Recovering Pixel Coordinates

Ronnie Johansson

February 28, 2001

Abstract

Image processing can be made more efficient by reducing the amount of data being processed. One problem that arises is how the coordinates of a pixel in the original image can be recovered when its coordinates in the reduced image are known (pixel coordinates recovery). This article explains how mathematical functions can be defined that easily maps coordinates between a modified and an original image under the condition that the modified image is the result of only cropping and subsampling. Crop and subsample transforms, which generate the suitable pixel coordinates mapping functions, are defined. The main focus of this work is to map row and column coordinates to other row and column coordinates, but it also considers the case where the pixels of the modified image are stored in a vector, and the index in that vector is the only information available about the location of a pixel.

Preface

In the summer of 2000, while I was working on my Master's Thesis at *The Institute of Physical and Chemical Research*[1] (RIKEN), Saitama, Japan, my supervisor at the institute was working on his project which, at the time, involved analyzing image data. At one point, he asked asked me to have a closer look at the following problem:

An image Img_{trunc} has been created by, firstly, cutting a rectangular piece out of an initial image Img_{org} (cropping), yielding an intermediate image Img_{imt}, and, secondly, keeping only every kth pixel in every row of Img_{imt} (in effect subsampling Img_{imt}). Given a pixel in Img_{trunc} with coordinates (x_{trunc}, y_{trunc}), how can the corresponding coordinates of Img_{org} (x_{org}, y_{org}) be recovered?

This article is the result of my study.

I would like to thank Svante Hellstadius for his comments on this article, and Dr. Igor Paromtchik for posing the problem.

Even though some parts of this article may look complex, it is based on fundamental mathematics. The intent of the article was to make its contents general and, hence, to satisfy many needs. To many readers, however, this generality is not necessary, and they will probably find it useful to read Section 1 and Appendix A first.

1 Introduction

Many applications (e.g., in surveillance, robotics, medicine etc) concerns the processing of image data, but algorithms that operate on images are quite often slow and,

[1] `http://www.riken.go.jp`

therefore, not suitable for real-time applications. Sometimes, however, input images contain more information than necessary, and efficiency can be gained by *truncating*[2] them. Two motivations to truncate an input image are:

1. only the information in some region of the input image is relevant,
2. *subsampling*[3] will, to some degree, not prevent the success of the applied image processing algorithm.

Motivation 1 arises when, e.g., only a part of the image is interesting for the moment (in a surveillance application this could be a region in which possible motion has been detected, which should be studied more closely). Motivation 2 arises when, e.g., an image is being scanned to detect (fairly large) regions of a particular color.

It is evidently useful, in some applications, to truncate an image to decrease the execution speed of image processing algorithms. However, how does the *pixels*[4] in the truncated image Img_{trunc} relate to the pixels in the original image Img_{org}? This question is important if the purpose of the image processing is to locate some image data in the coordinates of Img_{org}.

E.g., say that we have an application which wants to display an image Img_{org} and mark a possible "blue color"-region of the image with, e.g., a polygon. Let us further assume that it is useful to create a truncated image Img_{trunc} to increase the efficiency of the localization of the "blue color"-region. If the region is found, it can be expressed in the coordinates of Img_{trunc}, but this is not sufficient information for the application since it has to draw a polygon, surrounding the "blue color"-region, in the coordinates of Img_{org}.

One way to solve this problem, to acquire the corresponding Img_{org} coordinates given the Img_{trunc} coordinates, would be to create a lookup table of all pixels p in Img_{trunc} mapping to coordinates in Img_{org} (created in $O(p \log p)$). This lookup table would be built at the same time Img_{trunc} is created. It would be fairly fast to look up image coordinates ($O(\log p)$ using binary search), but the memory requirement would grow linearly ($O(p)$) in the number of pixels in Img_{trunc}.

Another way is to design, if possible, a formula $\mathbf{f_{trunc}}$ that takes the Img_{trunc} coordinates (denoted, e.g., (x_{trunc}, y_{trunc}) or (c_{trunc}, r_{trunc})) and returns the coordinates of Img_{org} (denoted, e.g., (x_{org}, y_{org}) or (c_{org}, r_{org})). Unlike, the solution with the lookup table, it requires no preparatory work[5], coordinates are acquired in constant time (simply computing the formula $\mathbf{f_{trunc}}$), and even the memory complexity is constant (no extra memory is needed).

The properties of the two solutions are summarized in Table 1.

Table 1: Comparison between lookup table and formula solutions

Solution	**Initial cost**	**On-line cost**	**Memory cost**
Lookup table	$O(p \log p)$	$O(\log p)$	$O(p)$
Formula	$O(1)$	$O(1)$	$O(1)$

[2] I.e., removing information from the image.

[3] Subsampling of an image is a sort of destructive data compression. An image that is subsampled loses, for instance, every other row and column, and the result is an image with smaller dimensions.

[4] "Pixel" is an abbreviation of "picture element" and simply refers to the smallest indivisible entity of a computer image.

[5] No initial work is necessary if the formula is not created in real-time.

This article deals with the latter solution.

Section 2 defines some of the terms used in this article. Section 3 describes the two image modification tools: crop and subsample, and their corresponding pixel coordinates mapping transforms. Section 4 gives examples of how to apply the transforms. Section 5 explains how row and column coordinates can be exchanged with pixel indices. Section 6 summarizes the article. Appendix A gives a very concrete example of how to use the contents of this article. In Appendix B, derivations of the pixel index to row and column functions are presented.

