Chapter 3

The Discrete Pulse Transform

3.1 The Discrete Pulse Transform

The Discrete Pulse Transform based on the LULU operators for sequences was derived in [183] [113] [184]. Using the extension of the LULU operators to functions on $\mathbb{Z}^d$ we present the DPT for functions in $\mathcal{A}(\mathbb{Z}^d)$. Similar to the case of sequences we obtain a decomposition of a function $f \in \mathcal{A}(\mathbb{Z}^d)$, with finite support. As usual $\text{supp}(f) = \{p \in \mathbb{Z}^d : f(p) \neq 0\}$. Let $N = \text{card}(\text{supp}(f))$. We derive the DPT of $f \in \mathcal{A}(\mathbb{Z}^d)$ by applying iteratively the operators L_n, U_n with n increasing from 1 to N as follows

$$DPT(f) = (D_1(f), D_2(f), ..., D_N(f)), \tag{3.1}$$

where the components of (3.1) are obtained through

$$D_1(f) = (id - P_1)(f) \tag{3.2}$$
$$D_n(f) = (id - P_n) \circ Q_{n-1}(f),\ n = 2, ..., N, \tag{3.3}$$

and $P_n = L_n \circ U_n$ or $P_n = U_n \circ L_n$ and $Q_n = P_n \circ ... \circ P_1$, $n \in \mathbb{N}$. We will show that this decomposition has the property that each component D_n in (3.1) is a sum of discrete pulses with pairwise disjoint supports of size n, where in this setting a discrete pulse is defined as follows.

Definition 15 *A function $\phi \in \mathcal{A}(\mathbb{Z}^d)$ is called a pulse if there exists a connected set V and a nonzero real number α such that*

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

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

The concept of a pulse as defined in Definition 15 is similar to the idea of a flat zone from mathematical morphology. It should be remarked that the support of a pulse may generally have any shape, the only restriction being that it is connected. Note that the smoothing process ultimately results in the last component $D_N(f)$ being a constant image, that is, one pulse the size of the entire image, and the remaining image component is $Q_N(f) = 0$.

It follows from (3.2) and (3.3) that

$$f = \sum_{n=1}^{N} D_n(f). \tag{3.4}$$

The usefulness of the representation (3.4) of a function $f \in \mathcal{A}(\mathbb{Z}^d)$ is in the fact that all terms are sums of pulses as stated in the sequel. First we need to state a technical lemma.

Lemma 16 *Let $f \in \mathcal{A}(\mathbb{Z}^d)$, $\operatorname{supp}(f) < \infty$, be such that f does not have local minimum sets or local maximum sets of size smaller than n, for some $n \in \mathbb{N}$. Then we have the following two results.*

a)

$$(id - P_n)f = \sum_{i=1}^{\gamma^-(n)} \phi_{ni} + \sum_{j=1}^{\gamma^+(n)} \varphi_{nj}, \tag{3.5}$$

where $V_{ni} = \operatorname{supp}(\phi_{ni}), i = 1, 2, ..., \gamma^-(n)$, are local minimum sets of f of size n, $W_{nj} = \operatorname{supp}(\varphi_{nj}), j = 1, 2, ..., \gamma^+(n)$, are local maximum sets of f of size n, ϕ_{ni} and φ_{nj} are negative and positive discrete pulses respectively, and we also have that

- $V_{ni} \cap V_{nj} = \emptyset$ *and* $\operatorname{adj}(V_{ni}) \cap V_{nj} = \emptyset$,
  $$i, j = 1, ..., \gamma^-(n),\ i \neq j, \tag{3.6}$$
- $W_{ni} \cap W_{nj} = \emptyset$ *and* $\operatorname{adj}(W_{ni}) \cap W_{nj} = \emptyset$,
  $$i, j = 1, ..., \gamma^+(n), i \neq j, \tag{3.7}$$
- $V_{ni} \cap W_{nj} = \emptyset$
  $$i = 1, ..., \gamma^-(n)\ , j = 1, ..., \gamma^+(n). \tag{3.8}$$

b) For every fully trend preserving operator A

$$U_n(id - AU_n) = U_n - AU_n,$$
$$L_n(id - AL_n) = L_n - AL_n.$$

Proof
a) Let $V_{n1}, V_{n2}, ..., V_{n\gamma^-(n)}$ be all local minimum sets of size n of the function f. Since f does not have local minimum sets of size smaller than n, then f is a constant on each of these sets, by Theorem 8. Hence, the sets are disjoint, that is $V_{ni} \cap V_{nj} = \emptyset$, $i \neq j$. Moreover, we also have

$$\text{adj}(V_{ni}) \cap V_{nj} = \emptyset,\ \ i,j = 1, ..., \gamma^-(n). \tag{3.9}$$

Indeed, let $x \in \text{adj}(V_{ni}) \cap V_{nj}$. Then there exists $y \in V_{ni}$ such that $(x,y) \in r$. Hence $y \in V_{ni} \cap \text{adj}(V_{nj})$. From the local minimality of the sets V_{ni} and V_{nj} we obtain respectively $f(y) < f(x)$ and $f(x) < f(y)$, which is clearly a contradiction. For every $i = 1, ..., \gamma^-(n)$ denote by y_{ni} the point in $\text{adj}(V_{ni})$ such that

$$f(y_{ni}) = \min_{y \in \text{adj}(V_{ni})} f(y). \tag{3.10}$$

Then we have

$$U_n f(x) = \begin{cases} f(y_{ni}) & \text{if } x \in V_{ni},\ i = 1, ..., \gamma^-(n) \\ \\ f(x) & \text{otherwise (by Theorem 9)} \end{cases}$$

Therefore

$$(id - U_n)f = \sum_{i=1}^{\gamma^-(n)} \phi_{ni} \tag{3.11}$$

where ϕ_{ni} is a discrete pulse with support V_{ni} and negative value (down pulse).

Let $W_{n1}, W_{n2}, ..., W_{n\gamma^+(n)}$ be all local maximum sets of size n of the function $U_n f$. By [8, Theorem 12(b)] every local maximum set of $U_n f$ contains a local maximum set of f. Since f does not have local maximum sets of size smaller than n, this means that the sets W_{nj}, $j = 1, ..., \gamma^+(n)$, are all local maximum sets of f and f is constant on each of them. Similarly to the local minimum sets of f considered above we have $W_{ni} \cap W_{nj} = \emptyset$, $i \neq j$, and $\text{adj}(W_{ni}) \cap W_{nj} = \emptyset$, $i,j = 1, ..., \gamma^+(n)$. Moreover, since $U_n(f)$ is constant on any of the sets $V_{ni} \cup \{y_{ni}\}$, $i = 1, ..., \gamma^-(n)$, see Theorem 8, we also have

$$(V_{ni} \cup \{y_{ni}\}) \cap W_{nj} = \emptyset,\ \ i = 1, ..., \gamma^-(n),\ \ j = 1, ..., \gamma^+(n), \tag{3.12}$$

which implies (3.8).

Further we have

$$L_nU_nf(x) = \begin{cases} U_nf(z_{nj}) & \text{if } x \in W_{nj},\ j = 1, ..., \gamma^+(n) \\ U_nf(x) & \text{otherwise} \end{cases}$$

where $z_{nj} \in \text{adj}(W_{nj})$, $j = 1, ..., \gamma^+(n)$, are such that $U_nf(z_{nj}) = \max\limits_{z\in\text{adj}(W_{nj})} U_nf(z)$. Hence

$$(id - L_n)U_nf = \sum_{j=1}^{\gamma^+(n)} \varphi_{nj} \tag{3.13}$$

where φ_{nj} is a discrete pulse with support W_{nj} and positive value (up pulse). Thus we have shown that

$$(id - P_n)f = (id - U_n)f + (id - L_n)U_nf = \sum_{i=1}^{\gamma^-(n)} \phi_{ni} + \sum_{j=1}^{\gamma^+(n)} \varphi_{nj}.$$

b) Let the function $f \in \mathcal{A}(\mathbb{Z}^d)$ be such that it does not have any local minimum or local maximum sets of size less than n. Denote $g = (id - AU_n)(f)$. We have

$$g = (id - AU_n)(f) = (id - U_n)(f) + ((id - A)U_n)(f). \tag{3.14}$$

As in (a) we have that (3.11) holds, that is we have

$$(id - U_n)(f) = \sum_{i=1}^{\gamma^-(n)} \phi_{ni}, \tag{3.15}$$

where the sets $V_{ni} = \text{supp}(\phi_{ni})$, $i = 1, ..., \gamma^-(n)$, are all the local minimum sets of f of size n and satisfy (3.6). Therefore

$$g = \sum_{i=1}^{\gamma^-(n)} \phi_{ni} + ((id - A)U_n)(f). \tag{3.16}$$

Furthermore,

$$U_n(f)(x) = \begin{cases} f(x) & \text{if} \quad x \in \mathbb{Z}^d \setminus \bigcup\limits_{i=1}^{\gamma^-(n)} V_{ni} \\ v_i & \text{if} \quad x \in V_{ni} \cup \{y_{ni}\},\ i = 1, ..., \gamma^-(n), \end{cases}$$

where $v_i = f(y_{ni}) = \min\limits_{y\in \mathrm{adj}(V_{ni})} f(y)$. Using that A is fully trend preserving, for every $i = 1, ..., \gamma^-(n)$ there exists w_i such that $((id - A)U_n)(f)(x) = w_i$, $x \in V_{ni} \cup \{y_{ni}\}$. Moreover, using that every adjacent point has a neighbor in V_{ni} we have that $\min\limits_{y\in \mathrm{adj}(V_{ni})} ((id - A)U_n)(f)(y) = w_i$. Considering that the value of the pulse ϕ_{ni} is negative, we obtain through the representation (3.16) that V_{ni}, $i = 1, ..., \gamma^-(n)$, are local minimum sets of g.

