped on the DigitalDesk must clearly determine which parts of the image are meant to be black or white in order to project line art back onto the desk, or to pass text images to an optical character recognition (OCR) server.

DigitalDeskでプロトタイプ化された多くのアプリケーションは、ラインアートをデスクに投影したり、テキスト画像を光学文字認識（OCR）サーバーに渡すために、画像のどの部分を黒または白にするかを明確に決定する必要があります。

The system cannot use the grey images (typically 8 bits per pixel), so it must convert them to black and white images (one bit per pixel).

システムはグレー画像（通常はピクセルあたり8ビット）を使用できないため、白黒画像（ピクセルあたり1ビット）に変換する必要があります。

There are many ways to do this.

これを行うには多くの方法があります。

In some cases, when the resulting black and white image is meant to be viewed by people, it should look as much as possible like a grey-scale image and the image is dithered or half-toned (using a technique such as [Floy76]) to produce a result like Figure 5-1.

場合によっては、結果として生じる白黒画像が人々によって見られることを意図しているとき、それはグレースケール画像のようにできるだけ見えるはずであり、（[Floy76]などの技術を使用して）図5-1のようなディザリングまたはハーフトーンされた結果を生成します。

Although this image looks like a grey-scale image, it is made up of small black and white spots of varying patterns, densities and sizes.

この画像はグレースケール画像のように見えますが、さまざまなパターン、密度、サイズの小さな黒と白のスポットで構成されています。

Human vision makes sense of these images almost as well as the grey images and the original black and white images.

人間の視覚は、これらの画像とグレー画像および元の白黒画像のほとんどを理解します。

But for machine processing operations, such as character recognition, select & paste operations, and merging together of many different images, the system cannot use dithered or half-toned images.

ただし、文字認識、選択と貼り付け操作、多くの異なる画像のマージなどの機械処理操作では、システムはディザリングまたはハーフトーン化された画像を使用できません。

Although only one bit per pixel, these images are even less suitable for machine processing than the original grey-scale images.

ピクセルあたり1ビットのみですが、これらの画像は、元のグレースケール画像よりも機械処理に適していません。

The system needs simple patterns of lines, characters, and relatively large patches of black and white.

システムには、線、文字、および白黒の比較的大きなパッチの単純なパターンが必要です。

The process by which these types of black and white images are generated from grey images is commonly referred to as thresholding, and it is an important requirement for any digital desk system.

これらのタイプの白黒画像がグレー画像から生成されるプロセスは、一般にしきい値処理と呼ばれ、デジタルデスクシステムの重要な要件です。

There are many ways to threshold an image, but the basic process is to examine each grey pixel and decide whether to make it either black or white.

画像の閾値を設定する方法は多数ありますが、基本的なプロセスは各グレーピクセルを調べて、黒にするか白にするかを決定することです。

This report describes the various techniques that were developed and tested for thresholding on the DigitalDesk, and it ends with the description of an algorithm that was found to be suitable.

このレポートでは、DigitalDeskでのしきい値設定のために開発およびテストされたさまざまな手法について説明し、最後に適切であることが判明したアルゴリズムの説明で終わります。

This algorithm (here called “quick adaptive thresholding”) may not be the best possible, but it works well enough that other problems of the DigitalDesk became more important when the work reached the stage described here.

このアルゴリズム（ここでは「クイックアダプティブスレッショルド」と呼ばれる）は最良の方法ではないかもしれませんが、DigitalDeskの他の問題がここで説明した段階に達したときにさらに重要になるほど十分に機能します。

要約

グレー画像から白黒画像を作成する

画像を2値する基本的なプロセスは, ピクセルごとに白であるか黒であるかを決定すればよい。

**Global Thresholding**

In a way, thresholding can be seen as an extreme form of contrast enhancement, or making light pixels lighter and dark pixels darker.

ある意味では、しきい値処理は極端な形のコントラスト強調、または明るいピクセルをより明るく、暗いピクセルをより暗くすることと見なすことができます。

The simplest (and most common) way to threshold an image is to set all pixels below a certain greylevel to black, and to clear all others to white.

