Robust and Fast Teat Detection and Tracking in Low-Resolution Videos for Automatic Milking Devices

Matthew van der Zwan[1], Alexandru Telea[1]
[1] *Institute Johann Bernoulli, University of Groningen, Nijenborgh 9, Groningen, the Netherlands*
{m.a.t.van.der.zwan, a.c.telea}@rug.nl

Keywords: time-of-flight cameras, video tracking, vision for robotics, automatic milking devices

Abstract: We present a system for detection and tracking of cow teats, as part of the construction of automatic milking devices (AMDs) in the dairy industry. We detail algorithmic solutions for the robust detection and tracking of teat tips in low-resolution video streams produced by embedded time-of-flight cameras, using a combination of depth images and point-cloud data. We present a visual analysis tool for the validation and optimization of the proposed techniques. Compared to existing state-of-the-art solutions, our method can robustly handle occlusions, variable poses, and geometries of the tracked shape, and yields a correct tracking rate for over 90% for tests involving real-world images obtained from an industrial AMD robot.

1 Introduction

Scale economies in the dairy industry increasingly shift manual labor to robotic devices. One such development is the advent of automatic milking devices (AMDs): Given a stable populated with cows, AMDs use vision devices to locate cows in the stable, reach under the cow *e.g.* with a mechanical arm, locate the udder and teats, and finally track the teats in order to couple a suction device to each teat to collect milk (LMI12; Sco13; MES14; Wes09; Hun06).

Vision devices used in AMDs must be small, shock-resistant, able to work in the dim lighting of a stable, and relatively cheap (Wes09). Separately, they have to operate in near-real-time to cope with the cow's motion, handle occlusions, locate features of interest with sub-centimeter precision, and work fully automatically. In recent years, time-of-flight (ToF) range cameras have become increasingly popular as the core building-block of such systems (MES14; Sco13). Given a 3D scene, a ToF camera produces a per-pixel depth map of the occluding surfaces found in front of the camera, with a relatively high frame-rate (24 frames per second (fps)). Compared to traditional stereo vision (Hun06) or laser-scanning (HAP05) devices, ToF cameras are less sensitive to lighting conditions and dust specks, generate a full depth-map with depth data at each pixel, are shock-resistant, come in compact form-factors, need no delicate calibration, and provide many 3D vision functions in embedded software (DPC10; DDL10). Hence, high hopes are placed on using ToF cameras in industrial AMD applications. However, their relatively low spatial resolution creates new challenges that are not typically handled by mainstream vision algorithms.

In this paper, we present a vision-based solution for AMD robots built using ToF cameras. We focus on the robust, accurate, automatic, and fast detection and tracking of the cow teats, *i.e.*, the last step of the milking process. We present the entire vision pipeline from depth image acquisition, feature extraction and filtering, and udder tracking, and detail a simple and efficient implementation thereof. We then present both qualitative and quantitative validation of our system in an industrial context.

In the following, Section 2 overviews graphics and vision methods relevant to our goal. Section 3 describes our solution. Section 4 presents the results. Section 5 presents a visual analysis tool developed for validating the quality of our tracking results. Section 6 concludes the paper.

2 Related Work

We next overview computer vision methods for feature detection and tracking for natural deformable moving objects. Given our application context, we focus only on methods which can comply with all our requirements: (1) automation, (2) low-cost, (3) robustness, (4) low computational complexity, and (5) implementation simplicity.

Marker-based tracking: A standard solution to 3D shape tracking is to mark salient keypoints thereof by textures which can be easily detected in a 2D image. If correspondences can be robustly found between stereo image pairs, stereo vision solutions can then be used to compute 3D positions of such fiducial marker-pairs by triangulation (LSG08). Marker-based solutions are fast, simple to implement, and quite robust, but not applicable to our context, as industry guidelines discourage the placement of markers on cow teats. Monocular marker-based tracking solutions also exist, but they are considerably more complex and computationally expensive for non-rigid, complicated, shapes (AT06; ST01).

Marker-less tracking: Marker-less tracking solutions typically find keypoints at the naturally salient image *features* (corners, edges, or edge crossings), *e.g.* using SIFT (Low04) and SURF (BETG08) descriptors. For very low-resolution texture-less images, like our cow udders, the robustness of such approaches is very low. *Template*-based methods try to find pre-defined templates (small predefined patterns) in the image, using statistical approaches such as correlation (SW99). Deformable dynamic templates (DDTs) can search for more complex configurations, by adapting a deformable template model to fit image silhouettes (YHC92). However, DDTs require well-chosen energy functions, initialization points, and high-resolution images, and are too computationally expensive for our real-time context.

