The tree structure for time-of-flight dataset representation

Minjie Chen Matthias Plaue Günter Bärwolff Hartmut Schwandt

Institut für Mathematik, Technische Universität Berlin, Germany

minjie.chen@.. plaue@.. baerwolf@.. schwandt@math.tu-berlin.de

March 18, 2010

Abstract: In the current paper, we present a tree-like structure for the object representation in time-of-flight (TOF) datasets which enables an efficient computation of the object contour lines. The generated contour lines can be further processed in object detection and recognition for the TOF dataset. We illustrate this idea in the application of a monitoring system using the TOF ranging technology. In addition, we also present a fast algorithm of visualizing the object contour lines as unstructured (that is, raw) pixels.

Keywords: tree, time-of-flight, contour line, object detection/recognition

1 Introduction

Three-dimensional (and semi-three-dimensional) images of real-world objects can be acquired by various means, among which radar, interferometry and triangulation are frequently applied. A major component of three-dimensional imaging is the ranging system. We refer to [1] for an overview of the optical ranging systems. Time-of-flight (TOF) imaging is a new emerging technique of contactless distance measurement; the key of this method is the light travel duration over the spatial distance. This technique dates back to the famous, though due to the technical limitation at that time, unsuccessful light speed experiment of Galileo Galilei in which the measurement of light propagation time between two observers was attempted, using a mechanical clock. However, robust measurement of light travel duration over small distances (the light travel duration for a typical distance of 10 m is 33 n s approximately) is not an easy task even within the realm of contemporary technology. In the context of three-dimensional imaging of real-world objects, the measurement is to be carried out on points of a sufficient density, and this poses an additional difficulty. A possible solution for this is to measure the phase shift $\Delta\phi$ of an amplitude-modulated light wave which is proportional to the flight duration rather than this time span directly, this can be expressed as

$$\Delta\phi = \frac{4\pi f}{c} \cdot d, \tag{1}$$

where d denotes the distance, $c \doteq 3.0 \cdot 10^8\ \mathrm{m\,s^{-1}}$ the speed of light under standard conditions, and f the modulation frequency. Although laser-based TOF systems of this kind have been known at least since 1977 [13], real-time "on-chip" systems did not appear until with the development of novel CMOS technologies [15, 17].

The proportionality (1) of the light phase shift and the spatial distance is the functional cornerstone of the current TOF sensor systems. We explain this principle idea briefly. After the scene is actively illuminated by an incoherent infrared light amplitude-modulated with a specific frequency f, this optical signal is reflected by the object being measured at a distance d, and returns after the time lapse of $T = \frac{2d}{c}$ to an array of CMOS pixels. The CMOS pixel array captures both the spatial distribution of intensity of the reflected signal and the phase-shift, the latter of which—up to an unambiguous range of $d_{\max} = \frac{c}{2f}$—is proportional to the distance of the object. A typical modulation frequency is $f = 20\,\mathrm{MHz}$ which amounts to a maximum distance range of $d_{\max} = 7.5\,\mathrm{m}$. For an overall survey and technical details on the subject we refer to [10, 12].

The actual phase shift $\Delta\phi$ and intensity I is usually computed from four sampled values of the received optical signal $I_0, \ldots, I_3$; or rather their differences $J_1 = I_0 - I_2$ and $J_2 = I_3 - I_1$,

$$\begin{aligned} I &= \tfrac{1}{2}\left|J_1 + iJ_2\right|, \\ \Delta\phi &= \arg\left(J_1 + iJ_2\right) - \phi_0; \end{aligned} \tag{2}$$

see for example [6]. In case $\phi \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, (2) can be written as

$$\Delta\phi = \arctan\left(\frac{J_2}{J_1}\right) - \phi_0.$$

The sampled intensity values are obtained by an on-chip correlation of the emitted and the received signal. ϕ_0 is a constant offset due to the propagation time of the internal signal and has to be calibrated away. It should be noted that this technique functions under the assumption of a harmonic signal modulation which is actually not given with real-world systems and leads to systematic errors dependent of the actual range [14]. Furthermore, the standard deviation of the phase-range measurement is directly reciprocal to the intensity. This allows one to simply discard measurements with an intensity that is too low or to filter the range image with the intensity as a confidence value adaptively [6].

Another pre-processing step is the transformation of chip coordinates (m, n, d) into Cartesian world coordinates (x, y, z):

$$\begin{aligned} x &= \frac{nd}{\sqrt{\alpha^2+m^2+n^2}}, \\ y &= \frac{md}{\sqrt{\alpha^2+m^2+n^2}}, \\ z &= \frac{\alpha d}{\sqrt{\alpha^2+m^2+n^2}}. \end{aligned} \tag{3}$$

Here, m and n are the vertical and horizontal distances from the centre of the chip in pixel units, and d is the measured radial distance. The world coordinate system has its origin at the centre of the chip, and the z-axis is parallel to the optical axis of the camera. Furthermore, $\alpha = \frac{l_f}{l_p}$, where l_f is the focal length of the camera and l_p is the physical edge length of the pixel. Near the chip's centre, (3) can be approximated as

$$\begin{aligned} x &\doteq \frac{nd}{\alpha}, \\ y &\doteq \frac{md}{\alpha}, \\ z &\doteq d. \end{aligned}$$

Growing applications of TOF ranging systems have been found in the fields of object/human detection and recognition [3, 7], object/human tracking [2, 8, 18], robotics and automated production [9, 19], human-machine interaction [4, 16], surveillance [5] etc.