Next we show that g does not have any other local minimum sets of size n or less. Indeed, assume that V_0 is a local minimum set of g such that $\mathrm{card}(V_0) \le n$. Since $V_0 \cup \mathrm{adj}(V_0) \subset \mathbb{Z}^d \setminus \bigcup\limits_{i=1}^{\gamma^-(n)} V_{ni}$ it follows from (3.16) that V_0 is a local minimum set of $((id - A)U_n)(f)$. Then using that $(id - A)$ is neighbor trend preserving and using [8, Theorem 17] we obtain that there exists a local minimum set W_0 of $U_n(f)$ such that $W_0 \subseteq V_0$. Then applying again [8, Theorem 17] or [8, Theorem 12] we obtain that there exists a local minimum set $\tilde{W}_0$ of f such that $\tilde{W}_0 \subseteq W_0 \subseteq V_0$. This inclusion implies that $\mathrm{card}(\tilde{W}_0) \le n$. Given that f does not have local minimum sets of size less than n we have $\mathrm{card}(\tilde{W}_0) = n$, that is $\tilde{W}_0$ is one of the sets V_{ni} - a contradiction. Therefore, V_{ni}, $i = 1, ..., \gamma^-(n)$, are all the local minimum sets of g of size n or less. Then using again (3.11) we have

$$(id - U_n)(g) = \sum_{i=1}^{\gamma^-(n)} \phi_{ni} \tag{3.17}$$

Using (3.15) and (3.17) we obtain

$$(id - U_n)(g) = (id - U_n)(f)$$

Therefore

$$\begin{aligned} (U_n(id - AU_n))(f) &= U_n(g) = g - (id - U_n)(f) \\ &= (id - AU_n)(f) - (id - U_n)(f) \\ &= (U_n - AU_n)(f). \end{aligned}$$

This proves the first identity. The second one is proved in a similar manner.
■

Theorem 17 *Let $f \in \mathcal{A}(\mathbb{Z}^d)$. For every $n \in \mathbb{N}$ the function $D_n(f)$ derived through (3.2) and (3.3) is a sum of discrete pulses with pairwise disjoint* *support, that is, there exist $\gamma(n) \in \mathbb{N}$ and discrete pulses ψ_{ns}, $s = 1, ..., \gamma(n)$, such that*

$$D_n(f) = \sum_{s=1}^{\gamma(n)} \psi_{ns}, \tag{3.18}$$

and

$$\mathrm{supp}(\psi_{ns_1}) \cap \mathrm{supp}(\psi_{ns_2}) = \emptyset \text{ for } s_1 \neq s_2. \tag{3.19}$$

Moreover, if $n_1, n_2, s_1, s_2 \in \mathbb{N}$ are such that $n_1 < n_2$, $1 \leq s_1 \leq \gamma(n_1)$ and $1 \leq s_2 \leq \gamma(n_2)$. Then

$$\mathrm{supp}(\psi_{n_1s_1}) \cap \mathrm{supp}(\psi_{n_2s_2}) \neq \emptyset \Longrightarrow \mathrm{supp}(\psi_{n_1s_1}) \subset \mathrm{supp}(\psi_{n_2s_2}) \tag{3.20}$$

Proof
According to (3.3), D_n is obtained by applying $id - P_n$ to the function $Q_{n-1}(f)$ which, by the results in Chapter 1, does not have local maximum or minimum sets of size less than n. Thus by Lemma 16(a) we have that $D_n(f) = (id - P_n)Q_{n-1}(f)$ is a sum of pairwise disjoint discrete pulses as given in (3.5). More precisely,

$$D_n(f) = \sum_{s=1}^{\gamma(n)} \psi_{ns} = \sum_{i=1}^{\gamma^-(n)} \phi_{ni} + \sum_{j=1}^{\gamma^+(n)} \varphi_{nj},$$

where $\gamma(n) = \gamma^-(n) + \gamma^+(n)$. Property (3.19) follows from (3.6)–(3.8).

Let $\mathrm{supp}(\psi_{n_1s_1}) \cap \mathrm{supp}(\psi_{n_2s_2}) \neq \emptyset$. It follows from the construction of (3.18) derived above that the functions $Q_n(f)$ and $L_{n+1}(Q_n(f))$, $n \geq n_1$, are constants on the set $\mathrm{supp}(\psi_{n_1s_1})$. Furthermore, the set $\mathrm{supp}(\psi_{n_2s_2})$ is a local maximum set of $Q_{n_2-1}(f)$ or a local minimum set of $L_{n_2}(Q_{n_2-1}(f))$. From the definition of local maximum set and local minimum set it follows that $\mathrm{supp}(\psi_{n_1s_1}) \subset \mathrm{supp}(\psi_{n_2s_2})$. ■

Using Theorem 17, the identity (3.4) can be written in the form

$$f = \sum_{n=1}^{N} \sum_{s=1}^{\gamma(n)} \psi_{ns}. \tag{3.21}$$

The equality above is the discrete pulse decomposition of f, where the pulses ψ_{ns} have the properties (3.19) and (3.20).

3.2 Connectivity

Besides the formal connectivity definition presented in [207] and Definition 3, various examples have been investigated to deal with certain applications as well as specific problems encountered when using standard connectivity. (See [186] for a good summary on connectivity.) The most active researchers in the development of connectivity have been Serra [207, 214, 215, 42, 200, 208, 209, 210, 187, 212] and Braga-Neto [22, 23, 25, 24, 26, 20, 21, 27], as well as their collaborators.

The usefulness of Definition 3 in image processing arises from a further result of Serra's, namely, that the sets of a connection $\mathcal{C}$ of subsets of B is equivalent to the family of openings $\{\gamma_x : x \in B\}$ such that (i) $\forall\ x \in B, \gamma_x(x) = \{x\}$, (ii) $\forall\ A \subset B, x, y \in B, \gamma_x(A)$ and $\gamma_y(A)$ are equal or disjoint, and (iii) $\forall\ A \subset B, \forall\ x \in B, x \notin A \Rightarrow \gamma_x(A)\emptyset$. The class

$$\mathcal{C} = \{\gamma_x(A) : x \in B, A \subset B\}$$

is a connection such that each A is partitioned into the smallest possible number of components belonging to the class $\mathcal{C}$ and if $A_1 \subset A_2$ then any connected component of A_1 is included in a connected component of A_2. Note that as mentioned in [25] if one has a means of extracting connected components, we have unambiguously defined a connection and vice versa. We discuss variations of Serra's connectivity.

λ-connectivity

Morphological filters by reconstruction have a leakage problem, that is two connected blobs connected by a thin pixel-sized filament is deemed connected although they should more realistically be considered two separate connected components. Serra [212] introduced a new underlying lattice called the viscous lattice, obtained as all the dilated (with a circular structuring element with radius λ) subsets of B instead of simply the subsets, namely

$$\mathcal{L} = \{\delta_\lambda(A), A \subset B, \lambda > 0\}.$$

Santillán et al [203] define the lattice

$$\mathcal{L}_\lambda = \{\delta_\lambda(A), A \subset B\}$$

for a specific viscosity λ and the connected components of $\mathcal{L}_\lambda$ as λ-connected components. It requires that a connected set be made up of at the very least continuous paths made by disks with radius λ whose centers move along the path.

Pseudo-connectivity

To combat the leakage problem pseudo-connections are introduced in [198] to improve segmentation. The minima of the negative distance function, namely $-\text{dist}(f)$, are ultimate erosions obtained via the watershed transform. Instead a thresholded version of the distance function is used, $\mathcal{D}_\ell(f) = -\left(\text{dist}(f) \bigwedge \ell\right)$. The case $\ell = 0$ corresponds to the classical connection by an erosion of size ℓ and $\ell = \infty$ corresponds to the case of ultimate erosions. Choosing a value of $\ell > 0$ results in a pseudo-connectivity. This solves the leakage problem but does introduce false contours for larger values of ℓ, so to combat this instead of applying the watershed transform to $-\text{dist}(f)$ it is applied to the distance transform of the closing by reconstruction which fills the holes which cause the false contours.

Hyperconnectivity Filters

In his 2009 PhD thesis, Ouzounis [162] describes hyperconnectivity introduced by Serra [208] for the improvement of attribute filters due to their leakage and overlap issues. Hyperconnectivity is a relaxation of the connection defined in Definition 3.

Definition 18 *Operator* $\perp$: $\mathcal{P}(\mathcal{P}(B)) \mapsto \{0,1\}$ *is an* overlap criterion *such that* $\perp$ *is decreasing* $A_1 \subset A_2 \Rightarrow \perp(A_1) \leq \perp(A_2)$ *and* $\perp(A) = 1$ *means that* A *is overlapping and non-overlapping otherwise.*

Definition 19 *Let* B *be an arbitrary non-empty set. A family* $\mathcal{H}$ *of subsets of* B *is called a hyperconnected class or a* hyperconnection *on* B *if*
i) $\emptyset \in \mathcal{H}$
(ii) $\{x\} \in \mathcal{H}$ *for all* $x \in B$
(iii) for any family $\{C_i\} \subseteq \mathcal{H}$ *for which* $\perp(\{C_i\}) = 1$ *we have* $\bigcup_{i \in I} C_i \in \mathcal{H}$.
If a set C *belongs to a hyperconnection* $\mathcal{H}$ *then* C *is called* hyperconnected.

