The Discrete Pulse Transform in Two Dimensions

I. Fabris-Rotelli†, S.J. van der Walt⋆

† Department of Statistics,
University of Pretoria, South Africa
⋆ Applied Mathematics Division, Department of Mathematical Sciences,
University of Stellenbosch, South Africa

inger.fabris-rotelli@up.ac.za, stefan@sun.ac.za

Abstract

During the last three decades, the LULU operator and its close relative, the Discrete Pulse Transform (DPT), were developed and applied to signal processing problems. Its recent extension into higher dimensions paved the way for applications on signals such as images. In this paper, we present a memory-efficient implementation of the two-dimensional Discrete Pulse Transform and use it to explore the scale space exposed by the DPT.

1. Introduction

Research initiated by Rohwer on the so-called LULU smoothers culminated in a well-developed theory for nonlinear operators in signal processing [28, 26, 27, 29, 21] . The LULU smoothers are given by the operators L_n, U_n and their compositions. They act on sequences $x = \{x_i\} \in \ell_1$, such that $\sum |x_i| < \infty$ [1]. To extend these operators into higher dimensions, Serra's concept of morphological connections [30] is applied. This extension was first developed in [1] and [2].

Definition 1. Let B be an arbitrary non-empty set. A family $\mathcal{C}$ of subsets of B is called a connected class or a *connection* on B if

(i) $\emptyset \in \mathcal{C}$,

(ii) $\{x\} \in \mathcal{C}$ for each $x \in B$ and

(iii) for any family $\{C_i\} \subseteq \mathcal{C}$ we have

$$\bigcap_{i\in I} C_i \neq \emptyset \Longrightarrow \bigcup_{i\in I} C_i \in \mathcal{C}.$$

If a set C belongs to a connection $\mathcal{C}$, then C is called *connected*.

Given a point $x \in \mathbb{Z}^2$ and $n \in \mathbb{N}$ we denote by $\mathcal{N}_n(x)$ the set of all connected sets of size $n+1$ that contain point x, that is,

$$\mathcal{N}_n(x) = \{V \in \mathcal{C} : x \in V,\ \mathrm{card}(V) = n+1\}$$

where $\mathrm{card}(V)$ is the number of elements in the set V. In image analysis, the connectivity is defined on a graph via a neighbour relation, e.g., 4-connectivity or 8-connectivity. In the rest of this paper, we use 4-connectivity only. We define the operators L_n and U_n on $\mathcal{A}(\mathbb{Z}^2)$, the vector lattice (admitting a *supremum* and *infimum*) of all real functions defined on $\mathbb{Z}^2$ with respect to the usual point-wise defined addition, scalar multiplication and partial order. Thus, an image $f(x)$ is defined on the domain $\mathbb{Z}^2$ with values in $\mathcal{A}(\mathbb{Z}^2)$.

Definition 2. Let $f \in \mathcal{A}(\mathbb{Z}^d)$ and $n \in \mathbb{N}$. Then

$$\begin{aligned} L_n(f)(x) &= \max_{V\in\mathcal{N}_n(x)} \min_{y\in V} f(y),\ x \in \mathbb{Z}^d, \\ U_n(f)(x) &= \min_{V\in\mathcal{N}_n(x)} \max_{y\in V} f(y),\ x \in \mathbb{Z}^d. \end{aligned}$$

The properties of these operators are dealt with in [1, 2].

2. The Discrete Pulse Transform

The Discrete Pulse Transform is obtained by the successive removal of peaks (local maximum sets) and valleys (local minimum sets) from the image by applying L_n and U_n respectively.

Definition 3. Let $V \in \mathcal{C}$, a connected set. A point $x \notin V$ is called *adjacent* to V if $V \cup \{x\} \in \mathcal{C}$. The set of all points adjacent to V is denoted by $\mathrm{adj}(V)$, that is,

$$\mathrm{adj}(V) = \{x \in \mathbb{Z}^2 : x \notin V, V \cup \{x\} \in \mathcal{C}\}.$$

A connected subset V of $\mathbb{Z}^2$ is called a *local maximum set* of $f \in \mathcal{A}(\mathbb{Z}^2)$ if

$$\sup_{y\in\mathrm{adj}(V)} f(y) < \inf_{x\in V} f(x).$$

Similarly V is a *local minimum set* if

$$\inf_{y\in\mathrm{adj}(V)} f(y) > \sup_{x\in V} f(x).$$

[1] This condition is necessary to ensure that the total variation of the sequence, and that of the DPT, is defined [26].