In the current paper, we consider the problem of object detection using TOF data. The TOF data of a measurement is usually stored as a time series of frames which contain the distance value at every given two-dimensional position at a given time point. Contemporary technology supports a measurement frequency of up to fifty frames per second approximately. Due to the large data volume, we believe that an efficient data structure for the representation is indispensable. Our proposed data structure optimizes the computation of contour lines of objects in the scene. These contours preserve the volume information of the measured objects and can thus be further used in object detection, recognition and tracking.

2 Model

We assume that the measured objects are placed on a ground floor and the optical axis of the sensor is perpendicular to it. Applying the world coordinate system of (3), this ground floor is in fact the plane $z = h_0$, where h_0 denotes the distance from the sensor to the floor. For the simplicity of the notation, we let $(0, 0, h_0)$ be the origin of a new world coordinate system and reverse the direction of the z-axis, that is,

$$x \mapsto x, \quad y \mapsto y, \quad z \mapsto h = h_0 - z. \tag{4}$$

Thus, the sensor is positioned at $(0, 0, h_0)$ in the new world coordinate system. For a schematic description, we refer to Figure 1.

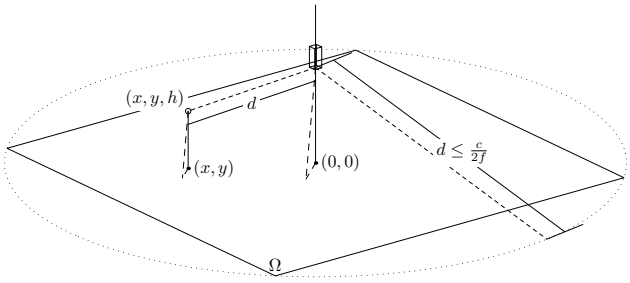

Figure 1: By proper pre-processing the distance value d will be projected onto the z-axis. For an object measured (drawn as a small circle) this value will be further transformed by (4) into a height value h in the new world coordinate system. The ground floor is the plane $z = 0$. In the ideal case, the optical axis of the installed sensor (drawn as a small box) should be perpendicular to the floor. The otherwise propagated errors are shown as small dashed line segments; if so, an additional affine transformation of the coordinates to compensate the errors would be necessary.

The measurable region of the ground floor is in most cases a rectangle (depending on the TOF sensor technology). We write this as a rectangular grid Ω of $\{1, \ldots, n_x\} \times \{1, \ldots, n_y\}$, where n_x and n_y denote the number of samples in the x- and y-dimension respectively. Each of the grid points on Ω is associated with

a pre-processed height value, $H_{x,y}$, $x = 0, \ldots, n_x - 1$, $y = 0, \ldots, n_y - 1$. The boundary of Ω is written as $\partial\Omega$.

An obvious result is that, apart from measurement error or inaccuracy, an object detected by the TOF sensor always exhibits positive height values in the new coordinate system. We are primarily interested in the grid positions with considerable positive height values, taking into consideration small random measurement errors.

2.1 TOF-node

In a TOF sensor monitoring scenario, the measured objects can be considered as being "placed" on the ground floor $z = 0$. After the transformation of (3), the outer surface in the positive z-direction of an object is in many cases geometrically convex. When we consider the height value as a continuous function on the x-y-plane, we observe the following

Theorem (Standard version)**.** *The height function h of a strict convex object has only one local maximum.*

Proof. Instead of "height function of a convex object" we simply say "concave function". We show the contraposition of the claim. Let the concave height function $h : D \to \mathbb{R}$ be defined on a convex set $D \subset \mathbb{R}^2$. Let $h(p)$ and $h(q)$ be two different strict local maxima of h at $p, q \in D$. Without loss of generality, we assume $h(q) \geq h(p)$. Since D is convex, there exists an $\varepsilon \in (0, 1]$ for the local maximum $h(p)$ such that $h(p') < h(p)$ for all $p' = (1-t)p + tq$ with $t \in (0, \varepsilon)$. For any t so chosen, we also have $h\left((1-t)p + tq\right) = h(p') < h(p)$. On the other hand, $h(p) < (1-t)h(p) + th(q)$ for $t \in (0, 1)$, since the right-hand side is monotonely increasing with respect to t. This contradicts the concavity of h. □

We recall that the strict convexity of h is formally defined on its convex domain of definition D,

$$\begin{aligned} &\forall\, p, q \in D, p \neq q :\ \forall\, t \in (0,1) : \\ &\quad h\left((1-t)p + tq\right) > (1-t)h(p) + th(q). \end{aligned} \tag{5}$$

In real-world scenarios, however, the domain of definition is discrete, composed of finitely many points $p_1, \ldots, p_n \in \mathbb{R}^2$. In correspondence with the measurement of the object, the height function h is given on these positions. We may divide the original domain $\mathbb{R}^2$ into geometrically convex sub-domains $D_1, \ldots, D_n$ associated with $p_1, \ldots, p_n$ (for example, by constructing the Voronoi diagram). See Figure 2 for a schematic example. To simplify the notation, we write $N = \{1, \ldots n\}$ for the index set. Since for every $i \in N$, p_i is the only position where a measurement is committed, we may define the height function h on the sub-domain D_i,

$$h(D_i) := h(p_i), \tag{6}$$

for all $i \in N$.