画像をしきい値処理する最も簡単な（そして最も一般的な）方法は、特定のグレーレベルより下のすべてのピクセルを黒に設定し、他のすべてのピクセルを白にクリアすることです。

The question then is how to select this grey-level.

問題は、このグレーレベルを選択する方法です。

One possibility is to pick the centre of the range of possible values, so if pixels are eight bits deep (ranging from 0 to 255), for example, then 128 would be selected.

1つの可能性は、可能な値の範囲の中心を選択することです。そのため、たとえば、ピクセルが8ビットの深さ（0〜255）の場合、128が選択されます。

This approach works well if all the “dark” pixels of the image really do have values under 128, and light pixels have values above 128, but if the image is over or under exposed, then the result might be all white or black.

このアプローチは、画像のすべての「暗い」ピクセルが実際に128未満の値を持ち、明るいピクセルが128を超える値を持つ場合にうまく機能しますが、画像が露出不足または露出不足の場合、結果はすべて白または黒になります。

It is better to look at the range of actual values instead of possible values to determine the threshold.

しきい値を決定するには、可能な値ではなく実際の値の範囲を確認することをお勧めします。

The maximum and minimum values for each pixel in the image can be found, then the midpoint used as the threshold.

画像内の各ピクセルの最大値と最小値を見つけ、その中点をしきい値として使用します。

An even better way to select the threshold is not just to look at the range of actual values, but also their distribution.

しきい値を選択するさらに良い方法は、実際の値の範囲だけでなく、その分布も調べることです。

If, for example, you expect the image to be of black line art, or text on a white background then you expect that most pixels will be the intensity of the background and a smaller but significant proportion will be of the dark ink.

たとえば、画像が黒い線画、または白い背景のテキストであると予想される場合、ほとんどのピクセルは背景の強度であり、小さいがかなりの割合が暗いインクであると予想されます。

A histogram of the pixel intensities should look something like Figure 5-2.

ピクセル強度のヒストグラムは、図5-2のようになります。

A large background peak should be visible, as well as a smaller peak for the dark ink.

暗いインクの小さなピークと同様に、大きなバックグラウンドのピークが見えるはずです。

The whole curve might be shifted to the left or right depending on the ambient light level, but in any case, the best value to pick for thresholding is the local minimum between the two peaks.

周囲光のレベルに応じて、曲線全体が左または右にシフトする場合がありますが、いずれの場合も、しきい値設定のために選択する最適な値は、2つのピーク間の局所的な最小値です。

This is fine in theory, but how well does it work in practice?

これは理論上問題ありませんが、実際にはどの程度うまく機能しますか？

Figure 5-3 shows a frame-grabbed image and its histogram for which the technique works well.

図5-3は、フレームグラブされた画像とそのヒストグラムを示しています。

The smoothed histogram shows two promi- nent peaks, and it would not be difficult to calculate a near-ideal threshold by fitting a curve to this histogram or even taking the midpoint between the two peaks.

平滑化されたヒストグラムは2つの顕著なピークを示しており、このヒストグラムに曲線をあてはめるか、2つのピークの中間点をとることで理想に近いしきい値を計算することは難しくありません。

This is not a typical image, however, because it has large amounts of both black and white.

ただし、これは典型的な画像ではありません。黒と白の両方が大量に含まれているためです。

The DigitalDesk must also be able to threshold images like the one in Figure 5-4.

DigitalDeskは、図5-4のような画像をしきい値処理できる必要もあります。

In the histogram of this image, the smaller “dark” peak is almost lost in the noise, so it is impossible to reliably locate a local minimum between peaks.

この画像のヒストグラムでは、小さな「暗い」ピークがノイズでほとんど失われているため、ピーク間の局所的な最小値を確実に見つけることは不可能です。

In any case, a large (background) peak is always present and easy to find, so a

useful strategy for thresholding an image is the following:

いずれにせよ、大きな（バックグラウンド）ピークが常に存在し、見つけやすいので、画像のしきい値設定に役立つ戦略は次のとおりです。

1) Calculate the histogram (shown by the points in Figure 5-4).

ヒストグラムを計算します（図5-4のポイントで表示）。

2) Locate the large peak by finding the maximum value of a moving average of the histogram curve.