The LULU operators act as follows on local maximum and minimum sets.

- The application of L_n (U_n) removes local maximum (minimum) sets of size smaller or equal to n.
- Neither operator creates new local minimum or maximum sets where none existed before. It may happen that existing minimum or maximum sets are enlarged when adjacent sets are joined.
- $L_n(f) = f$ ($U_n(f) = f$) if and only if f does not have local maximum (minimum) sets of size n or less.
- $(L_n \circ U_n)(f)$and $(U_n \circ L_n)(f)$ have neither local maximum sets nor local minimum sets of size n or less. Furthermore, $(L_n \circ U_n)(f) = (U_n \circ L_n)(f) = f$ if and only if f does not have local maximum sets or local minimum sets of size less than or equal to n.

Let $N = \mathrm{card}(\mathrm{supp}(f))$, that is, the size of the image. We derive the DPT of $f \in \mathcal{A}(\mathbb{Z}^2)$ by applying iteratively the operators L_n, U_n with n increasing from 1 to N as follows

$$DPT(f) = (D_1(f), D_2(f), ..., D_N(f)), \qquad (1)$$

where the components of (1) are obtained through

$$\begin{aligned} D_1(f) &= (I - P_1)(f) \\ D_n(f) &= (I - P_n) \circ Q_{n-1}(f),\ n = 2, ..., N \end{aligned}$$

and $P_n = L_n \circ U_n$ or $P_n = U_n \circ L_n$ and $Q_n = P_n \circ ... \circ P_1$, $n \in \mathbb{N}$.

Definition 4. A function $\phi \in \mathcal{A}(\mathbb{Z}^2)$ is called a *pulse* if there exists a connected set V and a real number α such that

$$\phi(x) = \begin{cases} \alpha & \text{if} \quad x \in V \\ 0 & \text{if} \quad x \in \mathbb{Z}^2 \setminus V. \end{cases}$$

The set V is the support of the pulse ϕ, that is $\mathrm{supp}(\phi) = V$.

We thus obtain the following decomposition of the image $f \in \mathcal{A}(\mathbb{Z}^2)$,

$$f = \sum_{n=1}^{N} D_n(f) = \sum_{n=1}^{N} \sum_{s=1}^{\gamma(n)} \phi_{ns},$$

where each ϕ_{ns} is a discrete pulse of size n which can be positive or negative, and all discrete pulses of fixed size n have disjoint supports. For $n_1, n_2, s_1, s_2 \in \mathbb{N}$ such that $n_1 < n_2$, $1 \le s_1 \le \gamma(n_1)$ and $1 \le s_2 \le \gamma(n_2)$, we have that

$$\mathrm{supp}(\phi^*_{n_1 s_1}) \cap \mathrm{supp}(\phi^*_{n_2 s_2}) \neq \emptyset \Rightarrow \mathrm{supp}(\phi^*_{n_1 s_1}) \subset \mathrm{supp}(\phi^*_{n_2 s_2})$$

showing that as the decomposition progresses the supports of the pulses, which are either local maximum or minimum sets, are either nested (i.e., they grow), or are disjoint.

3. Implementation

The Discrete Pulse Transform decomposes a signal into a collection of pulses. In one dimension, a pulse is characterised by its start and end position, as well as by its amplitude (see Fig. 1). In two dimensions, a pulse describes a connected region over which function values are constant (we refer to 4-connection — in other words, where two function values are equal in the North-South or East-West directions). The number of pulses may vary from approximately 30,000 for a typical 300×300 image to over a hundred thousand for a 500×500 image. Since the decomposition produces such a large number of pulses, we need an efficient storage scheme to represent these in memory. Furthermore, we need to be able to calculate certain attributes of the pulses (such as the area and the boundary values) rapidly.

3.1. Storage

The storage scheme used is based on the popular Compressed Sparse Row (CSR) format, [7, 8], for representing sparse matrices. Using this scheme, the matrix

$$\begin{bmatrix} 5 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 0 & 9 & 0 \end{bmatrix}$$ is written as