Intuitively, a position p can be considered as a local maximum, if its height value exceeds all those in the surrounding sub-domains of its own.

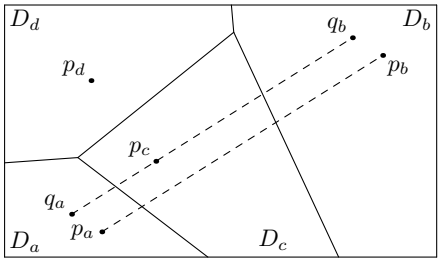

Figure 2: The Voronoi diagram associated with four points.

When we adapt (5) into

$$\begin{aligned} &\forall\, c, a, b \in N, c \neq a, c \neq b : \\ &\quad \exists\, q_a \in D_a, q_b \in D_b :\ \exists\, t \in (0,1) : \\ &\qquad (1-t)q_a + tq_b = p_c \Longrightarrow \\ &\qquad\quad h(D_c) > (1-t)h(D_a) + th(D_b) \end{aligned} \tag{7}$$

for the discrete case, we have the following

Corollary (Discrete version)**.** *A function h defined on finitely many points, satisfying* (7)*, has only one local maximum.*

In words, (7) means that for every pair of the n sub-domains associated with the measurement points, if there exists a third sub-domain lying between them, the height function (6) defined on these three sub-domains is convex. We wish to draw the attention that q_a and q_b in (7) are not necessarily positions where the measurements are available (p_a and p_b), these are needed to locate the relevant sub-domains to compare the height values.

Thus it is possible for us to characterize the objects by their height values which are retrievable from their physical locations. These objects can be labelled and distinguished from each other by the relevant maximum height values in the associated regions. This means an object can be represented by

a tree-like data structure in which parent nodes always have larger height values than their child nodes. Therefore, we define a type of memory unit "TOF-node" to hold the original data (position and value) and four pointers to other memory units of the same kind (see Figure 3). The four pointers de-reference the data in the neighbouring points at the relative positions $(\pm 1, 0)$ and $(0, \pm 1)$, they are written as W (west, $(\Delta x, \Delta y) = (-1, 0)$), N (north, $(\Delta x, \Delta y) = (0, 1)$), E (east, $(\Delta x, \Delta y) = (1, 0)$) and S (south, $(\Delta x, \Delta y) = (0, -1)$). For a TOF-node representing no local maximum, at most three of W, N, E and S de-reference further TOF-nodes.

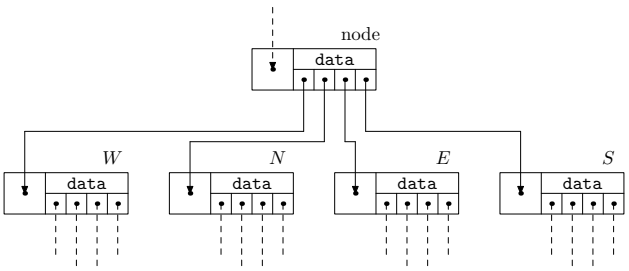

Figure 3: Basic structure of a TOF-node. Each node holds the pointers to up to four child nodes (W, N, E and S). The node "node" may be the root node of a TOF-tree, therefore its incoming pointer may not exist. The child nodes hold further pointers to their child nodes. Some of these pointers may be `NULL` (de-referencing no further TOF-nodes), therefore, they are drawn as dashed.

Consequently, a convex object can be represented by a tree-like object with a starting TOF-node as its root. The root TOF-node is associated with the local maximum the convex object exhibits. An object in reality, which may not be perfectly convex, can be decomposed into multiple TOF-trees.

2.2 TOF-tree construction

To represent the objects in the TOF dataset, we investigate the local maxima of height value function. We start from each of these maxima to construct the related TOF-trees.

procedure `main`**:**
parameter: global information Ω and T
  repeat
    find grid point p with maximum height;
    create TOF-node pointer `tp` with p;
    call `build_TOF_tree` with `tp`;
  until all grid points processed
return

The next procedure illustrates the construction of the TOF-tree starting with a TOF-node pointer `tp`.

procedure `build_TOF_tree`**:**
parameter: `tp`
  allocate memory for W, N, E and S;
  initialize W, N, E and S with default (`NULL`);
  mark grid point de-referenced by `tp` as processed;
  call `build_TOF_tree_W`;
  call `build_TOF_tree_N`;
  call `build_TOF_tree_E`;
  call `build_TOF_tree_S`;
return

This procedure checks the neighbouring grid points at the relative positions $(\pm 1, 0)$ and $(0, \pm 1)$, whether the TOF-tree further expands or the construction terminates. This can be realized by recursion through four internal wrapper procedures `build_TOF_tree_W`, `build_TOF_tree_N`, `build_TOF_tree_E` and `build_TOF_tree_S` to expand the current TOF-tree in the four directions. In the first direction:

procedure `build_TOF_tree_W`**:**
parameter:
  if candidate grid point not defined in Ω $(*)$
    ;
  elseif candidate grid point not already processed
    and associated height value not increasing $(**)$
    set W as TOF-node pointer to candidate;
    call `build_TOF_tree` with W;
  fi
return

The other three are analogous.

2.3 The expansion of TOF-tree

We recall that the criterion applied in the internal procedures of TOF-tree construction is the monotonicity of decreasing height values. In addition to the filtering in the pre-processing, a tolerance value can be used to smooth out small measurement errors (caused by technical limitations of the TOF sensor hardware) so that the construction of the TOF-tree should not terminate unexpectedly, since the monotonicity of height

values can be greatly affected by even a single captured pixel. However, the set of the TOF-trees associated with the local maxima always presents a partitioning of the floor Ω.