3D reconstruction: Having a depth camera, one can reconstruct the 3D visible-object surface from the depth field, which comes as a 3D point cloud. From this surface, teat tips could be found at maxima of mean or Gaussian curvature, similar to polyp detection approaches used in medical science, see *e.g.* (CFFD09). Yet, reconstructing clean, differentiable, 3D surfaces from point clouds given by ToF cameras is very challenging. Most surface reconstruction methods in existence have constraints on the sampling density, complexity, connectivity, and water-tightness of the sampled surface, and are also quite slow (KBH06; HDD*92; DG04; DLRW09; KJT14). Additionally, such methods cannot find features (like our cow teats) which are occluded in the input image.

Specific solutions: Many techniques have been proposed and fine-tuned to find and track features in moving natural shapes such as humans or parts thereof, *e.g.* faces or hands. However, such techniques are not directly usable for cow udder morphologies, as they have other shape priors. In the milk industry, very few solutions exist and have been implemented into AMD robots (LMI12; Sco13; MES14; Wes09; Hun06). All these solutions assume a *fully* unoccluded *and* zoomed-in bottom or side view of the udder, given by a *fixed* robot arm that places the camera close to the udder, and given a cow constrained in a small space, to limit motion. In contrast, we do not assume that our robot is initially correctly placed close to the cow udder, nor do we assume that the cow cannot move *vs* the robot.

3 Method

As input device, we use a SwissRanger SR4000 ToF camera (Mes10), which has one of the best quality-price ratios to date (DPC10; DDL10). The camera gives a 24-fps stream $\{I_i\}$. Each frame I_i has two 176×144 pixel images (A_i, D_i). A_i is a standard amplitude (luminance) image. D_i is a depth map, where each pixel stores the distance, in millimeters, to the closest occluding object, with an accuracy of a few millimeters for distances up to roughly 1 meter. The camera also delivers a point-cloud $P_i = \{\mathbf{p}_j\}$ with the world-space locations of all visible-surface points in frame i. The camera is rigidly mounted on a robot which can reach the zone under the cow to be milked. As outlined in Sec. 1, we focus on the milking stage, where the camera is already under the cow, roughly between the legs and looking towards the tail. The cow stands upright, so its legs and teats appear as vertically-oriented shapes in the image (Fig. 2 a).

Our solution is split in two parts: A *detection* module finds teats from the image-and-point-cloud $\{I_i, P_i\}$ of the current frame i. Next, a *tracking* module integrates this information over time, handling occlusion and other model priors (Fig. 1). The two modules are described below.

3.1 Detection

To find teats in the a frame I_i, we can use one or several of the fields A_i, D_i, and P_i given by the camera. After extensive studies, we found that our images A_i are too low-contrast and noisy to be useful, due to poor lighting in the stable. Hence, we use only the depth image D_i and point cloud P_i for teat detection. As the images D_i still contain a small noise amount, caused by dust specks floating in the stable, we first apply a median filter to them. The filtered images $\tilde{D}_i$ are almost noise-free and show little blurring (Fig. 2 b).

We next propose two separate methods to find teats from filtered depth images $\tilde{D}_i$ (Sec. 3.1.1) and point clouds P_i (Sec. 3.1.2) respectively.

3.1.1 Template-based detection

Our first teat-detection method treats $\tilde{D}_i$ as regular grayscale images. To find teats, we use a template-matching technique consisting of four steps:

a. Edge detection: First, we find edges in the depth image $\tilde{D}_i$, using a gradient-magnitude filter $\|\nabla \tilde{D}_i\|$. The result E_i of this filter highights values where $\tilde{D}_i$ has strong jumps,