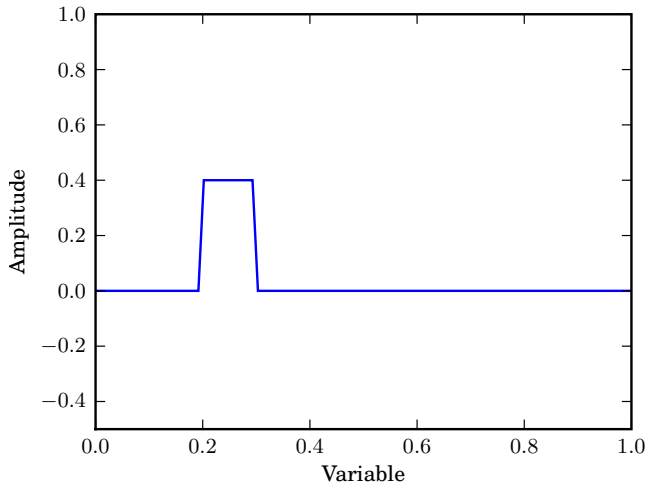
- `values` $= \begin{bmatrix} 5 & 1 & 2 & 3 & 6 & 9 \end{bmatrix}$
- `columns` $= \begin{bmatrix} 0 & 2 & 3 & 3 & 1 & 3 \end{bmatrix}$
- `row_offset` $= \begin{bmatrix} 0 & 3 & 4 & 4 & 6 \end{bmatrix}$

The values of the non-zero elements are stored in `values`, and their column-positions given by `columns`. The elements of `row_offset` specify where in `columns` the elements of each row start. The number of elements row j is given by `row_offset`$[j+1] -$ `row_offset`$[j]$.

When storing 2-dimensional pulses, we know that the pulse

- may only occupy a small portion of the image,
- has a single amplitude value across the pulse and
- consists of regions connected horizontally and vertically.

We therefore modify the storage structure, so that the pulse



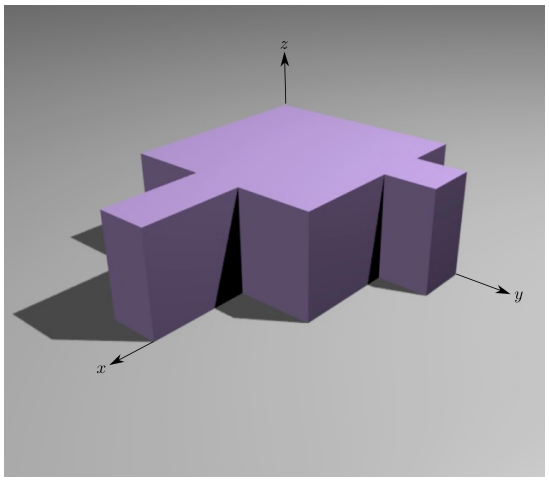


Figure 1: *Pulses in one and two dimensions to illustrate Definition 4. In the two-dimensional pulse, the amplitude is indicated by the z-axis, whereas the position is determined by x and y.*

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$ is written as

- `value` $= 1$
- `columns` $= \begin{bmatrix} 0 & 5 & 1 & 3 & 4 & 5 & 1 & 4 \end{bmatrix}$
- `start_row` $= 1$
- `row_offset` $= \begin{bmatrix} 0 & 2 & 6 & 8 \end{bmatrix}$

Instead of specifying column values, `columns` now indicates the start and end positions of the one-dimensional pulses that comprise the row. For example, the third row of the two-dimensional pulse above consists of two one-dimensional pulses: the first stretching from 1 up to (but excluding) 3, the other from 4 up to 5. This corresponds with the values seen in `columns`. The pulse may only cover a few rows of the entire image, therefore we use `start_row` to indicate the occurrence, saving us from storing every single row. The values of `row_offset`, as in the previous example, specify where in `columns` each new row starts.

An advantage of this storage scheme is that it can also be used to store connected regions, and we'll make use of this capability to initialise the algorithm as shown later.

3.2. Queries

Given a pulse in the above format, we'd like to calculate the following queries rapidly:

Area / number of non-zeros The area of the pulse is the sum of the lengths of the one-dimensional pulses comprising its rows. Each such length is given as the corresponding difference between the pulse start-end positions in `columns`. In the example above, the area would be $(5-0)+(3-1)+(5-4)+(4-1) = 5+2+1+3 = 11$.

Adjacent Set / Boundary positions Each pulse has four or more boundary positions – connected to the pulse in a 4-connected sense (see Fig. 2) – that form the adjacent set. To find the boundary positions, we follow a scanline approach, with three scanlines moving from the top of the pulse to the bottom (see Fig. 3). Here, we describe the operation once the scanlines have entered the pulse (in other words, neglecting top and bottom boundaries, which need to be handled separately):

1. The scanlines are centred around row j and are formed by constructing the pulse at rows $j-1$, j and $j+1$.

2. For each element of the central scanline that does *not* belong to the pulse, determine whether any of its neighbours (above, below, left or right) belong to the pulse. If they do, then that element lies on the boundary.