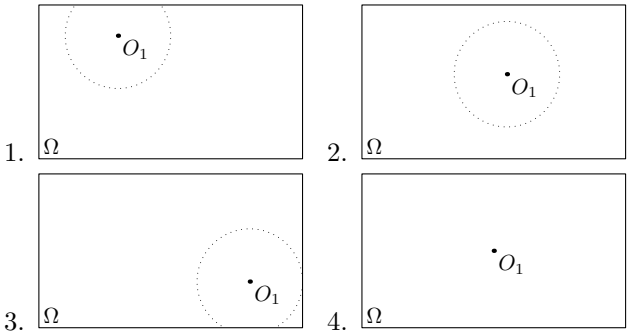

Figure 4: Four scenarios of a single object. The root node of the TOF-tree representing the object is drawn as a point. Boundary type III ($h = 0$, where the height values drop to 0) is drawn as dots. This boundary may or may not be completely visible within Ω.

In the internal wrapper procedures of TOF-tree construction, the candidate must be checked whether it is defined in Ω, in case the condition $(*)$ is fulfilled, the expanding TOF-tree reaches the boundary of the geometrical setting. We call this boundary type I. If the condition $(**)$ is not fulfilled, the TOF-tree stops expanding and the grid position in Ω of the candidate has the shape of a "valley" (among two or more convex objects) and thus forms the common boundary of this TOF-tree and others. We call this type II boundary. To specify that this boundary is shared with another TOF-tree T', it can be written as "type II(T')" explicitly.

In case the condition $(**)$ is fulfilled (and implicitly, $(*)$ is not fulfilled), the TOF-tree expansion continues. In the expansion the height value can reach a certain height value h ($h \geq 0$), for example $h = 0$, when this happens, the grid points in Ω form a further type of boundary which we call type III in regard of level height $h = 0$, in short form: III(0). This is also the contour of the object when it is projected onto the ground floor vertically. In general, we may also consider the boundary of a TOF-tree at a given height level h, the respecting boundary boundary type of this will be called III(h). When it is clear in the context, "type III(0)" can be simply written as "type III".

In Figures 4 and 5 we present some schematic examples. Figure 4 shows the case of a single object O_1. The expansion of the TOF-tree associated with O_1 stops at the boundary of Ω, and is therefore of type

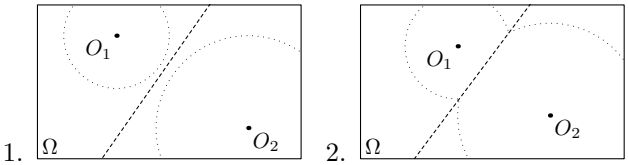

Figure 5: Two further scenarios of multiple objects. Boundary type II is drawn as dashed lines.

I. In the first three sub-figures, type III boundary is visible. The last sub-figure demonstrates an extreme case in which the would-be type III boundary is not within Ω. This means, on the boundary of Ω, the height values are always above floor level, this is the case that Ω is completely covered by the object captured by the TOF sensor.

Figure 5 gives a simple schematic example of two objects O_1 and O_2. Boundaries type I and type III can be identified just as in Figure 4. Due to the presence of two objects, there exist (at least two) local maxima in height value, then the boundary shared by O_1 and O_2 forms a "valley". This is the case of type II.

The distinction of these three boundary types is essential in further processing of the TOF datasets. It is frequently encountered in the so-called tracking problem where the traces/trajectories of the measured objects are to be investigated. An incomplete boundary type III(0), with the exception of the extreme case investigated in sub-figure 4 of Figure 4, signals the situation in which an object enters or leaves the actual measuring environment. (For the aforesaid extreme case, the situation is undecidable from a single frame in the whole time series of the TOF dataset.)

2.4 Reflections on concave objects etc.

A concave object exhibits multiple local maxima and will be represented by more than one TOF-trees consequently. For a schematic example we refer to subfigure 1 of Figure 6. For a rigid object, if not convex, the relevant TOF-trees representing the object have fixed relative positions—in the sense of the grid position of the nodes, especially the root node of the TOF-tree—with each other. Given a time series of TOF data frames, it is possible to distinguish a rigid concave object from multiple independent objects.[1]

1. If the object is not rigid, it is principally not possible to keep a continuous tracking of its movement using TOF data *exclusively*. In the real-world environment, it suffices to assume

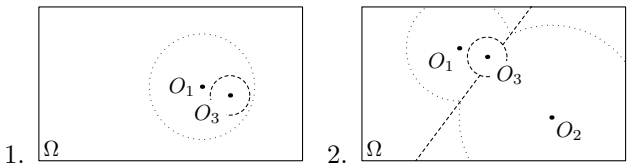

Figure 6: Two schematic examples of concave objects and occlusion. Type III ($h = 0$) and type II boundaries are drawn as dots and dashed lines respectively.

Another phenomenon is the so-called occlusion in which an object is (partly) covered by another/others. An example of this can be seen in sub-figure 2 of Figure 6 where objects O_1 and O_2 are partly covered by object O_3. Occlusion is caused by objects of independent behaviours of mobility, and therefore their relative positions do not stay unchanged in a time series of frames. By this criterion it is possible to distinguish them from concave objects.