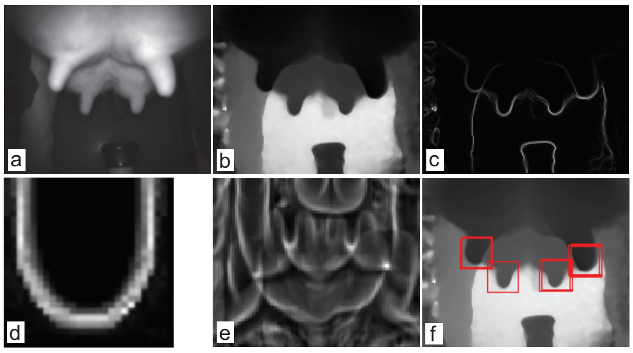


Figure 2: A frame from a typical video sequence. (a) Amplitude image A, with visible udder and four teats. (b) Filtered depth image $\tilde{D}$. (c) Edges E in depth image. (d) Canonical template image T. (e) Correlation image C_i. (f) Matches found (Sec. 3.1.1).

which are the silhouettes of shapes in our depth image. Figure 2c shows a typical edge-image E_i. Silhouettes of the cow teats and limbs are clearly visible in this image.

b. Template matching: To find teats, we use a template-matching approach. For this, we first compute the silhouette (edge-image) of a typical U-shape of a teat. We call this image a template T (Fig. 2d). Next, we use a normalized correlation coefficient (NCC) approach (SW99) to find instances of T in the edge-image E_i, by convolving E_i with T using the Fast Fourier Transform provided by OpenCV (Ope14). Besides speed, the advantage of NCC becomes apparent if we notice that a teat could be close by in front of a leg, or far away from the background (stable wall), resulting in edges of highly different intensities. NCC efficiently corrects for edge-intensity differences in both E_i and T, which matches our goal to capture the *shape* of objects described by the edges, rather than objects' relative *positions* with respect to the background.

The NCC computation yields a correlation image C_i where each pixel $C_i(x,y) \in [0,1]$ tells how well T matches the edge-image E_i at pixel (x,y), with higher values encoding better matches (Fig. 2e). The maxima of C_i are regions where T matches best. Thus, we can find potential teat locations, or *matches* t_i, by finding the N largest local maxima of C_i. For all our tests, we fixed $N = 6$. We also tried the option of upper-thresholding C_i with a fixed value. However, this yielded between none and tens of matches per image C_i, so we prefer the first approach (N-best selection). For each match $t_i = (x,y,z)_i$, we store its 2D position x_i, y_i in image-space, and also its depth from camera z_i.

The above template matching method is not scale-invariant – it only finds areas in C_i which match the template T at T's own scale. Figure 3a shows this: Here, we miss the front-right teat, which is about twice larger than the template. Still, the range of teat sizes (in image-space) is bounded by the fixed size of the cow and the positioning of the robot which is never more than 1.5 meters away from the udder. Analyzing several production videos, we determined that teats range between $1/30$ and $1/6$ of the image-width, *i.e.* between $T_{min} = 10$ and $T_{max} = 30$ pixels. To find teats in this scale-range, we use the NCC method described above with six template sizes T_i, $1 \le i \le 6$, uniformly distributed between T_{min} and T_{max}. This enables us to find small and large teats (Fig. 3b).

c. Match selection: We next collect all matches t_i from all different scales T_j, after which we apply the N-best selection procedure outlined above for the single-scale case.

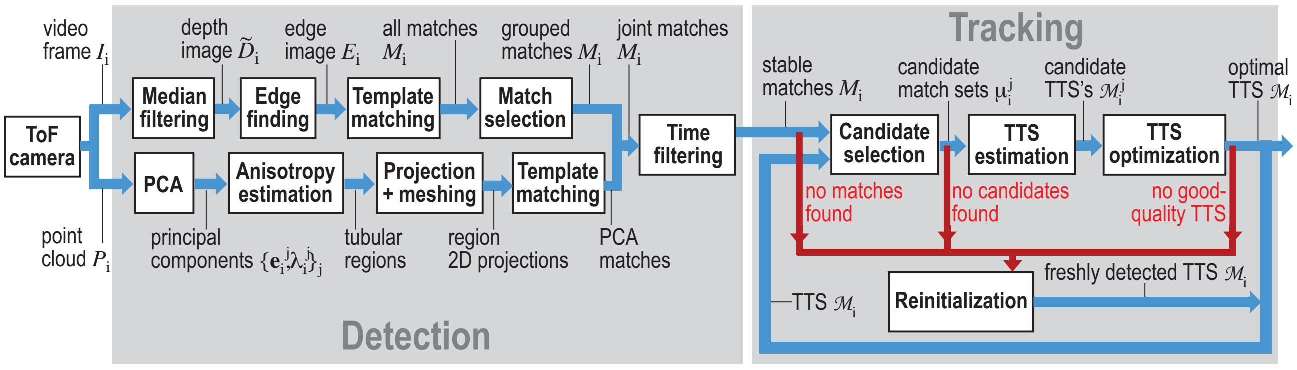


Figure 1: Proposed teat detection-and-tracking pipeline. Blue arrows indicate data streams from the input information acquired from the ToF camera to the output of four tracked teats $\mathcal{M}_i$. Red arrows indicate the control-flow for tracking reinitialization (see Sec. 3.2.5).