2 Definitions

An image is a 2-D array, a matrix, whose elements are pixels. In this article, images are classified[6] as

- original image, Img_{org},
- intermediate image, Img_{imt}, or
- truncated image, Img_{trunc}.

In the following discussion, Img_{org} refers to a "raw" image that has not been modified. Img_{imt} is an image that is the result of a modification made to another image. Finally, Img_{trunc} is the resulting image after some (possibly none) modifications.

There are two tools discussed in this article for modifying images:

- crop
- subsample

Cropping is the same as cutting a piece out of an image. In this article, only rectangular cuts are treated. Figure 1 provides an example.

The subsample tool removes columns and rows of pixels of an image. See Figure 2 for an example.

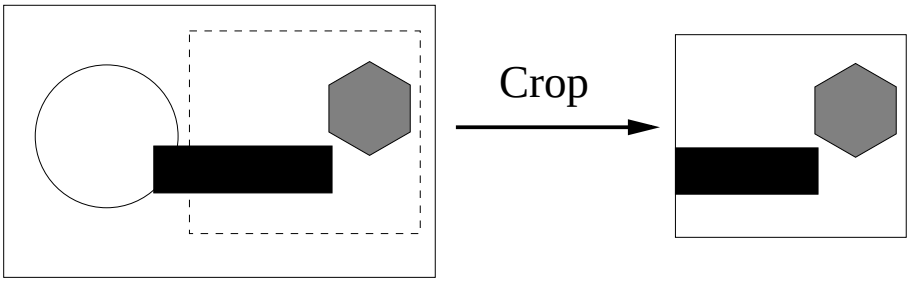

Figure 1: The left image is cropped resulting in the image on the right. Only the part of the left image that is within the dashed rectangle is preserved in the right image.

What is important when finding the corresponding coordinates of an original image and a modified one is not the values of the matrix (image) elements, rather it is the

[6] Please note that one and the same image can belong to more than one class, depending on the situation.

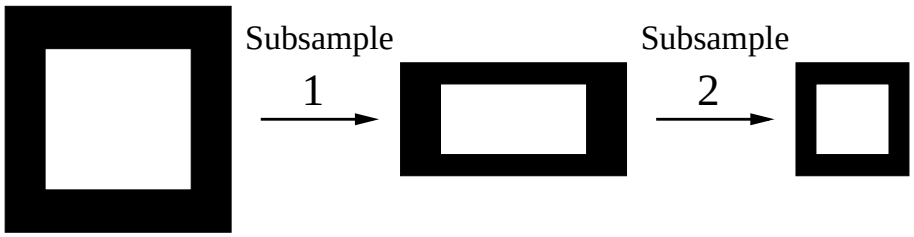

Figure 2: In a first subsampling (1), the image in the middle only keeps every other row of the original on the left. In the second subsampling (2), only every other column is preserved.

indices of the images. In the most simple case, let a *pixel coordinates mapping* (PCM) from the original image Img_{org} to itself be a function

$$\mathbf{f_{org}}(r,c) = \begin{bmatrix} row_{org}(r) = r \\ col_{org}(c) = c \end{bmatrix} \quad (1)$$

In Formula (1), r and c are a row and column value respectively, and $row_{org}(r)$ and $col_{org}(c)$ row and column mapping functions. $\mathbf{f_{org}}(r,c)$ is illustrated in Figure 3.

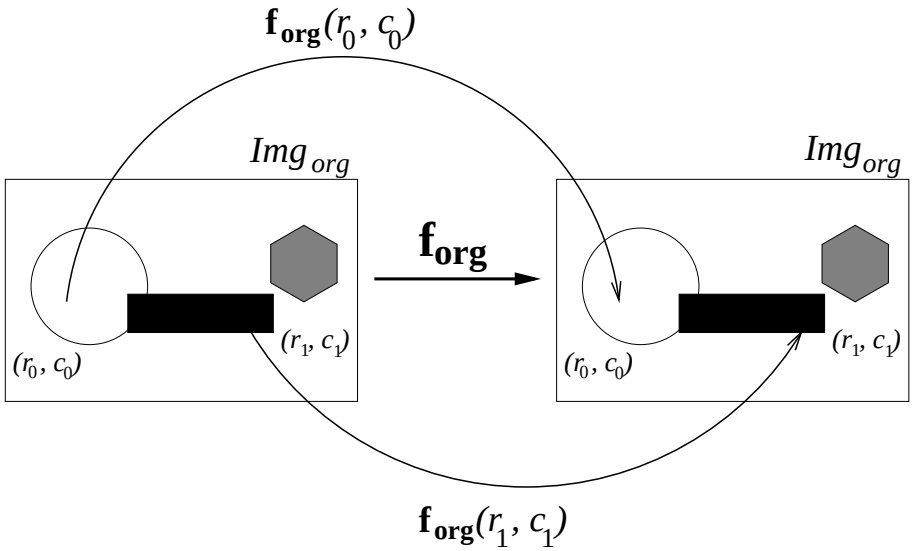

Figure 3: The $\mathbf{f_{org}}$ function maps rows and columns from an image Img_{org} to itself.

In general, let $\mathbf{f_{trunc}}$ be a PCM from an image Img_{trunc} to an original image Img_{org}. If $\mathbf{f_{trunc}} \equiv \mathbf{f_{org}}$ then Img_{trunc} and Img_{org} are identical.

Let the pixel coordinates mapping function $\mathbf{f}$ be transformed when the corresponding image is modified. The C is the crop transform, and the $\mathcal{S}$ the subsampling transform.