ヒストグラム曲線の移動平均の最大値を見つけて、大きなピークを見つけます。

The moving average smooths out the effects of noise on the maximum value, as shown by the curves in Figure 5-3 and Figure 5-4.

移動平均は、図5-3および図5-4の曲線に示すように、最大​​値に対するノイズの影響を滑らかにします。

3) Select the threshold value at a certain proportion of the distance between this peak and the minimum value (excluding zero counts).

このピークと最小値（ゼロカウントを除く）の間の距離の特定の割合でしきい値を選択します。

Experimentation seems to indicate that 1/2 of this distance produces quite good results in a broad range of lighting conditions ranging from very bright to almost completely dark.

実験により、この距離の1/2は、非常に明るいものからほぼ完全に暗いものまでの広範囲の照明条件で非常に良好な結果が得られることを示しているようです。

In Figure 5-4, for example the peak is at 215 and the lowest recorded value is 75, so a threshold of 145 would be used.

たとえば、図5-4では、ピークは215で、記録された最低値は75であるため、145のしきい値が使用されます。

The figures on the next page (Figure 5-5) show an image grabbed under four different lighting conditions and the result from applying this histogram-based global thresholding algorithm to each.

次のページの図（図5-5）は、4つの異なる照明条件下で取得した画像と、このヒストグラムベースのグローバルしきい値アルゴリズムをそれぞれに適用した結果を示しています。

Despite the wide range of lighting (as can be seen from the histograms) this algorithm selects an appropriate threshold in each case, and the thresholded result for each is almost identical.

広範囲の照明（ヒストグラムからわかる）にもかかわらず、このアルゴリズムはそれぞれの場合に適切なしきい値を選択し、それぞれのしきい値の結果はほぼ同じです。

This histogram-based global thresholding technique works quite well for images that are evenly lit or, as in the examples above, for images that represent very small parts of paper where the light does not vary much.

このヒストグラムベースのグローバルしきい値設定手法は、均等に照明された画像、または上記の例のように、光があまり変化しない紙の非常に小さな部分を表す画像に対して非常にうまく機能します。

But it fails to produce good results for larger areas and in normal office lighting where the range of brightness varies greatly across the desk.

しかし、広いエリアや、デスク全体で明るさの範囲が大きく異なる通常のオフィス照明では、良い結果が得られません。

Because a single global threshold value is used for the entire image, some parts of the image become too white and others too dark. Much of the text then becomes unreadable, as illustrated in Figure 5-6.

単一のグローバルしきい値が画像全体に使用されるため、画像の一部が白くなりすぎ、他の部分が暗くなります。図5-6に示すように、テキストの多くは読めなくなります。

Producing good thresholded images from paper that is unevenly lit requires an adaptive thresholding algorithm.

不均一に照らされた用紙から適切なしきい値の画像を生成するには、適応しきい値アルゴリズムが必要です。

This technique varies the threshold value across the image according to the background illumination at each pixel.

この手法は、各ピクセルの背景照明に応じて画像全体のしきい値を変更します。

The discussion below is illustrated with the results of algorithms applied to the image in Figure 5-6.

以下の説明は、図5-6の画像にアルゴリズムを適用した結果を示しています。

This is a challenging test because the image is lit from the side, and it has black text on a white background (the word “PaperWorks”), white text on a black background (“XEROX”), grey text on a white background (“The best way...”), various shadows, and a thin horizontal black line under the word “PaperWorks.”

これは挑戦的なテストです。なぜなら、画像は側面から照らされており、白い背景に黒いテキスト（「PaperWorks」という単語）、黒い背景に白いテキスト（「XEROX」）、白い背景に灰色のテキスト（「The best way...」）、さまざまな影、「PaperWorks」という言葉の下の細い水平の黒い線が表示されている。

要約

閾値の設定に適した方法は、ヒストグラムのピークと最小値（ゼロカウントを除く）の間の距離の特定の割合でしきい値を選択する方法。実験により、この距離の1/2が非常に明るいものからほぼ完全に暗いものまでの広範囲の照明条件で非常に良好な結果が得られることを示している。すなわち、ピーク215、記録された最低値75の場合、

となり、145の閾値が適用される。このヒストグラムベースのグローバル閾値設定手法は、均等に照明された画像、または上記の例のように、光があまり変化しない紙の非常に小さな部分を表す画像に対して非常にうまく機能するが、広いエリアや、デスク全体で明るさの範囲が大きく異なる通常のオフィス照明では、良い結果が得られない。よって適応的な閾値設定手法が求められる。

**Adaptive Thresholding**

An ideal adaptive thresholding algorithm would produce the same result when applied to an unevenly lit page as a global thresholding algorithm would produce when applied to a　perfectly evenly lit page.