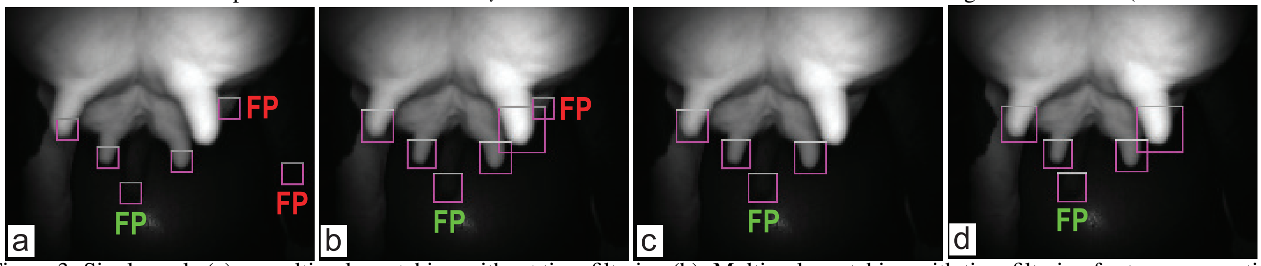


Figure 3: Single-scale (a) *vs* multiscale matching without time filtering (b). Multiscale matching with time filtering for two consecutive frames (c,d). Matches are indicated by rectangles, with *'FP'* showing false-positives. Red-marked FP's are removed by time filtering.

When using multiple scales, we can find two (or more) matches t_i and t_j, for two scales T_a and T_b, whose 2D positions (x_i, y_i) and (x_j, y_j) are close enough to represent the same teat. We consider such matches to be duplicates when the center of the inscribed circle in T_i falls in the inscribed circle of T_j or vice versa (Fig. 4). From any set of duplicates, we only keep a single match for further processing.

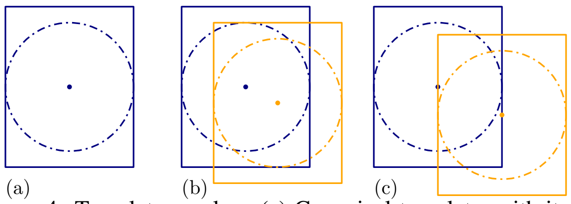


Figure 4: Template overlap. (a) Canonical template, with its inscribed circle and circle-center. (b) Two overlapping templates. (c) Two non-overlapping templates (see Sec. 3.1.1).

d. Match time filtering: Our teat-detection can find a teat where none actually exists. These are areas where the edge-structure in E_i has U-shapes similar to our templates, *e.g.* around the cow's tail-tip, or around some leg muscle structures. We call these *false positives* (FPs). Many such FPs appear only for a very few consecutive frames. In contrast, *true positives* (TPs) are visible for longer periods, until they get occluded or drift out of the camera view. We remove FPs by time filtering, as follows. Let $M_i = \{t_j\}$ be the set of matches found in frame i of our input stream. Given the sequence $\{M_k\}_{i-K<k<i}$ of matches found in the previous K frames, we remove from M_i those matches which are not visible in at least τ of the last K frames. This means that we have a delay (of K frames) in detecting teats. Choosing a low value for K keeps this delay small, as our camera operates at 24 fps. Fixing $K = 5$ and $\tau = 2$ frames effectively removed most FPs while keeping most TPs. Figure 3 shows this. The three FP matches marked red in images (a,b) are removed in image (c) by time filtering. The remaining FP, marked green, which corresponds to the cow tail, is however not removed, as this structure persists in several frames. We show next in Sec. 3.2 how such remaining FPs are removed by using tracking.

3.1.2 PCA Based Detection

The template-based method described above works well when teats are roughly vertical and parallel to the camera plane, *i.e.*, when the angle α between a teat's symmetry-axis and the camera plane is below roughly 10°. For such angles, the difference between the edge profiles of the vertically-aligned U structures in our templates T_i and those of actual teats in E_i is small enough to yield strong matches.