All connectivity classes are special cases of hyperconnectivity with

$$\perp(\{C_i\}) = \begin{cases} 1 & \text{if } \bigcap C_i \neq \emptyset \\ 0 & \text{otherwise} \end{cases}.$$

Constrained Connectivity

Soille [217] restricts Definition 3 by requiring that the intensity value difference between neighbouring pixels in a connected set do not exceed a specified value α. He calls this constrained connectivity α-connectivity and based on this develops a connectivity index for each pixel.

Jump Connection

Serra [211] defines a *jump connection* which provides good segmentations in that the are fewer point zones (regions of only a few pixels), visually significant clusters and can be computed fast. The connected components

$$A(m) = \{x : x \in B, 0 < f(x) - m \leq k\}$$

where m if the minimum of f form a connection.

Multiscale Connectivity

As Braga-Neto and Goutsias [24, 22] state, important information is not confined to a single scale, but rather is spread out over several scales. The connectivity should also thus depend on the scale. Let Σ be the set of possible scales. We provide a brief definition of multiscale connectivity.

Definition 20 *For a connectivity measure ρ defining the degree of connectivity of a set A on a lattice, the set A is* σ-connected *if $\rho(A) \geq \sigma$ for $\sigma \in \Sigma$. A set is then* fully connected *if it is σ-connected for all $\sigma \in \Sigma$ or* fully disconnected *otherwise.*

In [27] Braga-Neto and Goutsias discuss three techniques for constructing multiscale connectivities, namely, pyramids of clustering, granulometries and clustering of openings. In [20] they apply multiscale connectivity to determine a scale-space representation for automatic target detection. They also describe connectivity for greyscale images [26]. A greyscale image is connected if all level sets below a pre-specified threshold are connected, namely level-k connectivity requires all level sets at level k or below are non-empty and connected. They investigate applications in object extraction, segmentation, object-based filtering and hierarchical image representations.

Other Connections

Ronse and Serra [187] define geodesic connectivity. Braga-Neto and Goutsias [25] elaborate on fuzzy connectivity first introduced by Rosenfeld [188, 189, 190].

Connected Operators

Salembier and Wilkinson [201] describe how filters based on shapes not related to the input result in distortion and thus must be adapted to local structures as well as connected operators made use of. Connected operators preserve edge, and thus shape, information in an image [26, 42].

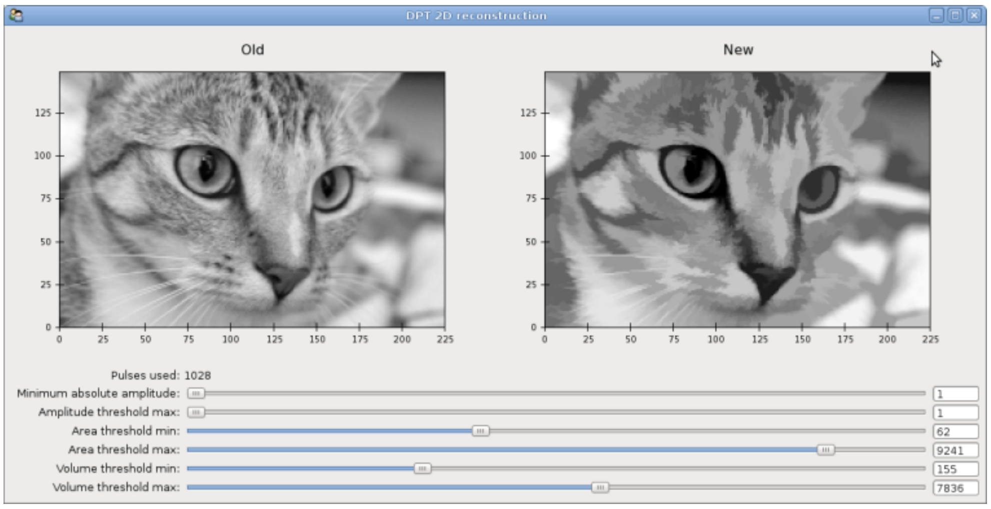
Definition 21 *An operator P is* connected *[26] if for any $f \in \mathcal{A}(\Omega)$ the partition $z_{P(f)}$ is coarser than the partition z_f, that is, $z_f(x) \subseteq z_{P(f)}(x) \; \forall \; x \in \Omega$, or equivalently $P(f)$ is constant over any flat zone of f. Here, $z_f(x) = \gamma_x(f(x)), x \in \Omega$ is the connectivity opening associated with the connection $\mathcal{C}$ discussed at the beginning of the section and a flat zone is that largest connected region where the value of f is constant.*

Connected operators have been widely made use of due to their preservation properties. Salembier et al [199, 197] state that the first reported connected operators were the binary openings by reconstruction done in 1976 and generalized to gray-level functions by reconstruction in 1993. They mention other connected operators as the λ-max operator, the area opening, dynamic filtering, volumic operator, complexity operator, motion operator, and moment-oriented operators which all preserve contours. Jones [102] introduces non-flat gray-level connected filters. Crespo and Schafer [41] specify two constraints for connected operators which morphological operators do satisfy, but the median operator, for example, does not.

Wilkinson and Ouzounis [244] state the obvious about standard morphological connected operators, that is, each type of structuring element has a limited power to represent an image due to its specific shape. The LULU operators act like the morphological area operators in that the shape is not specified, only the size, thus allowing far more flexibility. The fact that the LULU operators are connected according to Definition 21 adds further strength to their capabilities. We discuss these capabilities later in this chapter.

3.3 LULU Implementation

In [52] a `MATLAB` implementation of the DPT was presented. The algorithm used was by no means optimal nor real time. Since then collaboration with



Pulses used: 1028
Minimum absolute amplitude: 1
Amplitude threshold max: 1
Area threshold min: 62
Area threshold max: 9241
Volume threshold min: 155
Volume threshold max: 7836

Figure 3.1: *Snapshot of the DPT User Interface*

Dr S van der Walt from Stellenbosch University has resulted in a `Python` implementation of the DPT in two dimensions (for image analysis) which provides a user interface, see Figure 3.1, with which to choose scale levels and other parameters and view the transformed image, in addition to significantly faster processing times [55, 231]. We discuss this implementation now which is collaborative work presented in [55].

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. In two dimensions, a pulse describes a connected region over which function values are constant (for simplicity, we restrict ourselves here to 4-connectivity - 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.

3.3.1 Storage

The storage scheme used is based on the popular Compressed Sparse Row (CSR) format, [49, 14], 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` $= \begin{bmatrix} 0 & 3 & 4 & 5 \end{bmatrix}$

.

The values of the non-zero elements are stored in `values`, and their column-positions given by `columns`. Each entry of `row_offset` specifies an offset into `columns`, indicating the starting position of a new row. In the example above, we see that the second row (second element of `row_offset`) starts at position 3 of `columns`. The number of elements in 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 value across the pulse and consists of regions connected horizontally or vertically. We therefore modify the storage structure, so that the pulse

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

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 past-end indices of the one-dimensional pulses that comprise the rows. The values of `row_offset`, as in the previous example, specify where in `columns` each new row starts. The pulse may only cover a few rows of the entire image, therefore we use `start_row` to indicate the first occurrence, saving us from storing every single row.

As an example, consider the third row of the two-dimensional pulse above, which consists of two one-dimensional pulses: the first stretching from column 1 up to (but excluding) 3, the other from 4 up to 5. Since we are inter-

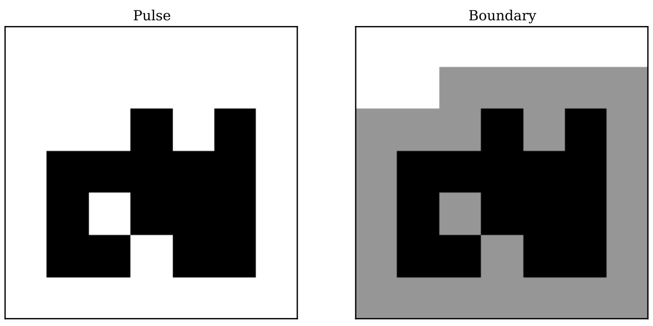


Figure 3.2: *Boundary positions of a pulse.*

ested in the third row (row number 2), and we only start recording rows at `start_row` $= 1$, we find the corresponding column indices in `row_offset`$[2-1] = 2$. At position 2, `columns` contains 1, 3 and 4, 5 as expected.

An advantage of this storage scheme is that it can also be used to store connected regions, a capability we exploit later to initialize the algorithm.

3.3.2 Queries

Given a pulse in the above format, we'd like to compute 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 3.2) – 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. 3.3). 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):

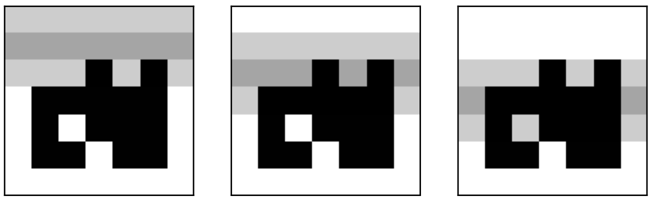


Figure 3.3: *Scanlines used to find boundary positions.*

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).