3 Object Contour and TOF-Tree

As mentioned earlier, a TOF-tree can be constructed in response to a given height value h: the TOF-nodes contained in the tree have a height value no less than the given h.

After building up the TOF-trees using the data in the original domain Ω, we now consider the problem of retrieving the data from the height value, this is sometimes called "slicing": by "sweeping" an imaginary "cutting" plane parallel to the domain (floor) of a given height through the objects, we get the contour lines of the objects at the given height level. The contour of a given object, in the special case $h = 0$, is equivalent with type III boundary.

On the other hand, the topological size of a TOF-tree is the number of the TOF-nodes it contains. When a TOF-tree is "cut" at a given height level, we may consider that the child node pointers W, N, E and S be set as "`NULL`" to terminate the expansion of the tree, when the height values they de-reference are below the given one. Thus it is obvious to conclude

Theorem. *The size of a TOF-tree, in number of the TOF-nodes, equals the area enclosed by the contour line, in number of the grid points in the original domain Ω, at arbitrary height h.*

Proof. By recursion. To simplify the notation, we write `tp` as the TOF-node pointer de-referencing the TOF-tree, `tp->h` as the height associated with this node. If `tp->h` $< h$, then both sides equal 0. If `tp->h` $= h$, then both sides equal 1. Otherwise, the left side equals 1 plus the sum of the sizes of the TOF-trees de-referenced by `tp->`W, `tp->`N, `tp->`E and `tp->`S, if these pointers are not "`NULL`". These numbers are consistent with the increment (in number of the grid points) of the contour area in the four relevant directions. □

We give an implementation as recursive function to calculate the size of a TOF-tree above a given height h:

```
procedure compute_size:
parameter: level height h
  if tp->h < h
    return 0;
  else
    call compute_size on non-NULL pointers
      tp->W, tp->N, tp->E, tp->S with h;
    sum the return values;
  fi
return the sum plus 1
```

In a similar way, the boundary information of a given TOF-tree at a given height level h can be retrieved:

```
procedure compute_boundary:
parameter: level height h
  if tp->h < h
    save "type III(h)";
    return;
  fi

  if W not NULL
    call compute_boundary on W with h;
  else
    if node de-referenced by tp on ∂Ω
      save "type I";
    elseif node of neighbouring position with
      (Δx, Δy) = (−1, 0) belongs to another
      TOF-tree T′
      save "type II(T′)";
```

that the objects are "roughly" rigid, i. e. in certain regions-of-interest. When the TOF ranging system is deployed as a monitoring system to record the trajectories of the moving human beings (or other objects), the former assumption is still safe to a large extent, the region-of-interest could be chosen as the head of shoulders. The shape of these can be fully captured by the contours at different height levels, see the next section.

```
    fi
  fi

  if N not NULL
    call compute_boundary on N with h;
  else
    if node de-referenced by tp on ∂Ω
      save "type I";
    elseif node of neighbouring position with
      (Δx, Δy) = (0, 1) belongs to another
      TOF-tree T'
      save "type II(T')";
    fi
  fi

  if E not NULL
    call compute_boundary on E with h;
  else
    if node de-referenced by tp on ∂Ω
      save "type I";
    elseif node of neighbouring position with
      (Δx, Δy) = (1, 0) belongs to another
      TOF-tree T'
      save "type II(T')";
    fi
  fi

  if S not NULL
    call compute_boundary on S with h;
  else
    if node de-referenced by tp on ∂Ω
      save "type I";
    elseif node of neighbouring position with
      (Δx, Δy) = (0, −1) belongs to another
      TOF-tree T'
      save "type II(T')";
    fi
  fi

return
```

4 Monitoring System: An Application

In this section we present a simple application of the TOF data representation. We have a monitoring system equipped with a SwissRanger4000 TOF sensor[2], similar to Figure 1. The data will be processed on the basis of the time frames. See Figure 7 for some raw data samples.

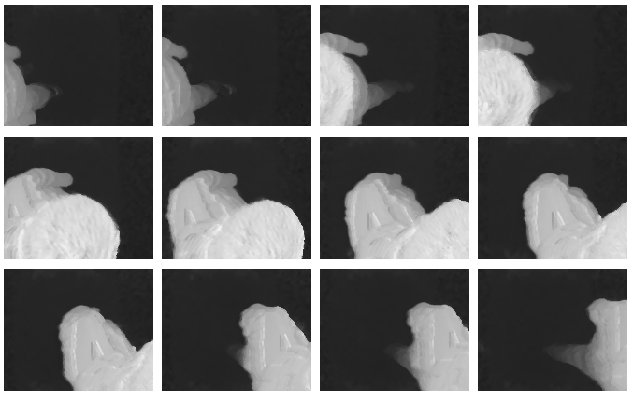

Figure 7: The range information of a walking person detected by a TOF sensor, recorded in a time sequence of twelve frames.

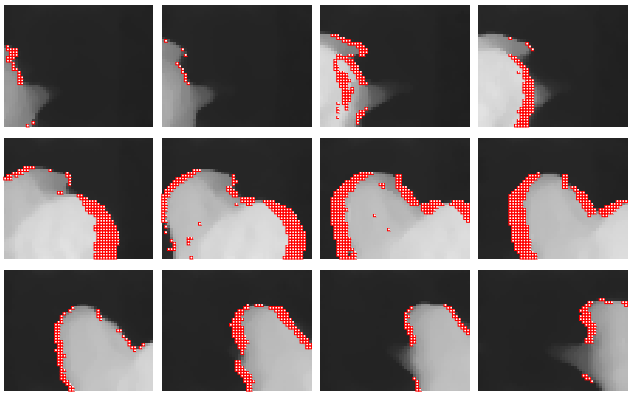

Figure 8: The smoothed data of Figure 7; pixels with extremely low confidence measure are drawn in red.