理想的な適応しきい値アルゴリズムは、不均一に点灯したページに適用した場合、グローバルな閾値アルゴリズムが完全に均等に点灯したページに適用した場合と同じ結果になります。

The brightness of each pixel is normalized to compensate for being more or less illuminated, and only then is a decision made as to whether the result should be black or white.

各ピクセルの明るさは、多少照度を補償するために正規化され、その結果のみが結果が黒か白かについての決定が行われます。

The question then, is how to determine the background illumination at each point.

問題は、各ポイントでの背景照度の決定方法です。

One simple way to do this is to grab the image of a blank page before grabbing the page to　be thresholded.

これを行う簡単な方法の1つは、閾値を設定するページを取得する前に、空白ページの画像を取得することです。

This blank page can then be used as a reference image.

この空白のページは、参照画像として使用できます。

For each pixel to be thresholded, the value of the corresponding pixel in the reference　image is subtracted before applying the threshold.

しきい値を設定する各ピクセルについて、しきい値を適用する前に、参照画像内の対応するピクセルの値が差し引かれます。

This method produces very good results as long as the lighting does not change between the time when the reference image and subject image are each grabbed.

この方法は、参照画像と被写体画像がそれぞれ取得される時間の間に照明が変化しない限り、非常に良い結果を生成します。

On a desk, however, the lighting changes frequently as the user's shadow moves across the desk or doors, lamps and other objects are moved about.

ただし、机の上では、ユーザーの影が机やドア、ランプ、その他のオブジェクトを動き回るにつれて、照明が頻繁に変化します。

If there is a window in the room, then the lighting also changes throughout the day.

部屋に窓がある場合、照明も一日中変わります。

One solution might be to immediately precede each grab with a reference grab of a blank page at the same location, but this would be almost as inconvenient to use as a scanner, thus defeating an important reason for using an overhead video camera in the first place.

1つの解決策は、同じ場所で空白ページの参照グラブを各グラブの直前に置くことですが、これはスキャナーとして使用するのとほとんど同じくらい不便であり、したがって最初にオーバーヘッドビデオカメラを使用する重要な理由を無効にします。

Another solution is to try to estimate the background illumination at each pixel by making some assumptions about what the image should look like.

別の解決策は、画像がどのように見えるかについていくつかの仮定を行うことにより、各ピクセルでの背景照明を推定しようとすることです。

We could assume, for example, that the image is mostly background (e.g. white), and that

dark markings make up a smaller part of the image.

たとえば、画像のほとんどが背景（たとえば白）であり、暗いマーキングは、画像の小さい部分を構成します。

Another assumption we could make is that the background lighting changes relatively slowly across the page compared to the changes in light and dark due to marks on page.

別の前提として、ページ上のマークによる明暗の変化と比較して、ページ全体で背景の照明の変化が比較的遅いということがあります。

Many algorithms based on such assumptions are possible.

そのような仮定に基づいた多くのアルゴリズムが可能です。

There is no mathematical theory of adaptive thresholding, and as a consequence, there is also no standard or optimal method for doing it [Prat91] (p. 597).

適応しきい値処理の数学的理論はありません。そのため、それを行うための標準的または最適な方法もありません[Prat91]（p。597）。

Instead, there are several ad hoc methods, some of which are more popular than others.

代わりに、いくつかのアドホックメソッドがあり、その一部は他のメソッドよりも一般的です。

Because the methods are ad hoc, it would be useful to be able to measure their performance.

メソッドはアドホックであるため、パフォーマンスを測定できると便利です。

For this, Haralick and Shapiro [Hara85] propose the following guidelines: regions should be uniform and homogeneous with respect to grey tone; region interiors should be simple and without many small holes; adjacent regions should have significantly different values; and boundaries of each segment should be simple, not ragged, and must be spatially accurate.