3.3.3 Operations

Merging Two Pulses Later on, when performing the Discrete Pulse Transform, we shall be required to merge 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 them if they touch.
4. Save the linked intervals as the representation of row j.
5. Proceed to row $j+1$ and repeat.

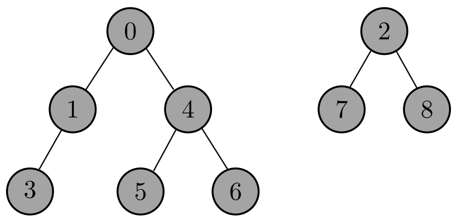


Figure 3.4: Two trees with labeled nodes.

3.3.4 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.

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), a pre-processing step also suggested by Lindeberg [123, Chapter 9.1] for comparing properties of constant grey-level regions. Our implementation uses the the Union-Find connected component algorithm of Fiorio and Gustedt [60], with the connectivity tree stored in an array as suggested by Wu et al. in [248]. It is shown in [248] that this algorithm executes in an optimal $\mathcal{O}(N)$, 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 3.4 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.

Labeling 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 initialized 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 $L_t = L_{t-N}$, joining node t to its parent in the previous row.
4. If, in addition, the pixel is connected to the left, assign $L_{t-1} = L_{t-N}$.
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 Section 3.3.1. 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 on pulses of area k. Thinking of the image as a height-map, 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 3.3.2. 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 3.3.3, after which the *label image* is updated. The L_{k+1} and U_{k+1} operators are now repeatedly applied, 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.

3.3.5 Algorithm Optimizations

Area Histogram 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. We can track these cases by constructing a histogram, $H[k]$, of the pulse sizes during the initial connected component search. Thereafter, whenever merging two regions, the histogram is updated. Then, when U_k or L_k is executed, we simply verify that $H[k] > 0$ before proceeding.

A Benchmark of Accidental Recombinations The decomposition was implemented and executed on a Intel Core Duo 3.16 GHz processor. Memory utilisation peaked at less that 150MB during decomposition of the 512×512 *Airplane* and roughly 60MB while processing the 300×451 *Chelsea* (including the memory required to store the decomposition itself). Computation times were 3.73s (*Airplane*) and 1.51s (*Chelsea).* Reconstruction executed in a few

Figure 3.5: *Two test images used for benchmarking the 2D DPT. On the left is* Chelsea the Cat *(*300×351*) and on the right is* Airplane *(*512×512*).*

milliseconds.

Figure 3.6 shows benchmark times for the Discrete Pulse Transform applied to random matrices. Random matrices with a large number of discrete values seem to be the worst case scenario—execution times are much lower for real photographs and for signals limited to, say, 255 discrete values. Both these observations are explained intuitively: a random matrix has many more pulses than a typical photograph and limited discrete values cause merging of pulses that would otherwise remain separated. It would be interesting to investigate whether a link exists between image entropy and the number of pulses generated.

3.3.6 Reproducibility and Code

The code for this `Python` implementation is available under the open source BSD licence at `http://dip.sun.ac.za/∼stefan/lulu`.

Further collaborative work into a parallel implementation as well as smarter storage methods for the pulses are being looked into to further improve the two dimensional implementation and provide a three dimensional implementation for video analysis.

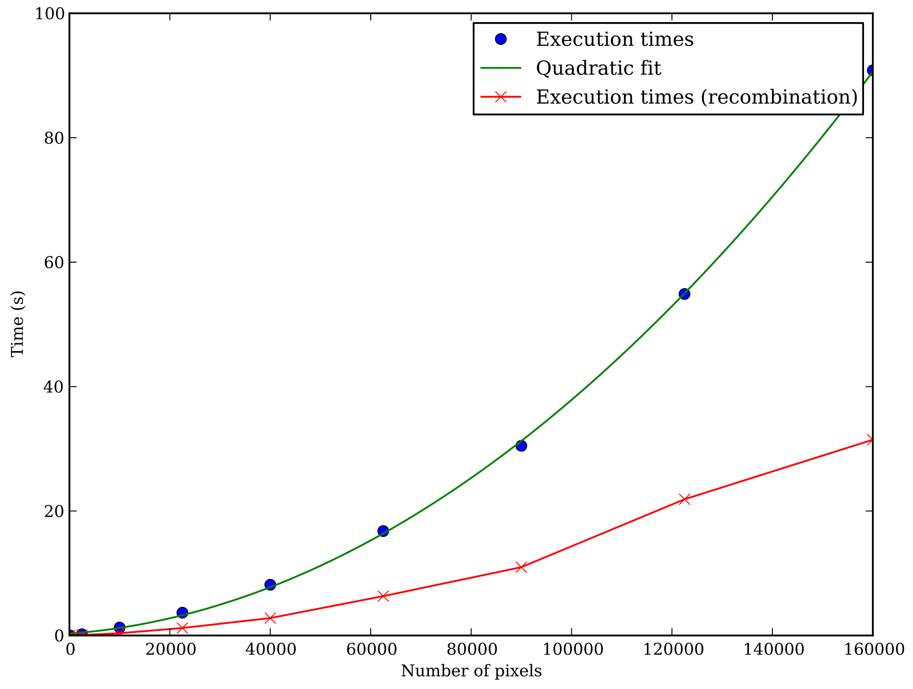


Figure 3.6: *Benchmark of the Discrete Pulse Transform on random images of varying size. Values on the x-axis indicate the total number of pixels, i.e., N^2 for an $N \times N$ matrix. In the bottom curve, labeled "recombination", the number of discrete input values were limited to 255. For large images, this quantisation causes the algorithm to execute more quickly than expected due to an increased number of merges.*

3.4 Properties of the Discrete Pulse Transform

3.4.1 Linear versus Nonlinear

As discussed in [52], the nonlinearity of the LULU smoothers, as mentioned in Chapter 2, make theoretical development more complicated than for linear operators. However, taking on the additional complexity is justified since in two dimensions an image is basically the transformation of data by a human eye or measuring instrument. This transformation is significantly complicated to be considered nonlinear [184]. Thus taking this stance the analysis of images via nonlinear operators is more logical than that of linear.

Linear processing techniques are however a natural starting point for analysis due to the simplicity of their application and theoretical backbone available. Examples of linear filters are the Fourier transform, Hadamard transform, the discrete cosine transform, and wavelets. They also provide sufficient results in most applications, but there are problems in which a nonlinear process would prove more viable and efficient [172]. Pitas and Venetsanopoulos [172] provide examples of such cases, such as signal dependent noise filtering e.g. photoelectron noise of photosensing devices; multiplicative noise appearing as speckle noise in ultrasonic imaging and laser imaging; and nonlinear image degradations e.g. when transmission occurs through nonlinear channels. Advantages of nonlinear filters are 1) the ability to handle various noise types, 2) edge preservation, 3) fine detail preservation, 4) unbiasedness (directional and illumination based) or invariance, and 5) computational complexity [172].

Nonlinear filtering techniques can be broadly classified accordingly in the following areas: order statistic filters, homomorphic filters, polynomial filters, mathematical morphology, neural networks, and nonlinear image restoration [172]. The LULU operators fit nicely into the areas of mathematical morphology as well as order statistics, two areas which have been integrated quite effectively in literature [172]. Examples of order statistics, discussed in detail in [172], are the median, rank-order filters, max-min filters, L_p-mean filters, and α-trimmed mean filters. The LULU operators are examples of max-min filters but with the disadvantages listed in [172] improved upon. The basic filters of mathematical morphology are the erosion and dilation, and subsequently the morphological opening and closing, to which the LULU filters are again closely related as area openings and closings.

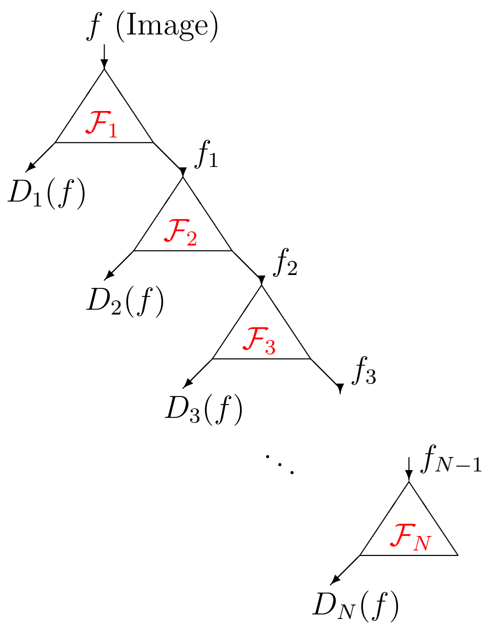


Figure 3.7: *A typical hierarchical decomposition*

3.4.2 Nonlinear Decompositions

Figure 3.7 presents the structure of a hierarchical decomposition. The operator $\mathcal{F}_1$ is applied to the input image f to obtain a decomposition of f into f_1, the smoother image, and D_1, the noise component removed. This process is repeated with $\mathcal{F}_2$, $\mathcal{F}_3$,...,$\mathcal{F}_N$ until there is nothing left to remove except the constant image D_N. The decomposition then has the form

$$f = D_1(f) + D_2(f) + ... + D_N(f). \tag{3.22}$$

Such a hierarchical decomposition has been investigated intensively in literature, see [223, 65, 247] for some nonlinear cases. However, in no literature we have come across have we found a unified theoretical backbone to connect such nonlinear hierarchical decompositions and provide methods of comparison nor methods of testing the capability of the structure of the decomposition. In Tadmor et al [223], for example, a decomposition $f = \sum_{j=1}^{k} u_j + v_k$ is obtained, where v_k is the noise component and the u_j's the decomposition components, by functional minimization. Tadmor et al discuss convergence of the minimizer, localization and adaptability, but nothing to indicate the strength of the decomposition save numerical visual examples. Wong et al [247] similarly do not provide a theoretical indication of the strength of their decomposition obtained as a probabilistic scale-space derived from the nonlinear diffusion equation of Perona and Malik [170]. In [65], Florack et al even state that comparisons with their proposed nonlinear scale-space and other nonlinear hierarchical decompositions 'are to be made with care'. It is thus clear that a unified theoretical setting for nonlinear decomposition escapes us.