In addition to the usual range information, the SwissRanger sensor series provide us with the information of the so-called confidence measure which further indicates how credible the range measurements are. Unfavourable light reflections and high moving speed are two common troublesome factors. Therefore, measurements with very low confidence measure (below a certain threshold) should be corrected or ignored in the further processing. See Figure 8 for an example. We leave the pixels with extremely low confidence measure unfilled with range information. (The correction of these measurements may be realized by interpolation. However, since sometimes it can happen that very

2. Produced by Mesa Imaging AG (Switzerland), homepage `http://www.mesa-imaging.ch`.

large areas are composed of pixels with extremely low confidence measure, this technique is clumsy, too.)

We then call the procedure `main` (see § 2.2), and get a series of TOF-trees. We recall the fact that only in the ideal case a perfectly convex object is represented by a single TOF-tree; in most cases the detected object will be understood as multiple connecting TOF-trees.[3] See Figure 9.

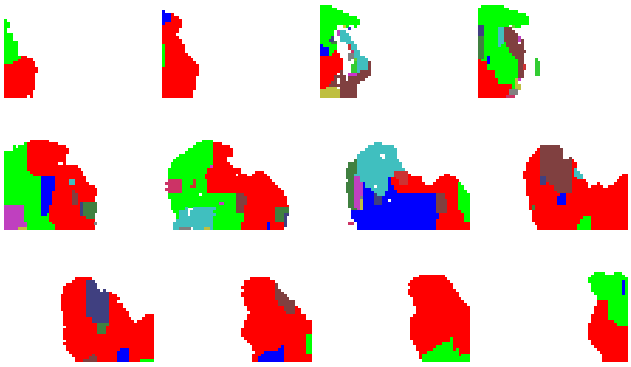

Figure 9: The expanded TOF trees in the time frames, drawn in different colours. Note the small white wholes in the coloured regions which denote the pixels with extremely low confidence measure.

Now the question is, given a set of TOF-trees $T_1, \ldots, T_m$, how to use them to re-construct the detected objects $O_1, \ldots, O_n$? That is, we need an index permutation

$$(1,1), \ldots, (1,k_1), \quad \ldots, \quad (n,1), \ldots, (n,k_n) \qquad (8)$$

of $1, \ldots, m$, so that

$$O_1 = T_{(1,1)} + \cdots + T_{(1,k_1)};$$
$$\ldots$$
$$O_n = T_{(n,1)} + \cdots + B_{(n,k_n)}.$$

(By "+" we mean that multiple TOF-trees can be combined into a single object.)

On the other hand, TOF-trees belonging to the same object must share directly (or indirectly through other TOF-trees) a common boundary (cf. procedure `compute_boundary`, § 3). By this criterion the problem of index allotment (8) can be greatly simplified. For Figure 9, a possible (and reasonable) solution is given in Figure 10, where every single object is given a different gray colour. In three (the 3rd, 4th and 7th) frames, there is a second (very small) object drawn in a darker gray colour. Introducing further criteria (size, shape etc.) we may easily exclude objects of this kind (caused by inaccurate measurements).

Figure 10: Combining the TOF-trees in Figure 9.

Finally with the type I boundary information, the current experimental system can be deployed as an electronic counter for objects entering and leaving the supervised area.

5 Further Discussion

The current paper discusses a new tree-like structure of TOF data representation. This structure preserves the complete information in the original dataset and can be therefore considered as "lossless". By traversing the TOF-trees the contour line information of the original objects measured by the TOF sensor can be retrieved (roughly speaking, in a similar way of comparing Lebesgue integration with the Riemann variant.) These generated contour lines can be used in further processing of object detection, recognition and tracking, since by the local change and evolution of these contour lines the geometric characteristics of the original measured object are completely captured.

The contour generated using the TOF data structure is physically identical with that computed by other simple algorithms (see Appendix). However, the latter is unstructured, that is, composed only of a set of raw pixels, and there is no direct and efficient way to compute the enclosed region of the contour. In the future work, we will investigate better solutions for object detection and tracking problems using the structured contours.

3. In the sense that some of their leaf nodes are neighbours with each other in Ω.

A Simplified Contour Line Visualization

In the current paper we introduced a tree structure for the TOF datasets. A convex object captured by the measuring TOF sensor can be thereby represented as a tree with a root node manifesting the local maximum. It is sometimes necessary to visualize the object contour at a given height level. This contour contains the sets of the nodes which can be retrieved by traversing the TOF-tree, at the given height.

The standard algorithm of contour generation for volume data is the marching cube algorithm [11]. This algorithm applies a sophisticated interpolation scheme to construct the contour (in the three-dimensional case, surface) with high resolution.

For the purpose of monitoring the acquired data, our task is less complicated: that is, since the resolution is already set (and presumably bottle-necked) by the TOF sensor technology, we concentrate here on a fast computation. From a hardware perspective, the visualization of the contour line can be easily carried out by a scanline-like algorithm. Let a scalar-valued function H on a two-dimensional grid $\Omega = \{1, \ldots, n_x\} \times \{1, \ldots, n_y\}$ be given. Let n_x and n_y denote the width and the height of the grid respectively. We further request that the integers n_x and n_y be larger than 2.