3. Move the scanlines one row down and repeat (it is only necessary to recalculate the bottom scanline at each step).

3.3. Operations

Merging Two Pulses Later on, when performing the Discrete Pulse Transform, we shall be required to merge

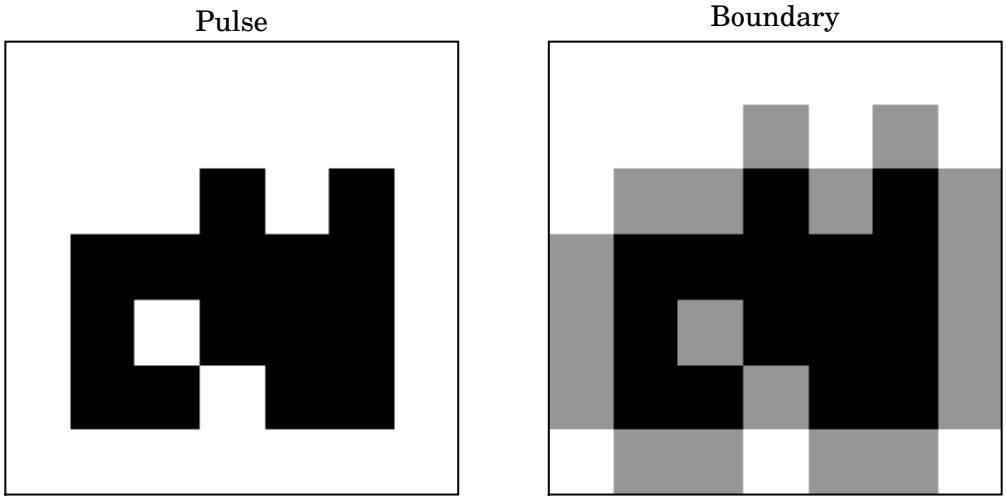


Figure 2: *Boundary positions of a pulse.*

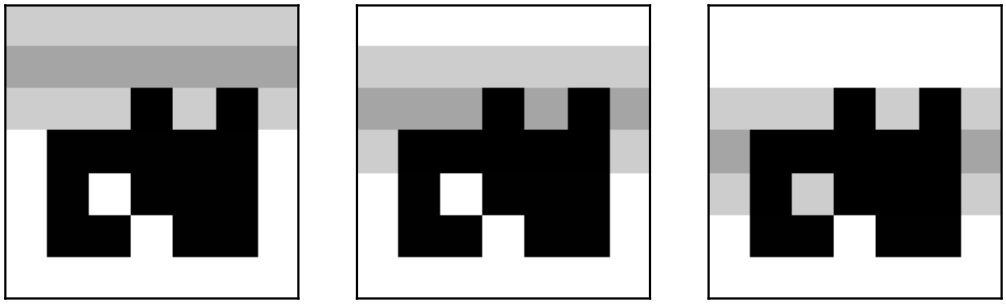


Figure 3: *Scanlines used to find boundary positions.*

two pulses that touch. This is done on a row-by-row basis. In the trivial case where a row is contained in only one of the two pulses, we simply include that row in the output. Otherwise, we need to sort and join the one-dimensional pulses that comprise the row carefully. Note, however, that these one-dimensional pulses cannot overlap in our problem description. We therefore:

1. Extract the stop-start intervals that form the one-dimensional pulses in row j.
2. Sort the intervals according to their starting position.
3. Step over the intervals and link (join) them if they touch.
4. Save the linked intervals as the representation of row j.
5. Repeat for row $j + 1$.

3.4. Algorithm Overview

Each step of the Discrete Pulse Transform is now described in more detail. We'll use the following terms:

Input image The input image or data – an $M \times N$ matrix of integer values between 0 and 255.

Label image An $M \times N$ array of integer values that indicate the connectivity of pixels in an image. If neighbouring pixels have the same value (i.e. are 4-connected), then they are assigned the same label value.

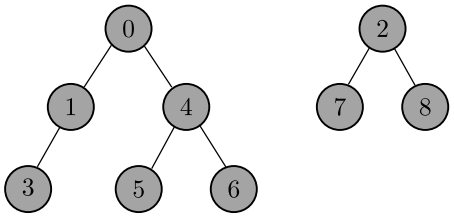


Figure 4: Two trees with labeled nodes.

Intermediate reconstruction An $M \times N$ image can be decomposed into pulses with areas ranging from 1 through MN . When summed, these pulses reconstruct the **input image**. It is also possible to only sum pulses with area $> k$. We call this an intermediate reconstruction, as it approximates the image up to a certain level only.