A major advantage of the Discrete Pulse Transform is that it comes with its own theory comprising properties like consistent decomposition and total variation preservation, as is discussed in the next sections.

3.4.3 Consistent Decompositions

Definition 22 *A decomposition of the form (3.22) is called* consistent *if for every f in the considered domain and a set of nonnegative number $\alpha_1, \alpha_2, ..., \alpha_N$ we have*

$$D_j\left(\sum_{i=1}^{N}\alpha_i D_i(f)\right) = \alpha_j D_j(f),\ \ j = 1, 2, ..., N.$$

Definition 22 essentially means that the function

$$g = \sum_{i=1}^{N}\alpha_i D_i(f)$$

decomposes into its summands as is illustrated in Figure 3.8. This property is trivially true if D_j is a linear operator. However, it is also a desirable property for nonlinear decompositions. It implies that the components of the decomposition are in some sense 'quantitatively' different from each other, for example, multiplying one component by a positive number will not change it to another. This can be considered as an indication that each component extracts a feature or information which is to some extent unaltered and independent of the remaining part of f.

The property of consistency has a long history within the development of the Discrete Pulse Transform. Recall that the DPT of $f \in \mathcal{A}(\mathbb{Z}^2)$ is given by (3.1)-(3.3). In the one-dimensional case, i.e. $f \in \mathcal{A}(\mathbb{Z})$, the first result is the consistent decomposition of sums of the form

$$g = \sum_{i=m}^{n} D_i(f), 1 \leq m < n \leq N.$$

This property is called *basic decomposition*, see [183, Theorem 8.3]. In the same reference the full consistency is also proved [183, Theorem 8.4].

Note that for the DPT we have

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

so that

$$f = \sum_{i=1}^{N} \sum_{s=1}^{\gamma(n)} \psi_{ns}.$$

The success with the consistency property made the author believe that a stronger property might hold, namely that the sum

$$g = \sum_{s=1}^{\gamma(n)} \alpha_{ns}\psi_{ns}, \ \ \alpha_{ns} \geq 0$$

decomposes consistently, that is,

$$D_n(g) = \sum_{i=1}^{\gamma(n)} \alpha_{ns}\psi_{ns}, n = 1, 2, ..., N.$$

This was formulated as the Highlight conjecture in [183, p. 100] and restated later in [113]. The proof in the one-dimensional case was claimed in [114] but not shown. In our work on the multidimensional DPT, we proved the basis consistency of the decomposition, published in [8]. This was followed by a publication of D Laurie [112] where he proved the Highlight conjecture for functions derived on graphs, hence also applicable to $\mathcal{A}(\mathbb{Z}^2)$.

Here we present theorems proving the basic consistency as in [8] and the Highlight conjecture in the setting of $\mathcal{A}(\mathbb{Z}^d)$ since they each use different methods. In particular, the method of proof of the Highlight theorem shows the power of the morphological approach adopted here and applies the techniques in Theorem 17 which was originally presented in [8].

Theorem 23 *Let $f \in \mathcal{A}(\mathbb{Z}^d)$. For any two integers m and n such that $m < n$ the function $g = D_m(f) + D_{m+1}(f) + ... + D_n(f)$ decomposes consistently, that is*

$$D_j(g) = \begin{cases} D_j(f) & \text{for } \ m \leq j \leq n \\ 0 & \text{otherwise} \end{cases}$$

The proof uses the following lemmas.

Lemma 24 *a) Given $V, W \in \mathcal{C}$ with $W \subset V$. Then for $x \notin V$ but $x \in \operatorname{adj}(W)$, we have $x \in \operatorname{adj}(V)$.*
b) Given distinct $V, W \in \mathcal{C}$ with $V \cap W \neq \emptyset$, there exists an $x \in V \backslash W$ such that $x \in \operatorname{adj}(W)$.
c) For any $V \in \mathcal{C}$, we have $\operatorname{card}(V) < \infty \implies \operatorname{card}(\operatorname{adj}(V)) < \infty$.

Proof
a) V and $W + \{x\}$ are connected and have a nonempty intersection thus their union $V \cup \{x\}$ is also connected. Then by Definition 40, $x \in \operatorname{adj}(V)$.
b) Note that the following is true:

$$V, W \in \mathcal{C},\ W \subsetneq V \implies \operatorname{adj}(W) \cap V \neq \emptyset. \tag{3.23}$$

Applying (3.23) to $V \cup W$, since $V \subsetneq V \cup W$, we get $\operatorname{adj}(W) \cap V = \operatorname{adj}(W) \cap (V \cup W) \neq \emptyset$. Thus there exists $x \in V \backslash W$ such that $x \in \operatorname{adj}(W)$.
c) This follows from condition (2.3). Let $\operatorname{card}(V) = n$ then for an arbitrary $x \in V$ we have $\{\{a\} \cup V : a \in \operatorname{adj}(V)\} \subset \mathcal{N}_{n+1}(x)$, so that $\operatorname{card}(\operatorname{adj}(V)) \leq \operatorname{card}(\mathcal{N}_{n+1}(x)) < \infty$. ■

Lemma 25 *Let $Q_n = P_n P_{n-1} ... P_1$ where $P_k = L_k U_k$ or $P_k = U_k L_k$. We have*

a) $Q_n Q_m = Q_{\max\{n,m\}}$

b) $Q_m(id - Q_n) = Q_m - Q_n = (id - Q_n)Q_m$ for all integers m, n such that $m \leq n$.

Proof
We consider only $P_k = L_k U_k$ as the other case is dealt with by symmetry. Let $f \in \mathcal{A}(\mathbb{Z}^d)$.
a) It follows from Corollary 11 in Chapter 2 that $Q_n(f)$ does not have any local minimum or local maximum sets of size n or less. Hence $P_k(Q_n(f)) = Q_n(f)$ for $k = 1, ..., n$. For $m \leq n$ this implies that $Q_m(Q_n(f)) = Q_n(f)$. If $m > n$ then we have
$(Q_m Q_n)(f) = (P_m ... P_{n+1} P_n ... P_1)(Q_n(f)) = (P_m ... P_{n+1})(Q_n(f)) = C_m(f)$.
b) We use induction on j as in the proof of this property in the one dimensional case, see [182]. Let $j = 1$. Using the result in Lemma 16(b), the full

trend preservation property of the LULU operators established in Chapter 2 and the absorbtion property in a) we have

$$\begin{aligned}&Q_1(id - Q_n) = L_1(U_1(id - Q_nL_1U_1)) = L_1(U_1 - Q_nL_1U_1)\\&\quad = L_1(id - Q_nL_1)U_1 = (L_1 - Q_nL_1)U_1 = Q_1 - Q_n = (id - Q_n)Q_1.\end{aligned}$$

Assume now that the statement is true for some $m = j < n$. From the inductive assumption we have

$$\begin{aligned}Q_{j+1}(id - Q_n) &= P_{j+1}Q_j(id - Q_n) = P_{j+1}(Q_j - Q_n)\\&= P_{j+1}(Q_j - Q_nQ_j) = P_{j+1}(id - Q_n)Q_j.\end{aligned}$$

Using Lemma 16(b), the fully trend preserving property and a) as for $j = 1$ we obtain further

$$\begin{aligned}&P_{j+1}(id - Q_n)Q_j = L_{j+1}U_{j+1}(id - Q_nL_{j+1}U_{j+1})Q_j\\&\quad = (L_{j+1}U_{j+1} - Q_nL_{j+1}U_{j+1})Q_j = Q_{j+1} - Q_n = (id - Q_n)Q_{j+1}.\end{aligned}$$

■

Proof of Theorem 23

Using Lemma 25, function g can be written in the following equivalent forms

$$\begin{aligned}g &= ((id - P_m)Q_{m-1} + (id - P_{m+1})Q_m + ... + (id - P_n)Q_{n-1})(f)\\&= (Q_{m-1} - Q_n)(f) = (id - Q_n)Q_{m-1} = Q_{m-1}(id - Q_n). \qquad (3.24)\end{aligned}$$

It follows from the fact that since for every $f \in \mathcal{A}(\mathbb{Z}^d)$ the functions $(L_n \circ U_n)(f)$ and $(U_n \circ L_n)(f)$ have neither local maximum sets nor local minimum sets of size n or less, as well as the neighbour trend preserving property of the LULU operators that g does not have any local maximum or local minimum sets of size less than m. Hence $P_k(g) = g$ for $k = 1, ..., m - 1$ and therefore $Q_k(g) = g$ for $k = 1, ..., m - 1$. Then it follows from (3.24) that $D_j(g) = (id - P_j)(g) = 0$ for $j < m$. Let $m \leq j \leq n$. Then using again Lemma 25 we obtain

$$\begin{aligned}D_j(g) &= (Q_{j-1} - Q_j)(g) = (Q_{j-1}(id - Q_n)Q_{m-1} - Q_j(id - Q_n)Q_{m-1})(f)\\&= ((id - Q_n)Q_{j-1}Q_{m-1} - (id - Q_n)Q_jQ_{m-1})(f)\\&= ((id - Q_n)Q_{j-1} - (id - Q_n)Q_j)(f) = (Q_{j-1} - Q_n - Q_j + Q_n)(f)\\&= (Q_{j-1} - Q_j)(f) = D_j(f).\end{aligned}$$