The idea of our algorithm is to construct a picture C composed of $(n_x-1)(n_y-1)$ points which shows the changes of the scalar values on Ω. (The size change of C from H should make no substantial impact on the visualization, if H is not composed of too few pixel elements.) First, given a level l height, we build a bitarray of $B_{x,y}$, for $x = 0, \ldots, n_x-1$, $y = 0, \ldots, n_y-1$

$$B_{x,y} = \begin{cases} 1 & \text{if } H_{x,y} \geq l, \\ 0 & \text{else.} \end{cases}$$

In the next step, we build up C row by row from $y = 0$ to $n_y - 2$. In each row, the index x goes from 0 to $n_x - 2$. The value of $C_{x,y}$ is then decided by $B_{x,y}$, $B_{x,y+1}$, $B_{x+1,y}$ and $B_{x+1,y+1}$. There are $2^4 = 16$ possible combination of these bit values. We associate each of them with an integer number of state s in the range $0, \ldots, 15$:

$$s_{x,y} \Leftarrow 0\,, \tag{9.0}$$
$$s_{x,y} \Leftarrow s_{x,y} + \quad B_{x,y} \cdot 1\,, \tag{9.1}$$
$$s_{x,y} \Leftarrow s_{x,y} + \quad B_{x,y+1} \cdot 2\,, \tag{9.2}$$
$$s_{x,y} \Leftarrow s_{x,y} + \quad B_{x+1,y} \cdot 4\,, \tag{9.3}$$
$$s_{x,y} \Leftarrow s_{x,y} + B_{x+1,y+1} \cdot 8\,. \tag{9.4}$$

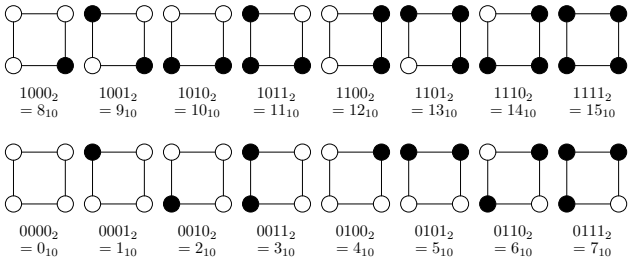

Figure 11: The 16 possibilities for $s_{x,y}$. The upper-left, lower-left, upper-right and lower-right circles stand for the grid positions of (x, y), $(x, y + 1)$, $(x + 1, y)$ and $(x + 1, y + 1)$ respectively. The circles are filled where the B values are 1.

We observe that only the first row element ($x = 0$) needs all the four additions (9.1)–(9.4); for the other elements, that is, $x = 1, \ldots, n_x - 2$, (9.0)–(9.2) are equivalent (in the high programming languages) with

$$s_{x,y} \Leftarrow s_{x-1,y} \ll 2\,,$$

where "$\ll$" denotes the bitwise shift operator.

It is obvious that no local contour will exist if the four grid positions have the same B value ($s_{x,y} = 0, 15$). Otherwise, we suggest that the pixel in C at (x, y) be given an indexed colour value, that is,

$$C_{x,y} = \begin{cases} 1 & s_{x,y} = 1, 2, 4, 7, 8, 11, 13, 14; \\ \frac{2}{3} & s_{x,y} = 3, 5, 10, 12; \\ \frac{1}{3} & s_{x,y} = 6, 9; \\ 0, & s_{x,y} = 0, 15. \end{cases} \tag{10}$$

In (10), the cases $s_{x,y} = 1, 2, 4, 7$ are the reverse of those $s_{x,y} = 14, 13, 11, 8$ respectively (the same is with $3, 5$ to $12, 10$ and 6 to 9 and 0 to 15). In these cases, a contour line should go through the current pixel in C; and $C_{x,y}$ is given the full colour. The cases $s_{x,y} = 3, 5, 10, 12$ is less ambiguous than $s_{x,y} = 6, 9$ and $C_{x,y}$ is given a higher value. (10) can be further simplified to be

$$C_{x,y} = \begin{cases} 1 & c_{x,y} = 1, \ldots, 14; \\ 0 & s_{x,y} = 0, 15, \end{cases}$$

if C is to be a bitmap. An example of this variant is presented in Figure 12.

We wish to point out that since the generated contour line is composed of unstructured two-dimensional grid points, there is no direct way to compute the enclosed area (that is, size) of the contour which should be applied as an important means for object recognition in the later processing.

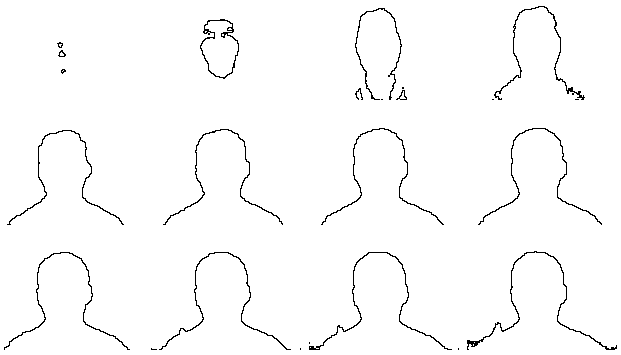

Figure 12: An example of contour lines of different level heights. Further refinement in regions-of-interest (respecting the level height) is possible. The images reveal that the detected object is not perfectly convex.