Finding Connection Regions First, we identify all 4-connected regions in the image (these are the initial pulses that are processed to yield the Discrete Pulse Transform). This is done using the Union-Find connected component algorithm of Fiorio and Gustedt [9], with the connectivity tree stored in an array as suggested by Wu et al. in [35]. It is shown in [35] that this algorithm executes in $\mathcal{O}(N)$, and we give a brief overview of its functioning:

Representing a tree using an array One or more trees consisting of N nodes can be stored in an array of length N. Examine the trees shown in Figure 4 with nodes labeled $n = 0, \ldots, 8$. These trees can be represented as the array

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 2 & 1 & 0 & 4 & 4 & 2 & 2 \end{bmatrix}$$

where $\mathbf{x}_n$ gives the parent of node n. For example, $x_3 = 1$, which tells us that the parent of node three is node one. Similarly, $x_2 = 2$ implies that node two has no parent—it is the root of a tree.

Labelling connected regions as trees The goal of the connected components algorithm is to assign unique labels to each connected region in an $M \times N$ image I. An array, L, of length MN is used to store trees as indicated in the paragraph above.

The image is traversed in raster scan order (i.e. along rows). A region counter, k, is initialised to zero. At each pixel position (r, c):

1. Calculate the offset into the tree array as $t = rN + c$.

2. If the pixel is not connected to (does not have the same value as) the pixel above it or to the left, assign $L_t = t$, effectively creating a new tree.

3. If the pixel is connected to the pixel above, assign $L_t = L_{t-M}$, joining node t to its parent in the previous row.

4. If, in addition, the pixel is connected to the left, assign $L_{t-1} = L_{t-M}$.

5. If the pixel is only connected to the left, assign $L_t = L_{t-1}$.

Appropriate care needs to be taken in the first row and column to prevent indexing errors on the image boundary.

The label vector, L, can also be seen as the flattened version of a *label image* so that $L_{r,c} = L_{rN+c}$. From this image, all connected regions are extracted as pulses and stored in the format discussed in Sectios 3.1.

We then proceed to perform the Discrete Pulse Transform as discussed next.

Identifying Pulses to Merge The Discrete Pulse Transform is performed by alternately executing the L_k (lower) and U_k (upper) operators that extract pulses of area k. If you think of the image as a height-map, then the U_1-operator removes all valleys of area one. Here, a valley is defined as a connected area that is surrounded only by higher values. Similarly, the L_1-operator removes peaks of area one, where peaks are connected areas surrounded only by lower values.

After applying the L_1- and U_1-operators and *storing the removed peaks and valleys* (those form the first level of the DPT), we need to merge pulses that were joined in the process. Note that, at each decomposition level, we have the *intermediate reconstruction* available. It is obtained by setting the image values corresponding to the removed positive (negative) pulses equal to the maximum (minimum) value on the adjacent set.

For each pulse, we calculate its boundary positions using the method described in Section 3.2. We then examine the boundary values on the *intermediate reconstruction*, and if any of those values are equal to the pulse value, a merge is required. After examining all boundary positions, a list is drawn up of all coordinates that fall on merge boundaries. At each of those positions, a merge is performed as described in Section 3.3, after which the *label image* is updated.

The L_{k+1} and U_{k+1} operators are now applied, repeating the merging process for higher values of k until the image has been entirely decomposed (in other words, until the final MN-sized pulse has been removed). All the removed pulses together from the Discrete Pulse Transform or decomposition.

3.5. Algorithm Optimisations

Pulses by Area For an $M \times N$ image, the discrete-pulse decomposition has pulses with areas ranging from $L = 0, \ldots, MN$. In practice, however, many values of

L have no corresponding pulses. When applying the L and U operators, time is saved by skipping these cases entirely. A list of pulses, sorted by area, is kept up to date whenever two regions are merged (due to pulses being constructed from small to large, when merging regions of size k we only need to update the list for areas $\geq k$).

3.6. Reproducibility and Code

The code used in these experiments is made available under the GNU General Public License and can be downloaded from `http://dip.sun.ac.za/~stefan/dpt`.