3 Truncation

This article describes two tools for truncation:

- crop
- subsample

Given an image *ImageA* (possibly already modified), the (possibly repeated) application of one or both of the truncation tools on the image will yield a new modified image *ImageB.* This section defines the transforms that should be applied to the PCM function of *ImageA* to produce the PCM function for *ImageB.*

3.1 Cropping

Let C be the crop transform, and $\mathbf{f_{imt}}$ be a PCM function from an image Img_{imt} to an original image Img_{org}. Furthermore, let Img_{trunc} be the result of applying the crop tool to Img_{imt}. The PCM function from Img_{trunc} to Img_{org} $\mathbf{f_{trunc}}$ will now be acquired by applying C to $\mathbf{f_{imt}}$, i.e., $\mathbf{f_{trunc}} \equiv C\{\mathbf{f_{imt}}\}$.

Let us have a closer look at a crop operation to find out the details of the crop transform C. Figure 4 shows the cropping of the image Img_{imt} to image Img_{trunc}. $srtrow_{imt}$ and $srtcol_{imt}$ are the least indices of the rows and columns of Img_{imt} that are included in Img_{trunc}. $lorow_{trunc}$ and $locol_{trunc}$ and the start indices of the rows and columns of Img_{trunc}.

If a PCM function $\mathbf{f_{trunc \rightarrow imt}}$ for each pair of coordinates in Img_{trunc} (r_{trunc}, c_{trunc}) to the corresponding (r_{imt}, c_{imt}) in Img_{imt} can be found, then the PCM function from Img_{trunc} to Img_{org} is also determined, since the PCM function $\mathbf{f_{imt}} = (row_{imt}(r_{imt}), col_{imt}(c_{imt}))$, i.e., the mapping from Img_{imt} to Img_{org}, is known.

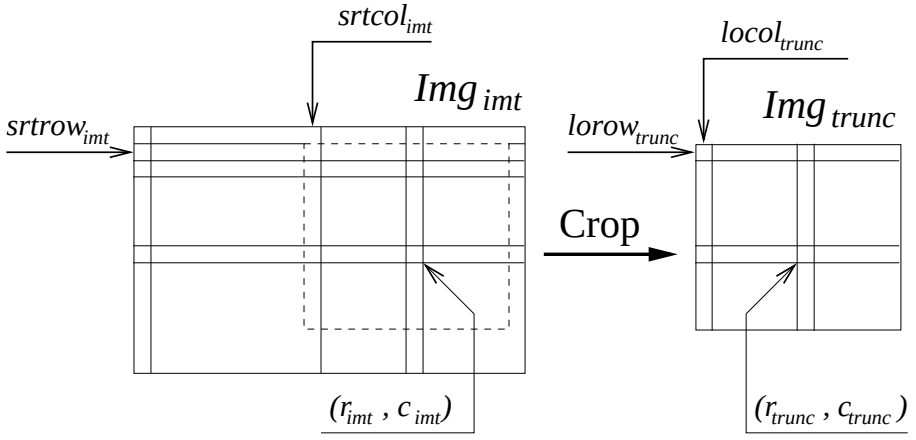

Figure 4: The image on the right is the truncated image Img_{trunc} which has a lowest row index $lorow_{trunc}$ and a lowest column index $locol_{trunc}$. The image on the left is the intermediate Img_{imt}. The $srtrow_{imt}$ and $srtcol_{imt}$ are the indices of the least row and column of Img_{imt} that is included in Img_{trunc}.

Figure 5 provides an example. The purpose of the example is to show that the transform takes row and column indices into consideration and does not restrict indices to start at certain numbers.

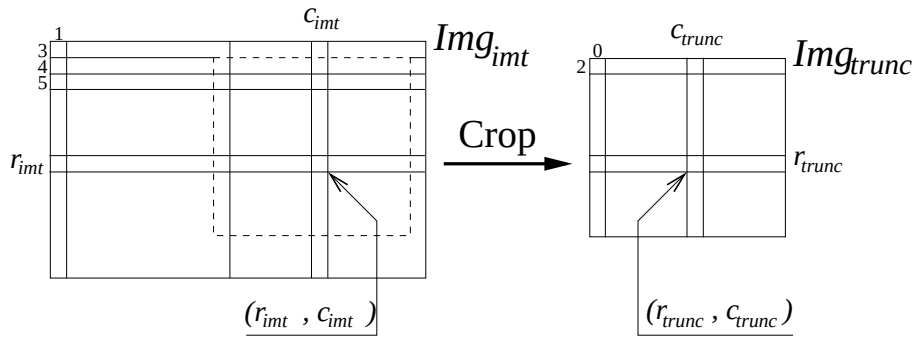

Figure 5: This figure illustrates that row and column indices are not restricted to start at certain numbers. I.e., the row numbering of Img_{trunc} starts with 2 and its column numbering with 0.