For larger angles α, template matching has difficulties. In such cases, the teats' silhouettes in E_i differ too much from the ones in our templates. We find two sub-cases here. First, a teat could be rotated *into* the camera-plane. To address this, we could use a solution akin to the one dealing with scale-variance (Sec. 3.1.1), *i.e.*, create a family of templates T_i^{rot} rotated in the camera plane. The second case occurs when teats are rotated *out* of the camera plane (see *e.g.* the two front teats in Fig. 5a). In such cases, the teat silhouette changes from a U-shape to an ellipse or parabola sector. We verified that rotation invariance cannot be dealt with in this case by using additional templates, as such shapes have too high an edge variability in the depth image.

We propose next a method to handle both rotation variance cases. Teats have a roughly cylindrical shape, which means that locally there is a clearly-oriented structure in the depth-image data. This structure can be lost in the projected edge image. To find such structures, consider a ball B of fixed radius, roughly 4 cm in world space, corresponding to the average half-length of a cow teat. We next center B consecutively at all locations $\mathbf{p}_i$ of the point cloud P_i delivered by the ToF camera, and compute the eigenvectors $\mathbf{e}_i^j$, $1 \leq j \leq 3$, and corresponding eigenvalues $\lambda_i^1 \geq \lambda_i^2 \geq \lambda_i^3$ of the covariance matrix of all points in $P_i \cap B$. Figure 5b illustrates this, by showing the direction of the major eigenvector $\mathbf{e}_i^1$ by color coding – red,

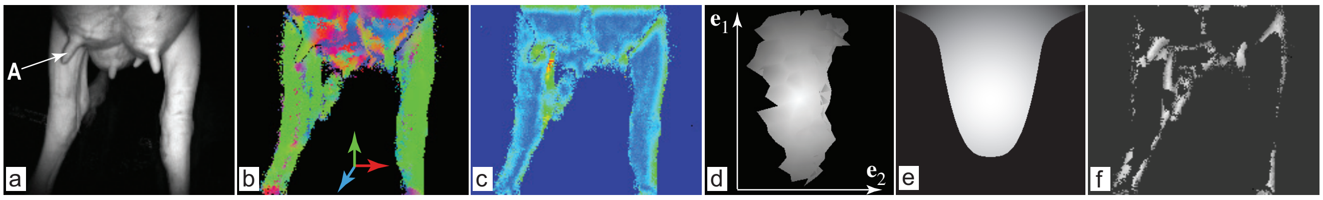


Figure 5: PCA-based detection. (a) Amplitude image. (b) Direction of major eigenvector. (c) Elongation values. (d) 2D projected neighborhood of a point marked *'A'* in the first image. (e) Template used for matching. (f) Correlation image (see Sec. 3.1.2).

green, and blue show eigenvectors $\mathbf{e}_i^1$ aligned with the x, y, and z axes respectively. Next, we find tube-shaped regions $P_i \cap B$ by computing the so-called linear anisotropy or elongation $c = \lambda_i^1/\lambda_i^2$ (WPG*97), and selecting only regions for which $c > 1.5$. These are potential teat locations. Figure 5c shows the elongation c with a rainbow colormap (blue=low, green=medium, red=high values). As visible, areas around teats are green, as they have a quite high elongation. Finally, we project such regions onto the plane defined by $(\mathbf{e}_i^1, \mathbf{e}_i^2)$. If a teat exists around $\mathbf{p}_i$, $\mathbf{e}_i^1$ should match its symmetry axis (given the teat's cylindrical shape), so the resulting 2D projection should show a vertical teat shape, like the ones in our templates. This corrects for the rotational variance. Additionally, we scale the 2D projection by the value of λ_i^1 divided by the height of the template T, which takes care of the scale variance. As such, we can now directly use our *single-scale* template matching to find rotationally-invariant teat matches in the projected images.