4. Application: Target Detection via Scale-Spaces

An image is made up of structures of varying sizes, or scales. The structures can be present at more than one scale with each representing information of differing importance. The number of pixels included in a connected region [30] defines its scale. Making use of scale-spaces or multi-resolution methods to analyze an image for feature detection allows the use of more information than the pixel luminosity only. See [22, 10, 20, 12, 24, 19, 14, 15, 16, 34, 32, 33] for a detailed history of scale-spaces. In the absence of prior knowledge on the feature characteristics and size we can still make use of scale-spaces for feature detection by keeping every scale. Reflecting on how a human eye picks out features in an image, the Human Vision System (HVS) Model [18] provides some insight. It consists of a first stage, the Pre-Attentive Stage, in which the features are detected and then a second stage, the Attentive Stage, in which matching takes place between the detected features of the first stage and the remaining image. It is clear that the HVS possesses a degree of saliency, to detect the 'pop-out' features. We will shortly show how the LULU operators can be used to detect theses 'pop-out' features. The most referenced and applied scale space is the Gaussian scale-space, but a drawback of the this scale-space is its linearity. It removes small scale features (noise) very well, but results in spatial distortions as scale increases, i.e. reduced sharpness of edges and shapes [22, 10]. To prevent this, a nonlinear smoothing step is introduced [23, 25, 17, 11, 5, 31] and we show that the LULU scale-space does the same.

The LULU scale-space satisfies the axioms of the Gaussian scale-space [1] but has the benefit of nonlinearity. This results in excellent shape and preservation properties, namely consistent separation, and total variation and shape preservation [1]. The Discrete Pulse Transform, $f = \sum_{n=1}^{N} D_n(f)$, forms a scale-space where the scaled image is $P_n(f)$ for discrete scales $n = 1, 2, 3, ..., N$. A second advantage of this LULU scale-space is therefore clear - it is already discrete and no approximations or sampling needs to be done, unlike with the Gaussian scale-space [3, 24, 19].

Following the HVS Model, the features are those areas in the image which are stable, that is, the areas which survive over a wide range of scales, [22]. This is simple to apply to the LULU scale-space. Indeed, each pixel belongs to a number of pulses, ϕ_{ns}, for $n \in N_0 \subseteq \{1, 2, 3, ..., N\}$. Here n represents the particular scale. For each pixel, we then have what we call a Discrete Pulse Vector (DPV),

$$\mathbf{p}_{x,y} = \begin{bmatrix} \ell_1 & \ell_2 & \ell_3 & \dots & \ell_k \end{bmatrix}^T$$

where for each scale n_i in N_0, we have the corresponding relative luminosity ℓ_i of the pulse $\phi_{n_i s}$, that is, the height or depth of the local maximum or minimum set. We then use these DPV's for feature detection by keeping only those pixels which belong to DPV's that contain a large number of scales, i.e. is spread across many values of k. This value is referred to as the scale-space lifetime at the pixel [24]. Figure 5 illustrates this method of feature detection. We apply ICM image segmentation [6] to the original image and to the lifetime image — that is, to the lifetime at each pixel. We note that the segmentation on the lifetime image better detects the trucks, given that the original image contains a lot of background scattering that hampers analysis. The scale space exposed by the DPT provides an ideal pre-processing step for efficient segmentation and object detection.

5. Conclusion

We have presented the implementation of the now well-developed theory for the LULU operators in higher dimensions. The extensive storage and processing required to obtain the DPT demands a carefully thought out algorithm, which we have provided. Owing to space restrictions, an extensive comparison to similar implementation techniques has not been discussed, and will be addressed in a future publication, as will performance evaluations.

A simple example of one application of the DPT is presented in Section 4, illustrating the strengthened segmentation achieved by making use of the DPT of the image. With the implementation of the DPT now in place, further applications can be developed.

6. References

[1] R Anguelov and I N Plaskitt (now Fabris-Rotelli), "A Class of LULU Operators on Multi-Dimensional Arrays", 2008, http://arxiv.org/PS_cache/arxiv/pdf/0712/0712.2923v1.pdf

[2] R Anguelov and I N Fabris-Rotelli, "Discrete Pulse Transform of Images", A. Elmoataz et al. (Eds.): ICISP 2008, LNCS 5099:1–9, 2008.