Notice that the mapping of rows and columns can be handled separately. The row mapping is only dependent on r_{trunc}, $lorow_{trunc}$, and $srtrow_{imt}$, while the column mapping only is dependent on c_{trunc}, $locol_{trunc}$, and $srtcol_{imt}$. Let $g(x;lo,srt)$ be a function which takes one variable x, and two parameters lo and srt:

$$g(x;lo,srt) = x - lo + srt \tag{2}$$

The mappings of rows and columns from Img_{trunc} to Img_{imt} can now be expressed as

$$\begin{aligned} row_{trunc\rightarrow imt}(r_{trunc}) = g(r_{trunc};lorow_{trunc},srtrow_{imt}) = \\ r_{trunc} - lorow_{trunc} + srtrow_{imt} \end{aligned} \tag{3}$$

and

$$\begin{aligned} col_{trunc\rightarrow imt}(c_{trunc}) = g(c_{trunc};locol_{trunc},srtcol_{imt}) = \\ c_{trunc} - locol_{trunc} + srtcol_{imt} \end{aligned} \tag{4}$$

respectively. Finally, the transform C can now be defined as

$$\begin{aligned} C[srtrow_{imt},srtcol_{imt},lorow_{trunc},locol_{trunc}]\{\mathbf{f_{imt}}\} = \\ \left[\begin{array}{l} row_{trunc}(r_{trunc}) = row_{imt}\big(g(r_{trunc};lorow_{trunc},srtrow_{imt})\big) \\ col_{trunc}(c_{trunc}) = col_{imt}\big(g(c_{trunc};locol_{trunc},srtcol_{imt})\big) \end{array}\right] = \\ \left[\begin{array}{l} row_{trunc}(r_{trunc}) = row_{imt}(r_{trunc} - lorow_{trunc} + srtrow_{imt}) \\ col_{trunc}(c_{trunc}) = col_{imt}(c_{trunc} - locol_{trunc} + srtcol_{imt}) \end{array}\right] \end{aligned} \tag{5}$$

3.2 Subsampling

Let $\mathcal{S}$ be the subsample transform, and $\mathbf{f_{imt}}$, once again, be a PCM function from an image Img_{imt} to an original image Img_{org}. Furthermore, let Img_{trunc} be the result of applying the subsample tool to Img_{imt}. The PCM from Img_{trunc} to Img_{org} $\mathbf{f_{trunc}}$ will now be acquired by applying $\mathcal{S}$ to $\mathbf{f_{imt}}$, i.e., $\mathbf{f_{trunc}} \equiv \mathcal{S}\{\mathbf{f_{imt}}\}$.

Figure 6 shows the subsampling of the image Img_{imt} to image Img_{trunc}. The parameters $steprow_{imt}$ and $stepcol_{imt}$ decide how many rows and columns respectively should

be kept. E.g., always keep the first row and then every $steprow_{row}$th row from there on. Columns are handled analogously. The same as with the crop transform, if we can find a PCM function $\mathbf{f_{trunc \to imt}}$ for each pair of coordinates in Img_{trunc} (r_{trunc}, c_{trunc}) to (r_{imt}, c_{imt}) in Img_{imt}, then the PCM function from Img_{trunc} to Img_{org} is also determined, since the PCM function $\mathbf{f_{imt}} = (row_{imt}(r_{imt}), col_{imt}(c_{imt}))$, i.e., mapping from Img_{imt} to Img_{org}, is known.

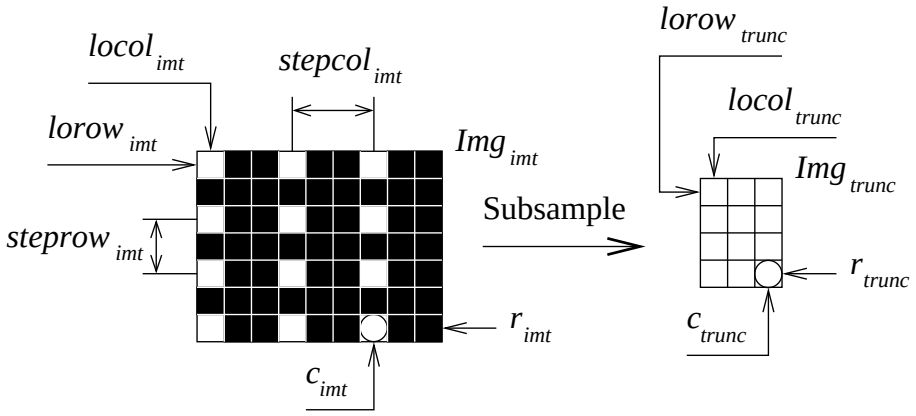

Figure 6: In the process of subsampling image Img_{imt} to image Img_{trunc}, only every $steprow_{imt}$th row and every $stepcol_{imt}$th column are kept. $lorow_{imt}$, $locol_{imt}$, $lorow_{trunc}$, and $locol_{trunc}$ are the indices of the first rows and columns of Img_{imt} and Img_{trunc} respectively.

Mapping of rows and columns can be handled separately also in the case of the subsample transform. The row mapping is only dependent on r_{trunc}, $lorow_{trunc}$, $lorow_{imt}$, and $steprow_{imt}$, while the column mapping only is dependent on c_{trunc}, $locol_{trunc}$, $locol_{imt}$, and $stepcol_{imt}$. Let $h(x; loNew, loOld, step)$ be a function which takes one variable x, and three parameters $loOld$, $loNew$, and $step$:

$$h(x; loNew, loOld, step) = (x - loNew) \cdot step + loOld \tag{6}$$

The mappings of rows and columns from Img_{trunc} to Img_{imt} can now be expressed as

$$\begin{aligned} row_{trunc \to imt}(r_{trunc}) &= h(r_{trunc}; lorow_{trunc}, lorow_{imt}, steprow_{imt}) = \\ &(r_{trunc} - lorow_{trunc}) \cdot steprow_{imt} + lorow_{imt} \end{aligned} \tag{7}$$