Finally, for $k \geq n$ we have

$$Q_k(g) = (Q_k(id - Q_n)Q_{m-1})(f) = (Q_kQ_n(id - Q_n)Q_{m-1})(f) = 0,$$

which implies that $D_j(g) = 0$ for $j > n$. ■

Theorem 26 *For a DPT decomposition of f*

$$DPT(f) = (D_1(f), D_2(f), ..., D_N(f)) \text{ where } D_n(f) = \sum_{s=1}^{\gamma(n)} \psi_{ns},\ n = 1, 2, ..., N$$

let $g = \sum_{n=1}^{N} \sum_{s=1}^{\gamma(n)} \alpha_{ns}\psi_{ns}$ where the constants α_{ns} are positive. Then the DPT decomposition of g is obtained as

$$DPT(g) = (D_1(g), D_2(g), ..., D_N(g)) \text{ where } D_n(g) = \sum_{s=1}^{\gamma(n)} \alpha_{ns}\psi_{ns},\ n = 1, 2, .., N$$

so that the pulses of g are obtained as $\alpha_{ns}\psi_{ns}$. If $\alpha_{ns} = \alpha_n$ for each n then $DPT(g) = \sum_{n=1}^{N} \alpha_n D_n(f)$, so that $D_n(g) = \alpha_n D_n(f)$.

Proof
We carry out the proof by using mathematical induction. Denote $g_m = \sum_{n=N-m+1}^{N} \sum_{s=1}^{\gamma(n)} \alpha_{ns}\psi_{ns}$. Note that $D_N(f)$ consists of only a single pulse ψ_N. Hence, the statement is trivially true for $g_1 = \alpha_N \psi_N$.

Assume that it holds for some $m < N$ and consider $g_{m+1} = \sum_{n=N-m}^{N} \alpha_{ns}\psi_{ns}$. Considering the properties of the supports of the DPT pulses as stated in (3.19) and (3.20) we deduce that

(i) g_{m+1} does not have local maximum or minimum sets of size less than $N - m$,

(ii) the local maximum and local minimum sets of g_{m+1} of size $N - m$ are exactly $\{\text{supp}\{\psi_{N-m,s}\}, s = 1, 2, ..., \gamma(N - m)\}$,

(iii) for every $s = 1, 2, ..., \gamma(N - m)$ the function g_{m+1} is constant on $\text{supp}\{\psi_{N-m,s}\}$ and the difference with the nearest value on the adjacent set is $\alpha_{N-m,s}$ multiplied by the height of $\psi_{N-m,s}$.

Then by Corollary 11

$$D_i(g_{m+1}) = (id - P_i)P_{i-1}...P_2P_1(g_{m+1}) = 0,\ i = 1, 2, ..., N - m - 1.$$

Further, it follows from Lemma 16 that

$$\begin{aligned} D_{N-m}(g_{m+1}) &= (id - P_{N-m})P_{N-m-1}...P_2P_1(g_{m+1}) \\ &= (id - P_{N-m})g_{m+1} \\ &= \sum_{s=1}^{\gamma(N-m)} \alpha_{N-m,s}\psi_{N-m,s}. \end{aligned}$$

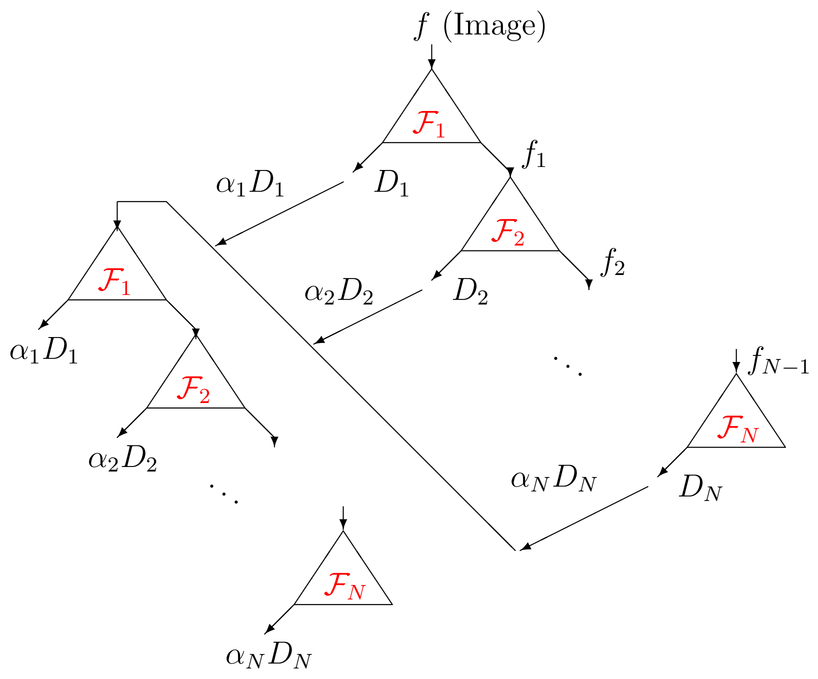


Figure 3.8: *Illustration of consistent decomposition*

By the inductive assumption the function $g_m = g_{m+1} - D_{N-m}(g_{m+1})$ decomposes consistently. Hence g_{m+1} also decomposes consistently. The statement of the theorem is the obtained for $m = N$.

■

3.4.4 Total Variation Preservation

Although the importance of total variation preservation for separators cannot be doubted, it is even more so for hierarchical decompositions like the Discrete Pulse Transform, due to the fact that they involve iterative applications of separators. Since the operators L_n, U_n, $n = 1, 2, ...$, and all their compositions, are total variation preserving, it is easy to obtain the statement of the following theorem, which shows that, irrespective of the length of the vector in (3.1) or the number of terms in the sum (3.21), no additional total variation, or noise, is created via the decomposition.

Theorem 27 *The discrete pulse decomposition in (3.1) is total variation* *preserving, that is*

$$TV(f) = \sum_{n=1}^{N} \sum_{s=1}^{\gamma(n)} TV(\psi_{ns}). \tag{3.25}$$

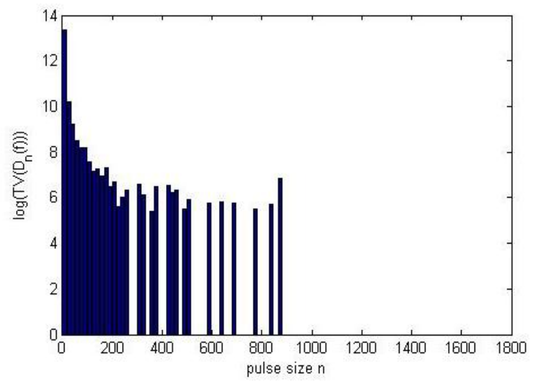
The proof of Theorem 27 can be found in [8, 52]. We should remark that representing a function as a sum of pulses can be done in many different ways. However, in general, such decompositions increase the total variation, that is, we might have strict inequality in (3.25) instead of equality. Based on Theorem 27 we can construct the total variation distribution of images. More precisely, this is the distribution of the total variation of an image among the different layers of the DPT. That is, essentially the plot of $TV(D_n(f))$ vs. n. In Figure 3.9 we present the total variation distributions of some images, where one can observe how the total variation of each image as given in Table 2.6 is distributed over the pulse size. A log scale is used on the vertical axis and the pulse size values are grouped to form a histogram. The different character of the images naturally manifests through different kinds of total variation distributions.

3.4.5 Measuring the Smoothing Ability of the LULU Operators

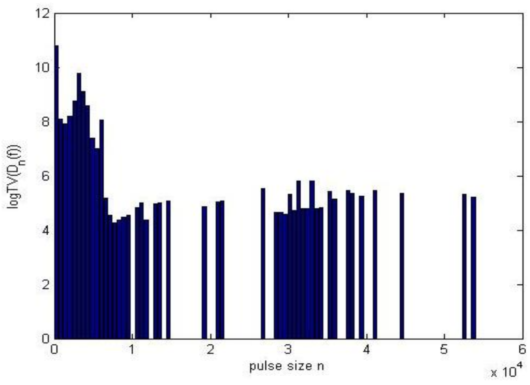
The ability of an operator to effectively remove noise and smooth the signal is usually measured by its output variance or the rate of success in the noise removal [172]. Other measures used to assess the performance are the mean square error (MSE) and signal-noise-ratio (SNR) [172]. We investigate the noise removal Chapter 5 in detail. In this section we shall present a method in which to measure the quality of smoother or equivalently the resulting smoothed image. In [131] an operator $E_\gamma : X \to X$, for a Hilbert space X, is a smoother in the sense that

$$w - \lim_{\gamma \to 0} E_\gamma f = f.$$

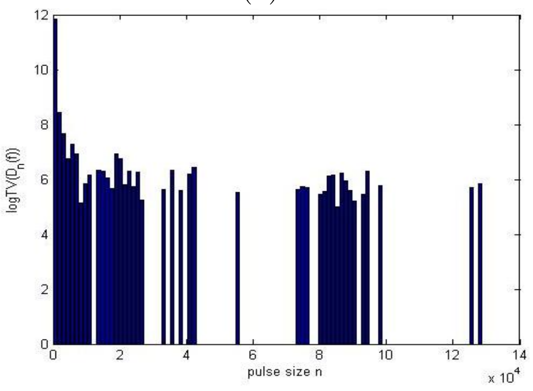
Velleman [234] states that it is required from a smoother to separate the signal into noise and a smooth signal and suggests measuring the success of a smoother by obtaining a regression coefficient near one for the fit of a least squares regression of the smoothed signal compared to the original signal. In [175], a smoothing function $\mathcal{G}_\epsilon$ of f is defined such that for $\epsilon > 0, \mathcal{G}_\epsilon : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable on $\mathbb{R}^n$ and for all $x \in \mathbb{R}^n$, $\|f(z) - \mathcal{G}_\epsilon(z)\| \to 0$ as $\epsilon \to 0, z \to x$.