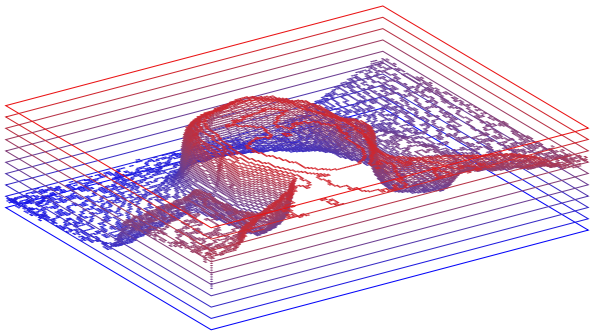

Figure 13: The contour lines of different level heights in a three-dimensional frame. The colour evolution stands for the change of level heights.

References

[1] P. J. Besl. Active, optical range imaging sensors. *Machine Vision and Applications*, 1(2):127–152, 1988.

[2] A. Bevilacqua, L. Di Stefano, and P. Azzari. People tracking using a Time -of-Flight depth sensor. In *Proceedings of the IEEE International Conference on Video and Signal Based Surveillance (AVSS '06)*, page 89, 2006.

[3] P. Breuer, C. Eckes, and S. Müller. Hand gesture recognition with a novel IR time-of-flight range camera—a pilot study. In A. Gagalowicz and W. Philips, editors, *Computer Vision/Computer Graphics Collaboration Techniques. Third International Conference, MIRAGE 2007, Rocquencourt, France, March 28–30, 2007. Proceedings*, volume 4418 of *Lecture Notes in Computer Science*, pages 247–260. Springer-Verlag Berlin Heidelberg, 2007.

[4] J. H. Cho, S.-Y. Kim, Y.-S. Ho, and K. H. Lee. Dynamic 3D human actor generation method using a time-of-flight depth camera. *IEEE Transactions on Consumer Electronics*, 54(4):1514–1521, 2008.

[5] D. Falie and V. Buzuloiu. Wide range Time of Flight camera for outdoor surveillance. In *Microwaves, Radar and Remote Sensing Symposium, 2008 (MRRS 2008)*, pages 79–82, 2008.

[6] M. Frank, M. Plaue, H. Rapp, U. Köthe, B. Jähne, and F. A. Hamprecht. Theoretical and experimental error analysis of continuous-wave time-of-flight range cameras. *Optical Engineering*, 48(1):013602, 2009.

[7] M. Haker, M. Böhme, T. Martinetz, and E. Barth. Geometric invariants for facial feature tracking with 3D TOF cameras. In *International Symposium on Signals, Circuits and Systems, 2007 (ISSCS 2007)*, volume 1, pages 109–112, 2007.

[8] D. W. Hansen, M. S. Hansen, M. Kirschmeyer, R. Larsen, and D. Silvestre. Cluster tracking with Time-of-Flight cameras. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, 2008.

[9] J. U. Kuehnle, Z. Xue, M. Stotz, J. M. Zoellner, and R. Dillmann. Grasping in depth maps of time-of-flight cameras. In *Robotic and Sensors Environments, 2008. ROSE 2008. International Workshop on*, pages 132–137, 2008.

[10] R. Lange. *3D Time-of-flight distance measurement with custom solid-state image sensors in CMOS/CCD-technology.* PhD thesis, Universität Siegen, 2000.

[11] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[12] X. Luan. *Experimental investigation of photonic mixer device and development of TOF 3D ranging systems based on PMD technology.* PhD thesis, Universität Siegen, 2001.

[13] D. Nitzan, A. E. Brain, and R. O. Duda. The measurement and use of registered reflectance and

range data in scene analysis. *Proceedings of the IEEE*, 65(2):206–220, 1977.

[14] H. Rapp, M. Frank, F. A. Hamprecht, and B. Jähne. A theoretical and experimental investigation of the systematic errors and statistical uncertainties of Time-Of-Flight-cameras. *International Journal of Intelligent Systems Technologies and Applications*, 5(3/4):402–413, 2008.

[15] R. Schwarte, H.-G. Heinol, Z. Xu, and K. Hartmann. New active 3D vision system based on RF-modulation interferometry of incoherent light. In D. P. Casasent, editor, *Proceedings of SPIE, Intelligent Robots and Computer Vision XIV: Algorithms, Techniques, Active Vision, and Materials Handling*, volume 2588, pages 126–134, 1995.

[16] S. Soutschek, J. Penne, J. Hornegger, and J. Kornhuber. 3-D gesture-based scene navigation in medical imaging applications using Time-of-Flight cameras. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008 (CVPRW '08)*, 2008.

[17] T. Spirig, P. Seitz, O. Vietze, and F. Heitger. The lock-in CCD—two-dimensional synchronous detection of light. *IEEE Journal of Quantum Electronics*, 31(9):1705–1708, 1995.

[18] R. Tanner, M. Studer, A. Zanoli, and A. Hartmann. People detection and tracking with TOF sensor. In *Proceedings of the IEEE Fifth International Conference on Advanced Video and Signal Based Surveillance (AVSS '08)*, pages 356–361, 2008.

[19] J. W. Weingarten, G. Gruener, and R. Siegwart. A state-of-the-art 3D sensor for robot navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004 (IROS 2004)*, volume 3, pages 2155–2160, 2004.