and

$$\begin{aligned} col_{trunc \to imt}(c_{trunc}) &= h(c_{trunc}; locol_{trunc}, lorow_{imt}, stepcol_{imt}) = \\ &(c_{trunc} - locol_{trunc}) \cdot stepcol_{imt} + locol_{imt} \end{aligned} \tag{8}$$

respectively. Finally, the transform $\mathcal{S}$ can be defined as

$$\begin{aligned} &\mathcal{S}[lorow_{imt}, locol_{imt}, lorow_{trunc}, locol_{trunc}, steprow_{imt}, stepcol_{imt}]\{\mathbf{f_{imt}}\} = \\ &\begin{bmatrix} row_{trunc}(r_{trunc}) = row_{imt}\big(h(r_{trunc}; lorow_{trunc}, lorow_{imt}, steprow_{imt})\big) \\ col_{trunc}(c_{trunc}) = col_{imt}\big(h(c_{trunc}; locol_{trunc}, locol_{imt}, stepcol_{imt})\big) \end{bmatrix} = \\ &\begin{bmatrix} row_{trunc}(r_{trunc}) = row_{imt}\big((r_{trunc} - lorow_{trunc}) \cdot steprow_{imt} + lorow_{imt}\big) \\ col_{trunc}(c_{trunc}) = col_{imt}\big((c_{trunc} - locol_{trunc}) \cdot stepcol_{imt} + locol_{imt}\big) \end{bmatrix} \end{aligned} \tag{9}$$

4 Recovery of Coordinates

With the help of the transforms from Section 3, it is now possible to construct a PCM function for any manipulation of an image Img_{org} as long as only the tools crop and subsample are used. The transforms (configured with the appropriate parameters) should be applied to $\mathbf{f_{org}}$ correspondingly to the transformation of the Img_{org} to acquire the PCM function $\mathbf{f_{trunc}}$ for Img_{trunc}.

Once the sequence of image transformations is known, the PCM function can be calculated.

Here is an example:

Say that we want to modify an image Img_{org} with one crop and one subsample. The parameters of the crop and subsample do not have to be known in advance; they may be calculated at run-time. The PCM function for the resulting image Img_{trunc} is $\mathbf{f_{trunc}}$ and is calculated this way

$$\begin{aligned}
\mathbf{f_{trunc}} &= \mathcal{S}[lorow_{imt}, locol_{imt}, lorow_{trunc}, locol_{trunc}, steprow_{imt}, stepcol_{imt}] \\
C[srtrow_{org}, srtcol_{org}, lorow_{imt}, locol_{imt}]\{\mathbf{f_{org}}\} &= \Big\{\text{suppressing the printing of all parameters}\Big\} = \\
\mathcal{S}[\ldots]C[\ldots]\{\mathbf{f_{org}}\} &= \left[\begin{array}{l} row_{trunc}(r_{trunc}) = row_{imt}(r_{trunc} - lorow_{trunc} + srtrow_{imt}) \\ col_{trunc}(c_{trunc}) = col_{imt}(c_{trunc} - locol_{trunc} + srtcol_{imt}) \end{array}\right] = \\
&\left[\begin{array}{l} row_{trunc}(r_{trunc}) = (r_{trunc} - lorow_{trunc}) \cdot steprow_{imt} + lorow_{imt} - lorow_{imt} + srtrow_{org} \\ col_{trunc}(c_{trunc}) = (c_{trunc} - locol_{trunc}) \cdot stepcol_{imt} + locol_{imt} - locol_{imt} + srtcol_{org} \end{array}\right] = \\
&\{\text{Eliminating } lorow_{imt} \text{ and } locol_{imt}\} = \\
&\left[\begin{array}{l} row_{trunc}(r_{trunc}) = (r_{trunc} - lorow_{trunc}) \cdot steprow_{imt} + srtrow_{org} \\ col_{trunc}(c_{trunc}) = (c_{trunc} - locol_{trunc}) \cdot stepcol_{imt} + srtcol_{org} \end{array}\right] \qquad (10)
\end{aligned}$$

A more general formula can easily be derived. In this case, it is assumed that the original image Img_{org} (with PCM function $\mathbf{f_{0,0}}$) is first cropped n times and then subsampled m times. The resulting image Img_{trunc} will have the PCM function $\mathbf{f_{n,m}}$.

$$\begin{aligned}
\mathbf{f_{n,m}} &= \mathcal{S}^m[[\ldots]_1 \ldots [\ldots]_m]C^n[[\ldots]_1 \ldots [\ldots]_n]\{\mathbf{f_{0,0}}\} = \mathcal{S}^m[[\ldots]_1 \ldots [\ldots]_m]\{\mathbf{f_{n,0}}\} = \\
&\mathcal{S}^m[[\ldots]_1 \ldots [\ldots]_m] \left[\begin{array}{l} row_{n,0}(r_{n,0}) = row_{0,0}\big(r_{n,0} + \sum_{i=0}^{n}(srtrow_{i,0} - lorow_{i+1,0})\big) \\ col_{n,0}(c_{n,0}) = col_{0,0}\big(c_{n,0} + \sum_{i=0}^{n}(srtcol_{i,0} - locol_{i+1,0})\big) \end{array}\right] = \\
&\left[\begin{array}{lcl} row_{n,m}(r_{n,m}) & = & row_{0,0}\big(r_{n,m} \cdot \prod_{j=1}^{m} steprow_{n,j} - \sum_{j=1}^{m} lorow_{n,j} \prod_{k=1}^{j} steprow_{n,k} + \\ & & lorow_{n,0} + \sum_{i=0}^{n-1}(srtrow_{i,0} - lorow_{i+1,0})\big) \\ col_{n,m}(c_{n,m}) & = & col_{0,0}\big(c_{n,m} \cdot \prod_{j=1}^{m} stepcol_{n,j} - \sum_{j=1}^{m} locol_{n,j} \prod_{k=1}^{j} stepcol_{n,k} + \\ & & locol_{n,0} + \sum_{i=0}^{n-1}(srtcol_{i,0} - locol_{i+1,0})\big) \end{array}\right] \qquad (11)
\end{aligned}$$