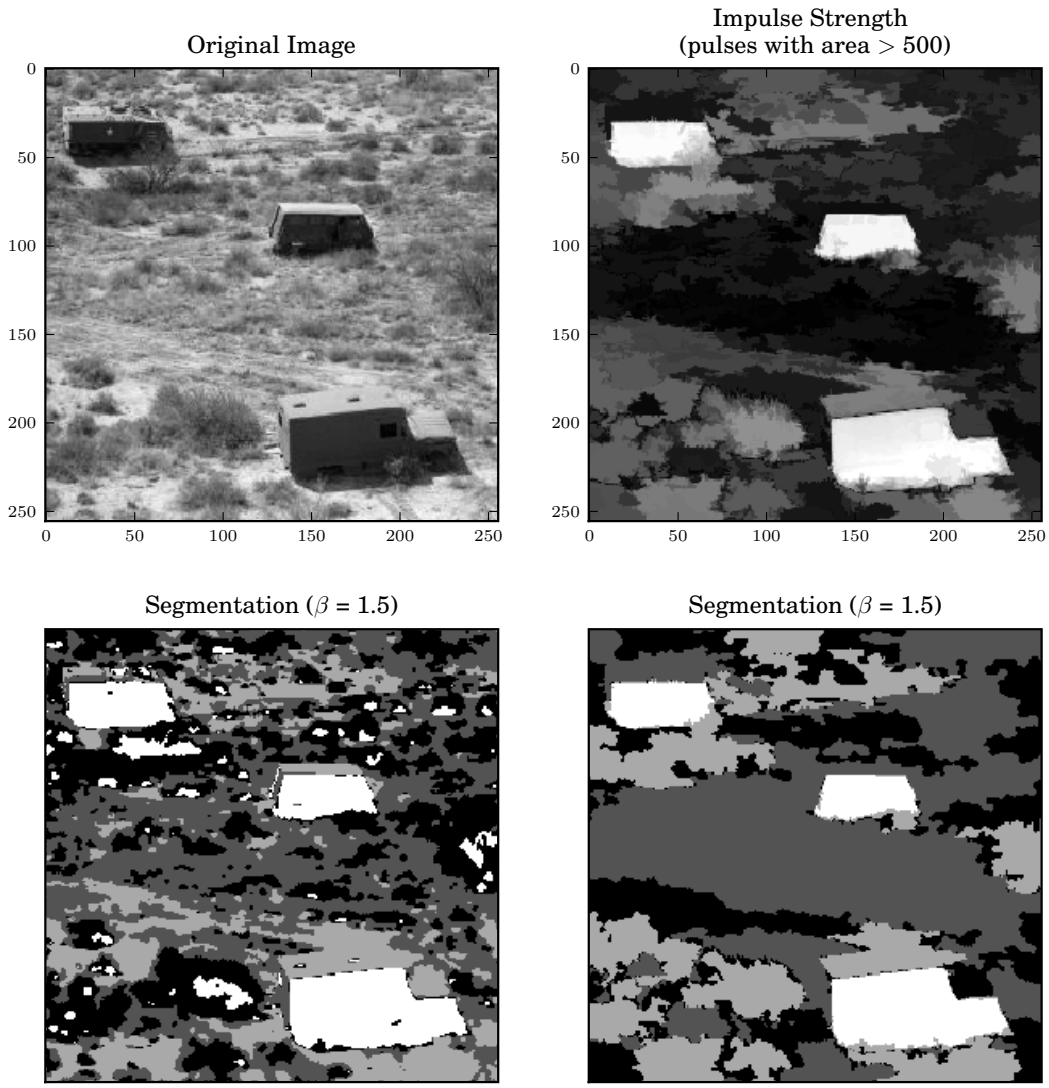


Figure 5: *Iterated Conditional Modes segmentation (seen in the second row) applied to an image as well as to its impulse strength (seen in the first row). The ICM smoothing parameter, β, is set to 1.5 as suggested in the literature* [4]. *By making use of the DPT of the image and keeping only pulses with area larger than 500, better segmentation is achieved. Compare the two images in the seocnd row.*

[3] J Babaud, A P Witkin and M Baudin, "Uniqueness of the Gaussian Kernel for Scale-Space Filtering", IEEE Transactions On Pattern Analysis And Machine Intelligence, **8**(1), 1986.

[4] J Besag, "On the Statistical Analysis of Dirty Pictures", Royal Statistical Society B, **48**(3), (1986) 259.

[5] R W Brockett and P Maragos, "Evolution Equations for Continuous-Scale Morphological Filtering", IEEE Transactions on Image Processing, **42**(12), 1994.

[6] P Debba, A Stein, F D van der Meer, E J M Carranza and A Lucieer, "The Optimal Field Sampling Scheme on a Remote Sensing Segmented Image", ICCSA 2008, Part I, LNCS **5072** (2008) 756-768.

[7] I. Duff, R. Grimes and J. Lewis, "Sparse Matrix Test Problems", ACM Trans. Math. Soft., **15**:1–14, 1989.

[8] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. Van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition, SIAM, 1994.

[9] C. Fiorio and J. Gustedt, "Two Linear Time Union-Find Strategies for Image Processing", Theor. Comput. Sci. **154**(2):165–181, 1996.