このため、ハラリックとシャピロ[Hara85]は次のガイドラインを提案しています。領域はグレートーンに関して均一で均質でなければなりません。リージョン内部はシンプルで、多くの小さな穴がないようにする必要があります。隣接する領域の値は大きく異なる必要があります。また、各セグメントの境界は、不規則ではなく単純でなければならず、空間的に正確でなければなりません。

According to Pratt, no quantitative performance metric has been developed for image thresholding.

プラットによると、画像のしきい値処理のための定量的なパフォーマンスメトリックは開発されていません。

It seems the main way to evaluate an algorithm is simply to look at the result and judge whether it looks right. For images of text, there is a quantitative possibility.

アルゴリズムを評価する主な方法は、結果を見て、それが正しいかどうかを判断することです。テキストの画像の場合、定量的な可能性があります。

The result of various algorithms under various lighting conditions could be fed to an OCR system, and the text produced from this can be compared to the original text.

さまざまな照明条件でのさまざまなアルゴリズムの結果をOCRシステムに送り、これから生成されたテキストを元のテキストと比較できます。

Although potentially useful, this approach was not systematically used for the algorithms described below, because it was judged unlikely to give a different evaluation from the “looks better” criteria.

潜在的には有用ですが、このアプローチは、「より良く見える」基準とは異なる評価を与える可能性が低いと判断されたため、以下で説明するアルゴリズムに体系的に使用されませんでした。

For interactive applications of the DigitalDesk, users must wait for thresholding to finish when completing interaction techniques such as “copy and paste” of text or graphics.

DigitalDeskの対話型アプリケーションの場合、ユーザーは、テキストまたはグラフィックの「コピーアンドペースト」などのインタラクションテクニックを完了するときに、しきい値設定が完了するのを待つ必要があります。

So another important quantitative performance metric is speed.

したがって、もう1つの重要な定量的パフォーマンスメトリックは速度です。

The following sections describe various adaptive thresholding algorithms and show the results they produce.

以下のセクションでは、さまざまな適応しきい値アルゴリズムについて説明し、それらが生成する結果を示します。

要約

二値化処理を評価する主な方法は、テキスト画像に対して処理を施したときの結果を目視で確認することと、処理時間を測ることです。

**Adaptive thresholding based on Wall’s algorithm**

One technique for calculating a threshold that varies across the image according to background illumination was developed by R. J. Wall and is described in [Cast79].

背景照明に応じて画像全体で変化するしきい値を計算するための1つの手法がR. J. Wallによって開発され、[Cast79]で説明されています。

The following algorithm is loosely based on this description. First, it breaks up the image into smaller tiles and calculates a histogram for each tile.

次のアルゴリズムは、この説明に大まかに基づいています。まず、画像を小さなタイルに分割し、各タイルのヒストグラムを計算します。

Based on the peaks of these histograms, a threshold is chosen for each tile.

これらのヒストグラムのピークに基づいて、各タイルのしきい値が選択されます。

Then, every point in the image is assigned a threshold by interpolating between the values chosen for each tile.

次に、各タイルに選択された値の間を補間することにより、画像内のすべてのポイントにしきい値が割り当てられます。

Figure 5-7 was generated from the same grey image as Figure 5-6, but using this technique.

図5-7は、図5-6と同じグレー画像から生成されましたが、この手法を使用しています。

The image was divided into nine tiles (3x3) and for each tile a threshold was selected 20% below the peak.

画像は9つのタイル（3x3）に分割され、タイルごとにピークより20％低いしきい値が選択されました。

From these values, 16 tiles were formed with their corners at the centre of each of the 9 tiles, and the threshold values were interpolated across each of these tiles from the corners.

これらの値から、16個のタイルが形成され、9個のタイルのそれぞれの中央にコーナーがあり、しきい値はコーナーからこれらのタイルのそれぞれにわたって補間されました。

The result is much better than global thresholding, but because it requires more than one pass through the image, it is quite slow. Another problem with this technique is that with some images, the local histograms can be fooled by a large amount of black or white, causing the threshold not to vary smoothly across the image, and the result can be very bad (see in Figure 5-8).