Formula (11) can be made much more simple. Since the row and column indexation is arbitrary, it can be fixed to 0, i.e., $lorow_{i,j} = locol_{i,j} = 0$ for all i and j. This yields the much more simple equation:

$$\begin{aligned}
\mathbf{f_{n,m}} &= \mathcal{S}^m[[\ldots]_1 \ldots [\ldots]_m]C^n[[\ldots]_1 \ldots [\ldots]_n]\{\mathbf{f_{0,0}}\} = \\
&\left[\begin{array}{l} row_{n,m}(r_{n,m}) = row_{0,0}(r_{n,m} \cdot \prod_{j=1}^{m} steprow_{n,j} + \sum_{i=0}^{n-1} srtrow_{i,0}) \\ col_{n,m}(c_{n,m}) = col_{0,0}(c_{n,m} \cdot \prod_{j=1}^{m} stepcol_{n,j} + \sum_{i=0}^{n-1} srtcol_{i,0}) \end{array}\right] \qquad (12)
\end{aligned}$$

5 Recovery With Pixel Index

Sometimes, the final image Img_{trunc} is treated as a one dimensional vector of pixels rather than a matrix with rows and columns. Given the pixel index i_{trunc}, the start pixel index $srtidx_{trunc}$, $lorow_{trunc}$, $locol_{trunc}$, and the maximum column index $hicol_{trunc}$, the transformations between pixel indices and rows and columns can be calculated this way (the derivations of these equations are shown in Appendix B):

$$r_{trunc}(i_{trunc}) = lorow_{trunc} + \left\lfloor \frac{\Delta i}{wth_{trunc}} \right\rfloor \tag{13}$$

$$c_{trunc}(i_{trunc}) = locol_{trunc} + \Delta i - \left\lfloor \frac{\Delta i}{wth_{trunc}} \right\rfloor \cdot wth_{trunc} \tag{14}$$

In the equations, the following abbreviations are used: $\Delta i = i_{trunc} - srtidx_{trunc}$ and $wth_{trunc} = hicol_{trunc} - locol_{trunc} + 1$.

Formula (13) and Formula (14) can be inserted directly into any PCM function $\mathbf{f}(r,c)$ yielding $\mathbf{f}(i) = (row(r(i)), col(c(i)))$. The equations inserted into Formula (10) yield

$$\begin{aligned}
\mathbf{f_{trunc}}(i_{trunc}) = \mathcal{S}[\ldots]C[\ldots]\{\mathbf{f_{org}}\} = \\
\left[\begin{array}{l} row_{trunc}\big(r_{trunc}(i_{trunc})\big) = row_{imt}\big(r_{trunc}(i_{trunc}) - lorow_{trunc} + srtrow_{imt}\big) \\ col_{trunc}\big(c_{trunc}(i_{trunc})\big) = col_{imt}\big(c_{trunc}(i_{trunc}) - locol_{trunc} + srtcol_{imt}\big) \end{array} \right] = \\
\left[\begin{array}{l} row_{trunc}\big(r_{trunc}(i_{trunc})\big) = \big(r_{trunc}(i_{trunc}) - lorow_{trunc}\big) \cdot steprow_{imt} + srtrow_{org} \\ col_{trunc}\big(c_{trunc}(i_{trunc})\big) = \big(c_{trunc}(i_{trunc}) - locol_{trunc}\big) \cdot stepcol_{imt} + srtcol_{org} \end{array} \right] = \\
\left[\begin{array}{l} row_{trunc}(i_{trunc}) = \big((lorow_{trunc} + \left\lfloor \frac{\Delta i}{wth_{trunc}} \right\rfloor) - lorow_{trunc}\big) \cdot steprow_{imt} + srtrow_{org} \\ col_{trunc}(i_{trunc}) = \big((locol_{trunc} + \Delta i - \left\lfloor \frac{\Delta i}{wth_{trunc}} \right\rfloor \cdot wth_{trunc}) - locol_{trunc}\big) \cdot stepcol_{imt} + srtcol_{org} \end{array} \right] = \\
\{\text{Eliminating } lorow_{trunc} \text{ and } locol_{trunc}\} = \\
\left[\begin{array}{l} row_{trunc}(i_{trunc}) = \left\lfloor \frac{\Delta i}{wth_{trunc}} \right\rfloor \cdot steprow_{imt} + srtrow_{org} \\ col_{trunc}(i_{trunc}) = (\Delta i - \left\lfloor \frac{\Delta i}{wth_{trunc}} \right\rfloor \cdot wth_{trunc}) \cdot stepcol_{imt} + srtcol_{org} \end{array} \right]
\end{aligned} \tag{15}$$

Unfortunately, wth_{trunc} is expressed with $hicol_{trunc}$, which also has to be calculated. This is not discussed in this article.

6 Summary and Discussion

This article presents two tools for digital image processing: crop and subsample, but the focus is on how to translate a pair of coordinates for a pixel in a modified (or truncated) image to the the corresponding coordinates in the original image. To construct the necessary pixel coordinates mapping (PCM) functions (that maps coordinates from a modified image to the original), transforms C and $\mathcal{S}$ are introduced.