[10] L Florack and A Kuijper, "The Topological Structure of Scale-Space Images", Journal of Mathematical Imaging and Vision **12:**65–79, 2000.

[11] L Florack, R Maas and W Niessen, "Pseudo-linear Scale-Space Theory", International Journal of Computer Vision **31**(2/3):247–259, 1999.

[12] L Florack, B Romeny, J Koenderink and M Viergever, "Scale and the Differential Structure of Images", Image and Vision Computing, **10**(6), 1992.

[13] T Iijima, "Basic Theory of Pattern Observation", Papers of Technical Group on Automata and Automatic Control, IECE, Japan, December 1959 (in Japanese).

[14] T Iijima, "Basic Theory on Normalization of Pattern (in Case of Typical One-Dimensional Pattern)", Bulletin of the Electrotechnical Laboratory, **26:**368–388, 1962 (in Japanese).

[15] T Iijima, "Observation Theory of Two-Dimensional Visual Patterns", Papers of Technical Group on Automata and Automatic Control, IECE, Japan, Oct. 1962 (in Japanese).

[16] T Iijima, "Basic Theory on Normalization of Two-Dimensional Visual Pattern", Studies on Information and Control, Pattern Recognition Issue, IECE, Japan, **1:**15–22, 1963 (in Japanese).

[17] P T Jackway, "Morphological Scale-Space", Proceedings of 11th IAPR International Conference on Image, Speech and Signal Analysis, Pattern Recognition, **III**, 1992.

[18] T Kadir and M Brady, "Saliency, Scale and Image Description", International Journal of Computer Vision **45**(2):83–105, 2001.

[19] K Koenderink, "The Structure of Images", Biological Cybernetics, **50:**363-370, 1984.

[20] A Kuijper, L Florack and M Viergever, "Scale-Space Hierarchy", Journal of Mathematical Imaging and Vision **18**:169–189, 2003.

[21] D P Laurie and C H Rohwer, "The Discrete Pulse Transform", SIAM J. Math. Anal., **38**(3), 2007.

[22] T Lindeberg and K V Mardia, "Scale-Space Theory: A Basic Tool for Analyzing Structures at Different Scales", Journal of Applied Statistics, **21**(1/2):225–271, 1994.

[23] T Lindeberg, "Scale-Space: A Framework for Handling Image Structures at Multiple Scales", in Proceedings of CERN School of Computing, Egmond aan Zee, The Netherlands, 8–21 September, 1996.

[24] T Lindeberg, "Scale-Space for Discrete Signals", IEEE Transactions on Pattern Analysis and Machine Intelligence, **12:**234-254, 1990.

[25] P Perona and J Malik, "Scale Space and Edge Detection using Anisotropic Diffusion", IEEE Trans. Pattern Anal. Mach. Intell., **12:**629–663, 1990.

[26] C H Rohwer, "Variation Reduction and LULU-Smoothing", Quaestiones Mathematicae **25:**163–176, 2002.

[27] C H Rohwer, "Fully Trend Preserving Operators", Quaestiones Mathematicae **27:**217–230, 2004.

[28] C H Rohwer, Nonlinear Smoothers and Multiresolution Analysis, Birkhäuser, 2005.

[29] C H Rohwer and M Wild, "LULU Theory, Idempotent Stack Filters, and the Mathematics of Vision of Marr", Advances in imaging and electron physics, **146:**57–162, 2007.

[30] J Serra, "A Lattice Approach to Image Segmentation", Journal of Mathematical Imaging and Vision, **24:**83–130, 2006.

[31] R van den Boomgaard and A Smeulders, “The Morphological Structure of Images: The Differential Equations of Morphological Scale-Space”, IEEE Transactions on Pattern Analysis and Machine Intelligence, **16**(11), 1994.

[32] J Weickert, S Ishikawa and A Imiya, “Linear Scale-Space has Fisrt Been Proposed in Japan”, Journal of Mathematical Imaging and Vision, **10:**237–252, 1999.

[33] J Weickert, S Ishikawa and A Imiya, “On the History of Gaussian Scale-Space Axiomatics”, Computational Imaging and Vision, **8:**45–49, 1994.

[34] A P Witkin, “Scale-Space Filtering”, in Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI ’83, Karlsruhe, Aug. 8–12, 1983), **2** (1983) 1019–1022.

[35] K Wu, E Otoo and A Shoshani, “Optimizing Connected Component Labeling Algorithms”, 2005, http://repositories.cdlib.org/lbnl/LBNL-56864.