結果は、グローバルなしきい値処理よりもはるかに優れていますが、イメージを複数回通過する必要があるため、非常に遅くなります。この手法のもう1つの問題は、一部の画像では、ローカルヒストグラムが大量の黒または白にだまされる可能性があり、画像全体でしきい値がスムーズに変化せず、結果が非​​常に悪くなる可能性があることです（図5-8を参照）。

Most algorithms in the literature are more complex than Wall’s algorithm, require more passes through the image and would take even longer to run *e.g*. [Hara85, Prat91, Wesz78].

文献のほとんどのアルゴリズムは、Wallのアルゴリズムよりも複雑で、画像をより多く通過する必要があり、実行にさらに時間がかかります。 [Hara85、Prat91、Wesz78]。

It seems possible to implement a much simpler and faster algorithm for adaptive thresholding, so this section describes the steps taken so far in that direction for the DigitalDesk.

適応しきい値処理のためのはるかに単純で高速なアルゴリズムを実装することが可能と思われるため、このセクションでは、DigitalDeskのその方向でこれまでに行った手順について説明します。

The basic idea is to run through the image while calculating a moving average of the last s pixels seen.

基本的な考え方は、最後に見られたsピクセルの移動平均を計算しながら画像を実行することです。

When the value of a pixel is significantly lower than this average it is set to black, otherwise it is left white.

ピクセルの値がこの平均よりも大幅に低い場合は黒に設定され、そうでない場合は白のままになります。

Only one pass through the image is necessary, and the algorithm is simple enough to be implemented in hardware.

イメージを1回通過するだけで済み、アルゴリズムはハードウェアに実装するのに十分なほど単純です。

It is interesting to note the similarities between the following algorithms and the one developed at IBM using analog hardware in 1968 [Bart68].

以下のアルゴリズムと1968年にアナログハードウェアを使用してIBMで開発されたアルゴリズム[Bart68]との類似点に注目することは興味深いです。

Let pn represent the value of a pixel at point n in the image being thresholded.

pnは、閾値処理される画像のポイントnのピクセルの値を表します。

For the moment we treat the image as though it were a single row of pixels composed of all the rows in the image lined up next to each other.

とりあえず、画像を、互いに並んでいる画像内のすべての行で構成されるピクセルの単一行であるかのように扱います。

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *pn-s* | . . . | *pn-3* | *pn-2* | *pn-1* | *pn* |  |  |  |  |

s

This gives rise to some anomalies when starting a new raster line, but less than starting each line from scratch (later we will carry information down as well as across).

これにより、新しいラスターラインを開始するときにいくつかの異常が発生しますが、各ラインを最初から開始するよりも少なくなります（後で情報を上下に移動します）。

Let fs(n) be the sum of the values of the last s pixels at point n.

ポイントnからsピクセル前までの値の合計fs（n）を計算する。

The value of the resulting image T(n) is either 1 (for black) or 0 (for white) depending on　whether it is t percent darker than the average value of the previous s pixels.

結果の画素値T（n）は、前のsピクセルの平均値よりもtパーセント暗いかどうかに応じて、1（黒の場合）または0（白の場合）に分けられます。

Using 1/8th of the width of the image for the value of s and 15 for the value of t seems to yield the best results for a variety of images.

sの値に画像の幅の1/8を使用し、tの値に15を使用すると、さまざまな画像で最良の結果が得られるようです。

Figure 5-9 shows theresult of using this algorithm scanning the rows of pixels from left to right.

図5-9は、このアルゴリズムを使用してピクセルの行を左から右にスキャンした結果を示しています。

Figure 5-10 is exactly the same algorithm except the moving average scans from right to left.

図5-10は、右から左への移動平均スキャンを除き、まったく同じアルゴリズムです。

Notice in this image, that the left-most letters of the smaller text are incomplete, and there are more holes in the word PaperWorks.

この画像では、小さいテキストの左端の文字が不完全であり、PaperWorksという単語に多くの穴があることに注意してください。

Also, the rightmost black edge is much narrower.

また、右端の黒い縁はずっと狭くなっています。

This is all because the background lighting of the image gets darker from left to right.