Pixel coordinates may be represented as pairs of row and column indices, but sometimes, e.g., when the modified image is stored in a vector and the corresponding row and column indices are unknown, pixel indices are better to use. The article shows how PCM functions can be modified to take pixel indices instead of row and column indices.

A few examples show how the transforms can be used.

The transforms are most likely to be useful in applications which crop or subsample images, and where there is a need to recover the original coordinates of pixels. It is also likely that a programmer uses the transforms to construct a PCM function that is hard coded into his or her program.

Also note that the transforms can be used with matrices containing any type of data as long as the modifications of them are limited to crop and subsample.

A Example

This section contains a fictitious example that will illustrate how the work in this article can be brought into practice. Let us say we have a monitoring system that monitors a pipe to detect emerging leaks. The system receives a digitized infrared video image of the pipe once every second. Leaks of hot water are characterized by the particular color mark they leave on the video image.

The infrared images have a 320x256 pixels resolution, and, hence, 81920 pixels (=320x256) each. The hardware and image processing algorithm in use does not allow all pixels to be considered before a new infrared image is generated in the system. The system designers make the assumption that it is sufficient to check just a subset of the pixels in each image. They divide the image matrix into sixteen cells and decide that for each image that is generated in the system, only the pixels of one of the sixteen cells will be considered. The image decomposition is illustrated in Figure 7.

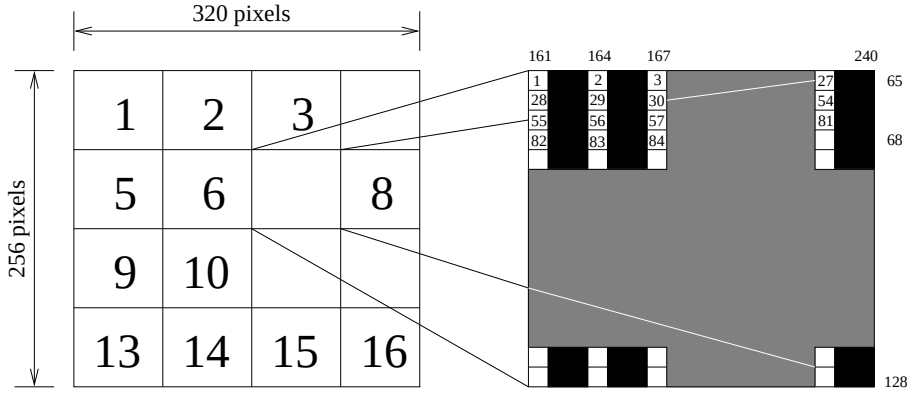

Figure 7: The infrared images are divided into 16 cells. For each image, only one cell is considered. In this figure, Cell 7 has been magnified. The numbers above the magnified cell are column indices of the original image, and the numbers to the right are the row indices of the original image. The white pixels of the cell are the ones left after a subsampling with $stepcol = 3$ and $steprow = 1$. The numbers in the white pixels are the pixel indices of the resulting image and *stridx* (the number of the first pixel index) is 1.

Each cell contains 5120 pixels, but the designers realize that even this is too much. What they also realize is that the accuracy of checking each pixel of the cell is unnecessarily high, that small errors in the image might be mistaken for leaks, that it is sufficient to detect larger regions of deviant colors, and that processing time can be gained by reducing the data even further. It is decided that the cell should be subsampled with the parameters $steprow = 1$ and $stepcol = 3$ keeping only every third pixel

of each row in the cell.

Furthermore, the designers decide that the 1728 pixels that are left, after the subsampling, should be stored in a result vector. They write a program based on Algorithm 1.

Algorithm 1: Image processing algorithm
Input: An unmodified IR image Img_{org}, and the number of the cell which should be processed $cell_{no}$
Output: A list of interesting coordinates of Img_{org}
PROCESS IMAGE(Img_{org}, $cell_{no}$)
(1) $coords_{vec} \leftarrow$ the empty set
(2) $subpic \leftarrow$ CREATEPICOBJ(Img_{org},$cell_{no}$)
(3) $srtrow_{org} \leftarrow$ GETROW($subpic$)
(4) $srtcol_{org} \leftarrow$ GETCOL($subpic$)
(5) **foreach** pixel pxl in $subpic$
(6) **if** INTERESTINGDATA(pxl)
(7) $coords_{vec} \leftarrow$ F(pxl,$srtrow_{org}$,$srtcol_{org}$)
(8) **return** $coords_{vec}$

In line (1) of the algorithm, the result vector is initialized to be empty. In line (2), Img_{org} is cropped with respect to $cell_{no}$ and then subsampled with the parameters described above. The result, which is stored in $subpic$, is a data object that contains, apart from the pixels that are left, the indices of the start row and start column. Line (3) and line (4) simply retrieves the start row and start column indices from the $subpic$ data object. In line (5), the loop that iterates through all pixels of $subpic$ begins. In line (6) a call is made to function INTERESTINGDATA that analyzes the pixel pxl in order to determine if it is interesting to the application. If it is, in line (7), the coordinates of pxl expressed in the the coordinates of Img_{org} are stored in $coords_{vec}$. In line (8), $coords_{vec}$ is returned and the execution of the algorithm completed.