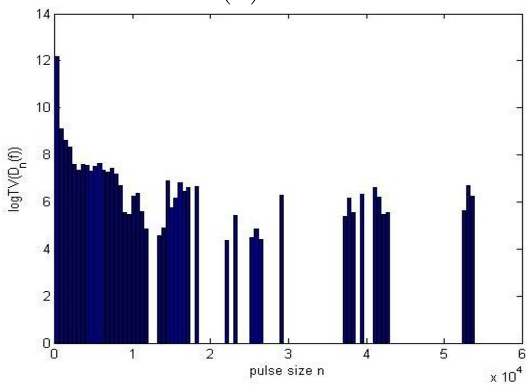
(a)



(b)



(c)



(d)

Figure 3.9: *Total Variation Distributions of Images in: (a) Figure 2.4(h)-Noise, (b) Figure 2.4(a)-Potatoes, (c) Figure 2.4(c)-Tank, (d) Figure 2.4(f)-Boat with Glint*

The question of measuring the smoothing ability of the DPT arises. The aim of a smoother primarily to remove the noise element present. The noise can be due to a number of factors, for example, acquisition, processing, compression, storage, transmission and reproduction of the image [235]. The easiest method of evaluation is purely subjective - namely, human visual investigation. In order for evaluation to be objective, quantitative methods need to be used instead. Quantitative methods can be divided into three categories [235]. First, *full-reference*, where the complete reference (undistorted) image is known with certainty, secondly, *no-reference*, where this reference image is not known at all, and third, *reduced-reference*, where only part of the original reference image is known, for example, a set of extracted features. We measure the similarity of the smoothed images $P_n(f)$ to the original unsmoothed image f with the structural similarity index [235],

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

for two corresponding sets of pixels, x and y, in each image, where $\mu_i, i = x, y$ is the mean of the pixel values in i; $\sigma_i^2, i = x, y$ is the variance of the pixel values in i; σ_{xy} the covariance between x and y; $c_j = (k_j L)^2, j = 1, 2$ constants to stabilize the division by the weak denominator; $L = 255$ for greyscale images and $k_j \ll 1$ constants (we used $k_j = 0.05$). This measure is a full-reference measure which provides a useful framework since we are comparing a smoothed version of the original distorted image with the original distorted image. The most widely used such measures are the mean-square-error (MSE) and the peak signal-noise-ratio (PSNR), but these measures do not compare well with the perceived visual quality of the human visual system [235]. Wang et al [235] introduce the SSIM measure in order to penalize errors based on their visibility, that is, to simulate the HVS as much as possible. This measure is applied to 8×8 windows in the image for each pixel and a final mean structural similarity index is calculated as the average of these SSIM values, called the MSSIM. An MSSIM value closer to 1 indicates stronger similarity. Wang et al provide MATLAB code for the implementation of the SSIM at `www.cns.nyu.edu/~lcv/ssim` which was made use of.

Figure 3.10 provides MSSIM values for various images as the LULU smoothing progresses through the DPT from scale $n = 1$ up to $n = N$. Notice how, based on the content of the images, the reduction in the MSSIM values as the DPT progresses varies from image to image. The graphs provide a mechanism to determine where visual structure is in the image, that is, when the HVS would pick out structures of significance. Figure 3.11 plots the differ-

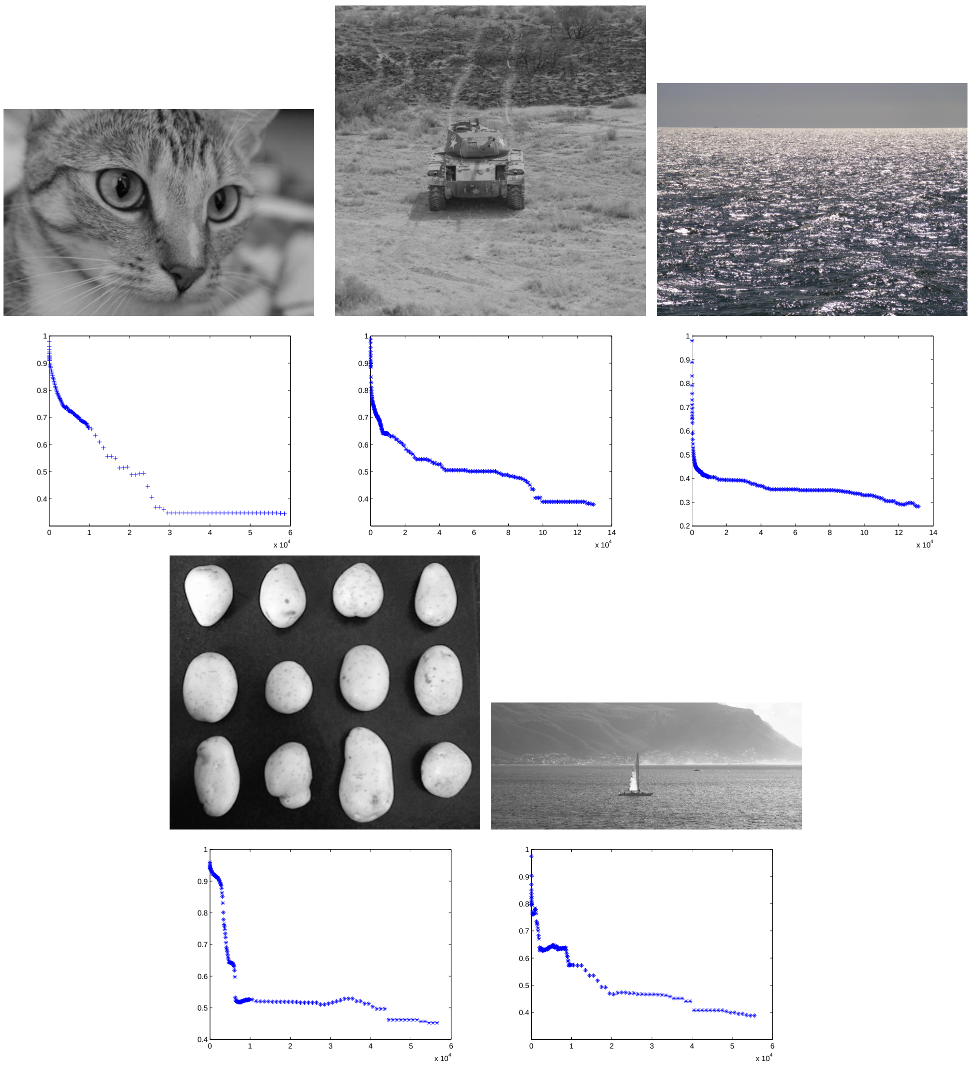


Figure 3.10: *MSSIM values comparing $P_n(f)$ with f values plotted against scale for* Chelsea, Tank, Potatoes, Ocean *and* Jetski *(values indicated are for increments of 10 up to scale 100, then increments of 100 up to scale 10000, and then increments of 1000 up to the maximum scale)*

ences between the smoothed images as the DPT progresses. The graphs also give an indication of how the information is removed at each step. Figures 3.12 and 3.13 provide an indication of the structure found at big jumps in Figures 3.10 and 3.11.