これは、画像の背景照明が左から右に暗くなるためです。

A fast way to calculate an approximate (weighted) moving average is to subtract

1/s part of it and add the value of only the latest pixel instead of using all s pixels.

近似（加重）移動平均を計算する高速な方法は、減算することです

その1 / s部分を使用して、すべてのsピクセルを使用する代わりに、最新のピクセルのみの値を追加します。

Thus g(n) is an approximation of f(n) where

したがって、g（n）はf（n）の近似です。ここで、

The main difference between f*(n)* and *g(n)* is that *g(n)* puts more weight on the pixels that are　closest to *n*, which is good (in fact the weighting function is exponential).

f（n）とg（n）の主な違いは、g（n）がnに最も近いピクセルにより多くの重みを付けることであり、これは良いことです（実際、重み関数は指数関数的です）。

Notice that if all values of *pn* are the same, then *g(n)* = *f(n)*.

*pn*のすべての値が同じ場合、g（n）= f（n）であることに注意してください。

For example, if *s* = 2, then

たとえば、s = 2の場合、

A remaining question is how to start the algorithm, or what value to use for *g(0)*.

残りの質問は、アルゴリズムを開始する方法、またはg（0）に使用する値です

One possibility is to use *sp0*, but because of edge effects, *p0* is not always a representative value, so another possibility is 127*s* (based on the midvalue for 8 bit pixels).

1つの可能性はsp0を使用することですが、エッジ効果のため、p0は常に代表値ではないため、別の可能性は127s（8ビットピクセルの中間値に基づく）です。

In either case, this choice only affects the first few values of *g*.

どちらの場合でも、この選択はgの最初のいくつかの値にのみ影響します。

The weight of g*(0)* relative to the total of all weights used in calculating *gs(n)* is

gs（n）の計算に使用されるすべての重みの合計に対するg（0）の重みは

so if s = 10, for example, then for any n > 6, g(0) contributes less than 10% of g10(n); for any n > 22, g(0) contributes less than 1%.

たとえば、s = 10の場合、n> 6の場合、g（0）の寄与はg10（n）の10％未満です。 n> 22の場合、g（0）の寄与は1％未満です。

For s = 100, the 10% level is passed after 8 pixels, and the 1% level is passed after 68 pixels.

s = 100の場合、10％レベルは8ピクセル後に渡され、1％レベルは68ピクセル後に渡されます。

The results from using this approximate moving average are similar (actually better) than the precise moving average, as can be seen in Figure 5-11 and Figure 5-12.

この近似移動平均を使用した結果は、図5-11および図5-12に示すように、正確な移動平均と同様（実際には優れています）です。

It would be good if the moving average did not work better with lighting from one direction or an other.

移動平均が、ある方向または別の方向からの照明でうまく機能しなかったら良いでしょう。

Figure 5-13 shows the result of using another method to calculate the moving average.

図5-13は、別の方法を使用して移動平均を計算した結果を示しています。

Instead of using the trailing line behind each pixel, it uses (an evenly weighted, not approximate) moving average that is centred about the *nth* pixel.

各ピクセルの後ろの末尾の線を使用する代わりに、n番目のピクセルを中心とした（平均ではなく、均等に重み付けされた）移動平均を使用します。

This uses a definition of *f(n)* where

これは、f（n）の定義を使用します。ここで、

Using this method there are still some significant gaps in the letters, however, and it is also slower to calculate.

ただし、この方法を使用すると、文字にまだかなりのギャップがあり、計算が遅くなります。

Instead of traversing the image from left to right or right to left, another possibility is to traverse it alternatively from the left and from the right as illustrated in Figure 5-15, addressing the line-end problem alluded to earlier.\*

図5-15に示すように、画像を左から右または右から左に走査する代わりに、左と右から交互に画像を走査し、先ほど言及したラインエンドの問題に対処することもできます。\*

It makes very little difference with the centred average (as seen in Figure 5-14), but if we go back to using the approximate moving average defined by g(n), then alternate scanning improves things.

中央平均との差はほとんどありません（図5-14を参照）が、g（n）で定義された近似移動平均の使用に戻ると、代替スキャンは物事を改善します。

The only anomaly is in the “grey” regions of an unevenly lit image, where going in one direction can produce the opposite result from going in the other, producing an every-other-line effect. (See Figure 5-16 which is printed a little larger to show the effect better.)