So for each "interesting" pxl pixel in the subsampled cell, the PCM function $\mathbf{f}(i; srtrow_{org}, srtcol_{org})$ calculates its coordinates in Img_{org}. In this example, Formula (15) defines the **F** function in line (7):

$$\mathbf{f}(i; srtrow_{org}, srtcol_{org}) = \begin{bmatrix} \left\lfloor \frac{\Delta i}{wth_{trunc}} \right\rfloor \cdot steprow_{imt} + srtrow_{org} \\ (\Delta i - \left\lfloor \frac{\Delta i}{wth_{trunc}} \right\rfloor \cdot wth_{trunc}) \cdot stepcol_{imt} + srtcol_{org} \end{bmatrix} =$$

$$\left\{ \begin{array}{l} srtrow_{org} = 1, stepcol_{org} = 3, wth_{trunc} = 27 \text{ (see Figure 7)} \\ \Delta i = i - srtidx = \{\text{the pixel index numbering starts with 1}\} = i - 1 \end{array} \right\} =$$

$$\begin{bmatrix} \left\lfloor \frac{i-1}{27} \right\rfloor + srtrow_{org} \\ \left((i-1) - \left\lfloor \frac{i-1}{27} \right\rfloor \cdot 27\right) \cdot 3 + srtcol_{org} \end{bmatrix} \tag{16}$$

E.g., let $i = 84$ in Figure 7 with $srtrow_{org} = 65$ and $srtcol_{org} = 161$. Insert the values into Formula (16):

$$\mathbf{f}(84; 65, 161) = \begin{bmatrix} \left\lfloor \frac{84-1}{27} \right\rfloor + 65 \\ \left((84-1) - \left\lfloor \frac{84-1}{27} \right\rfloor \cdot 27\right) \cdot 3 + 161 \end{bmatrix} = \begin{bmatrix} 68 \\ 167 \end{bmatrix} \tag{17}$$

The result in Formula (17) agrees with Figure 7.

B Pixel Index Formula Derivation

The purpose of this appendix is to show how Formula (13) and Formula (14) can be derived.

Let us consider the reverse, the pixel index i as a function of row and column coordinates. From Figure 8 the following formula is derived:

$$\begin{aligned} i(r,c) &= \big(row(i) - lorow\big) \cdot wth + \big(col(i) - locol + 1\big) + (srtidx - 1) = \ldots \\ &\ldots = \big(row(i) - lorow\big) \cdot wth + \big(col(i) - locol\big) + srtidx \end{aligned} \tag{18}$$

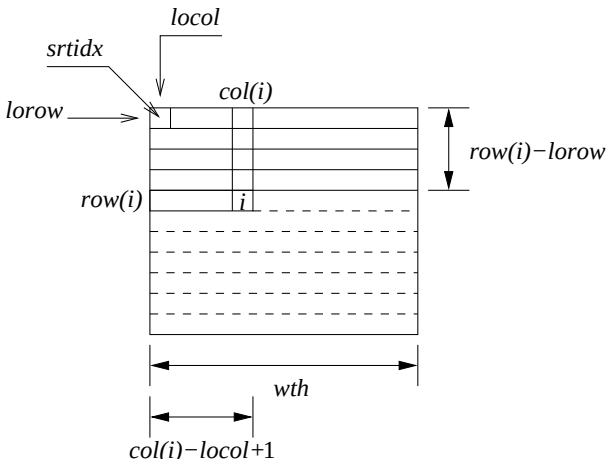

Figure 8: Pixel index i in the figure has the corresponding coordinates $\big($row(i),col(i)$\big)$. $srtidx$ is the first pixel index number. wth is the width in pixels of the image.

Try and solve Equation (18) with respect to $row(i)$:

$$\begin{aligned} \frac{i(r,c) - srtidx}{wth} &= \big(row(i) - lorow\big) + \frac{col(i) - locol}{wth} \Rightarrow \\ &\{\text{Let } \Delta i = i(r,c) - srtidx\} \Rightarrow \\ \left\lfloor \frac{\Delta i}{wth} \right\rfloor &= \left\lfloor \big(row(i) - lorow\big) + \frac{col(i) - locol}{wth} \right\rfloor \Rightarrow \\ \left\{ \begin{array}{l} \lfloor row(i) - lorow \rfloor = row(i) - lorow, \text{ since the subtraction yields an integer.} \\ 0 \le \frac{col(i) - locol}{wth} < 1 \text{ (since } wth = hicol - locol + 1) \Rightarrow \left\lfloor \frac{col(i) - locol}{wth} \right\rfloor = 0 \end{array} \right\} &\Rightarrow \\ \left\lfloor \frac{\Delta i}{wth} \right\rfloor = row(i) - lorow \Rightarrow row(i) &= lorow + \left\lfloor \frac{\Delta i}{wth} \right\rfloor \end{aligned} \tag{19}$$

■

Let us finally derive $col(i)$, once again using Equation (18):

$$\begin{aligned} col(i) = locol + \big(i(r,c) - srtidx\big) - \big(row(i) - lorow\big) \cdot wth \Rightarrow \\ \left\{ \begin{array}{l} \Delta i = i - srtidx \\ row(i) \text{ from Equation (19)} \end{array} \right\} \Rightarrow \\ col(i) = locol + \Delta i - \left(lorow + \left\lfloor \frac{\Delta i}{wth} \right\rfloor - lorow\right) \cdot wth \Rightarrow \end{aligned}$$

$$\{\text{Eliminating } lorow\} \Rightarrow$$

$$col(i) = locol + \Delta i - \left\lfloor \frac{\Delta i}{wth} \right\rfloor \cdot wth \tag{20}$$

■