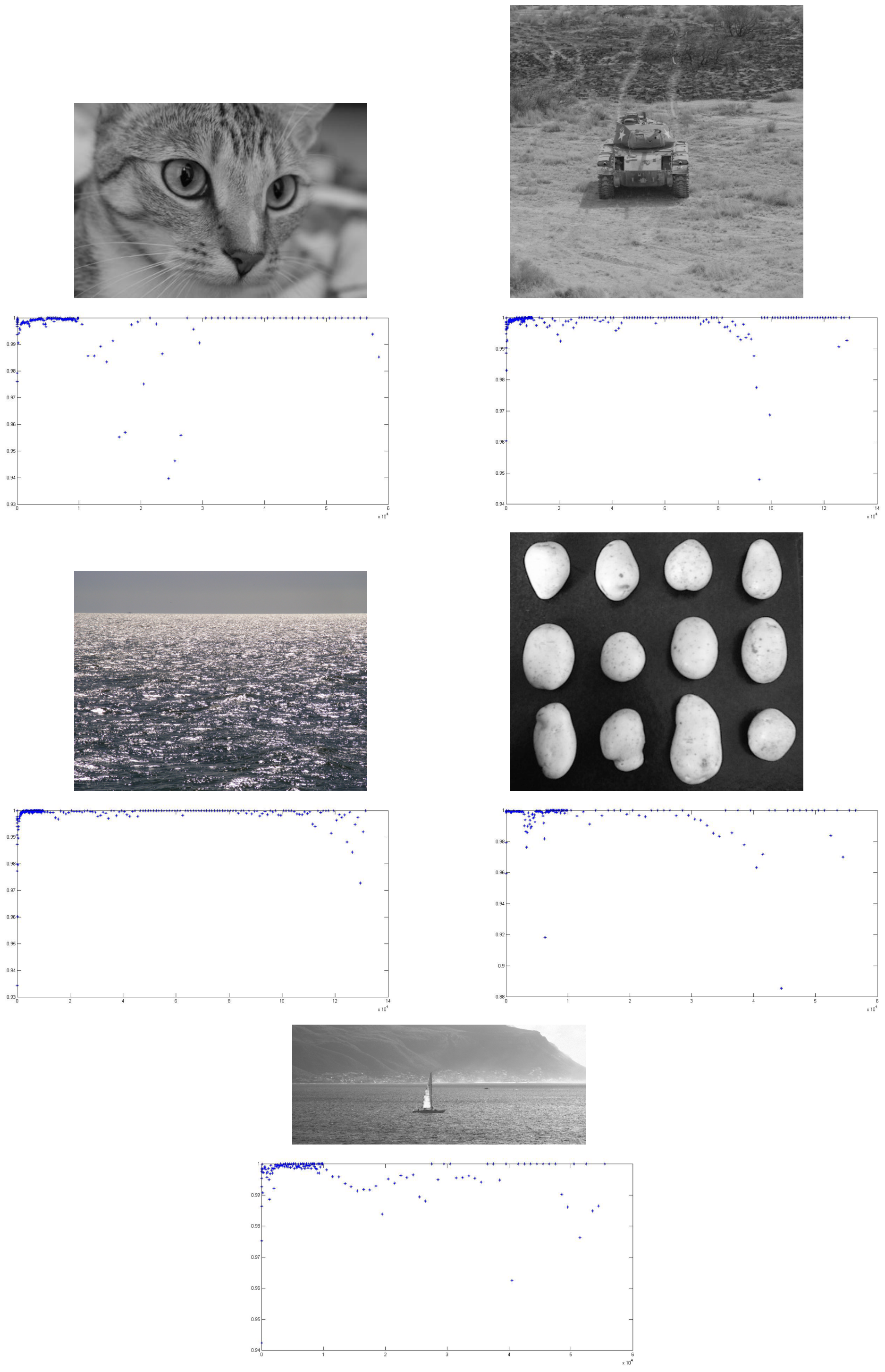


Figure 3.11: *MSSIM values comparing* $P_n(f)$ *with* $P_{n-1}(f)$ *plotted against scale for* Chelsea, Tank, Potatoes, Ocean *and* Jetski *(values indicated are for increments of 10 up to scale 100, then increments of 100 up to scale 10000, and then increments of 1000 up to the maximum scale)*

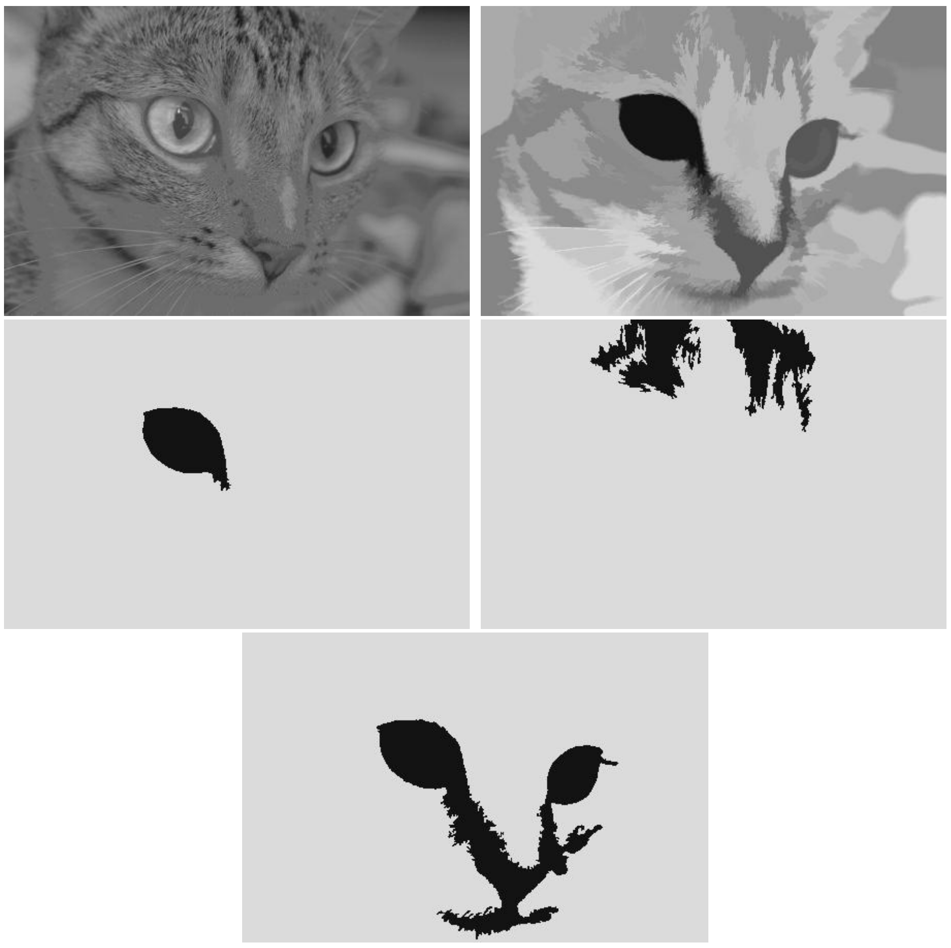


Figure 3.12: *Specific scales of* Chelsea *picked out using Figures 3.10 and 3.11: 1 to 4030 (detail), 4031 to 58571 (big background pulses), 4234 to 4236 (left eye), 4325 to 4335 (forehead), 14565 to 14575 (facial features)*

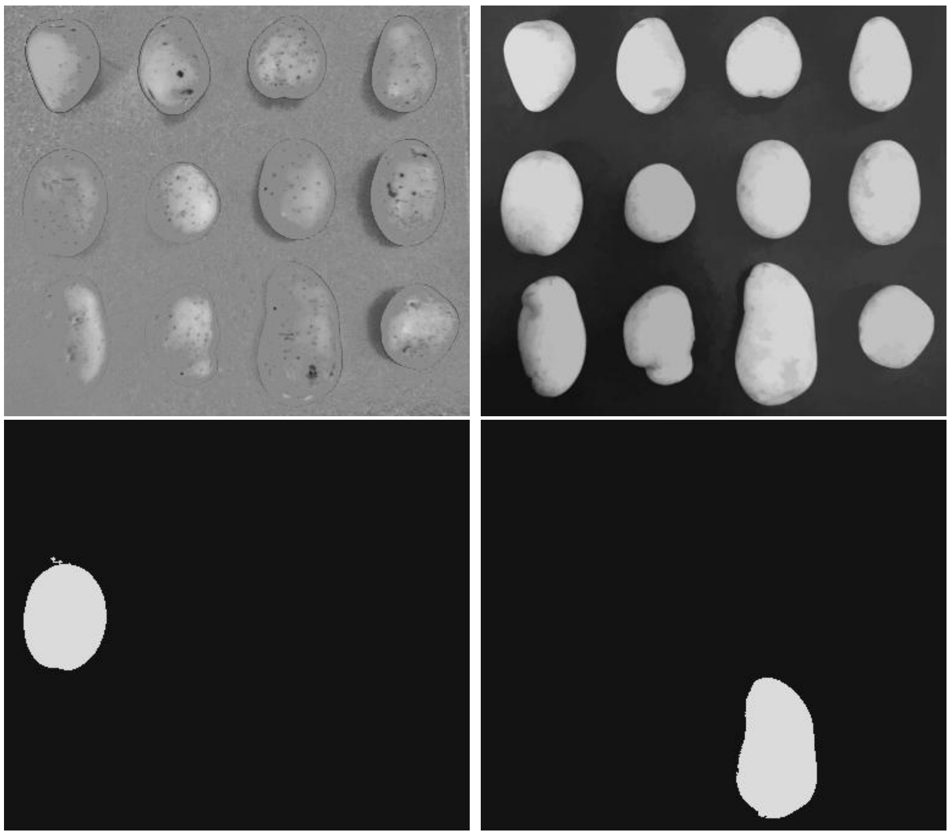


Figure 3.13: *Specific scales of* Potatoes *picked out using Figures 3.10 and 3.11: 1 to 2002 (detail), 2003 to 57478 (big background pulses), 4712 (far left, middle row potato), 5900 (right, last row potato)*

3.4.6 Pulse Shape

Lastly, an important aspect of the DPT is the provision of pulses without restricting their shape in any way. Recall they we only require the pulses to be connected, most commonly with 4-connectivity. This provides a vast area of investigation into the nature of the pulse shapes and what the shapes mean relative to the image structure, image patterns and image content. We look into exploiting this information in Section 4.8.2.

3.5 Conclusion

We have presented the Discrete Pulse Transform resulting from the alternative and recursive application of the LULU operators L_n and U_n, known as the DPT. In particular we looked at the characterization of the DPT with respect to its nonlinearity, consistency, shape preservation properties such as total variation preservation, as well as its ability as a smoother.