唯一の異常は、照明が不均一な画像の「グレー」領域にあり、一方の方向に進むと、もう一方の方向に行くのとは反対の結果になり、1行おきの効果が得られます。 （効果をより良く示すために、少し大きく印刷されている図5-16を参照してください。）

Superficially, this image may not look as pretty, but it is better than any of the previous algorithms in important ways because there are no big holes or parts of letters missing.

表面的には、この画像はきれいに見えないかもしれませんが、大きな穴や文字の一部が欠けていないため、重要な点で以前のアルゴリズムよりも優れています。

The result is basically a combination of Figure 5-9 and Figure 5-10, but the righthand shadow has the every-other-line effect because when coming from the left, the background gets darker and the intensity of the shadow falls below the threshold, producing black.

結果は基本的に図5-9と図5-10の組み合わせですが、左から来ると背景が暗くなり、影の強度がしきい値を下回るため、右側の影には1行おきの効果があります、黒を生成します。

When the moving average wraps around the edge and includes the dark black edge (due to over-scanning), the threshold falls below the intensity of the shadow and the result is white as it traverses from right to left.

結果は基本的に図5-9と図5-10の組み合わせですが、左から来ると背景が暗くなり、影の強度がしきい値を要するため、右側の影には1行おきの効果があります、黒を生成します。

A small modification of this algorithm gets rid of the every-other-line effect and produces consistently good results across a wide range of images.

このアルゴリズムを少し修正すると、1行おきの効果がなくなり、広範囲の画像にわたって一貫して良好な結果が得られます。

The modification is to keep the previous line of approximate averages (which were calculated by scanning in the opposite direction) and to average the current line’s average with the previous line’s average, so we use

変更は、前の行の近似平均（反対方向にスキャンして計算された）を維持し、現在の行の平均を前の行の平均と平均することです。したがって、

This lets the threshold take some account of the vertical axis and produces the results in Figure 5-17.

これにより、しきい値は垂直軸を考慮して、図5-17の結果を生成します。

Notice how well-formed all the letters are.

すべての文字がどのように整形されているかに注目してください。

Also, this is one of the few algorithms that does not eliminate the thin horizontal line below the word “PaperWorks.”

また、これは「PaperWorks」という単語の下の細い水平線を除去しない数少ないアルゴリズムの1つです。

Images whose intensity change mainly in the vertical direction are not as challenging as ones that change horizontally, but for the record, Figure 5-18 shows the result for the rotated image with the illumination now varying from top to bottom.

強度が主に垂直方向に変化する画像は、水平方向に変化する画像ほど難しくはありませんが、記録については、図5-18は、照明が上から下に変化した回転画像の結果を示しています。

It seems unlikely that further developments would produce a technique that generates significantly better results, so this is currently used for the DigitalDesk.

今後の開発で大幅に優れた結果を生成する技術が生み出される可能性は低いと思われるため、現在これはDigitalDeskに使用されています。

The maximum amount of time it takes to run (for a full 768 x 575 image) is about 2 seconds on a SPARCstation 2 without paying special attention to optimization.

実行にかかる最大時間（768 x 575の完全なイメージの場合）は、最適化に特別な注意を払わずに、SPARCstation 2で約2秒です。

Steve Freeman has modified this algorithm to go about 6 times faster by hard-coding multiplications and divisions to use powers of two so that shifting can be used instead.

Steve Freemanは、乗算と除算をハードコーディングして2のべき乗を使用し、代わりにシフトを使用できるようにすることで、このアルゴリズムを約6倍高速化するように修正しました。

Further optimizations are possible, but will not make much difference for most interactions because they typically only operate on a small part of the image.

さらに最適化することは可能ですが、通常は画像のごく一部でのみ動作するため、ほとんどの相互作用では大きな違いはありません。

For thresholded full-frame video as used by the DoubleDigitalDesk, and BrightBoard [Staf93] however, these optimizations will be more important.

ただし、DoubleDigitalDeskおよびBrightBoard [Staf93]で使用されるしきい値付きフルフレームビデオの場合、これらの最適化がより重要になります。