Given camera resolution limitations, the 2D projections of cloud points $P_i \cap B$ can yield very sparse point sets. To match these with a teat shape, we need a compact image. To create this, we render a quad mesh with points $P_i \cap B$ as vertices and connectivity given by the raster structure of I_i. Mesh vertices are colored by their depth to the projection plane. Figure 5d shows such a 2D projection for the neighborhood $P_i \cap A$ around point A in Fig. 5a. Such images typically have jagged edges, given (again) the low resolution of our cloud P_i clipped by the ball B. Computing edges on such images yields a high amount of noise, which makes our edge-template matching not robust. We solve this by a template matching using the *full* image of a teat, where pixel grayscale values indicate depth (Fig. 5e). The correlation result (Fig. 5f) emphasizes elongated regions whose maxima correctly capture positions of rotated teats.

The matches yielded by the PCA detection are merged with the ones delivered by the template-based detection (Sec. 3.1.1) to yield the final match-set M_i. This way, we increase the chances of capturing all matches visible in a single image. We use this joint match-set M_i to robustly detect and track all four teats, as described next.

3.2 Tracking

Our teat detection technique (Sec. 3.1) successfully finds about 90% of the visible teat tips in our typical videos. Yet, detection still suffers from two main problems:

Occlusion: In frames where one or more teats are occluded from the camera viewpoint(by cow limbs, other teats or robot parts), detection obviously fails to find such teats. As our AMD robot needs finding *all* teats in each frame to start the milking process, we must locate occluded teats too.

Robustness: Even for frames with no apparent teat occlusion, two additional teat detection problems exist. First, certain teat configurations are not detectable, due to resolution limitations of the ToF camera. We call these *false negatives* (FNs). Some FNs can be removed by relaxing the detection method's parameters, to accept more image structures as teats. However, this makes detection sensitive to small-scale noise, which next creates matches at spurious image locations, *i.e.*, yields unwanted *false positives* (FPs).

To reduce the amount of FPs and FNs described above, we need to use additional information not present in single video frames. For this, we choose a *model-based* approach: We define a parameterized model that describes the intrinsic variability (priors) of shape, size, orientation, and dynamics (change in time) of the *entire* set of four teats that a typical cow has. At frame i, this set of teats, called the tracked teat-set (TTS), is a quadrilateral $\mathcal{M}_i = \{\mathbf{p}_j \in \mathbb{R}^3\}$, $1 \leq j \leq 4$, whose vertices $\mathbf{p}_j$ are ordered counter-clockwise with $\mathbf{p_0}$ being the near-left teat from the camera viewpoint. To compute $\mathcal{M}_i$, we use a *tracking* procedure that fits the TTS $\mathcal{M}_{i-1}$ computed from frame $i-1$ to the match-set M_i detected in the current frame i, subject to our model's geometric and dynamic constraints. Figure 6 shows the TTS quad tracked in three frames in a video of several minutes. Our tracking proposal is detailed next.

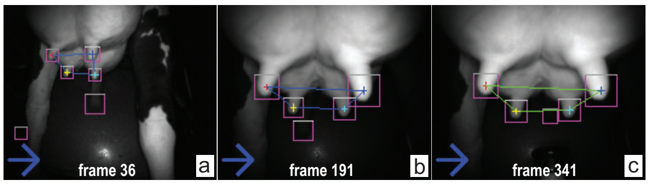


Figure 6: Three frames from a tracking sequence with matches shown as rectangles and TTS shown as a 3D quad (see Sec. 3.2).

3.2.1 Candidate matches

Key to tracking is finding how vertices of the TTS $\mathcal{M}_{i-1}$ from the previous frame correspond to teat-matches in M_i found in the current frame. To find these correspondences, we first construct a collection $S = \{\mu_i^j\}_j$ of all *candidate-match sets* $\mu_i^j \subset M_i$ each having between one and four matches as elements. We sort this sequence decreasingly on the number of elements $|\mu_i^j|$ in each candidate-match set (CMS), and then try to construct a candidate TTS $\mathcal{M}_i^j$ from each such μ_i^j, in increasing j order. This ordering models our preference to fit our TTS to more, rather than to fewer, matches in the current frame, so as to use most of the information present in that frame.

3.2.2 Correspondence finding

Given a CMS μ_i^j, we find its point-to-point correspondence with the previous TTS $\mathcal{M}_{i-1}$ as the set of point-pairs $\{(\mathbf{q}_k \in \mu_i^j, \mathbf{p}_k^{i-1} \in \mathcal{M}_{i-1})\}$, $1 \leq k \leq |\mu_i^j|$, which minimize the metric

$$E_{motion} = \frac{1}{|\mu_i^j|} \sum_{k=0}^{|\mu_i^j|} \|\mathbf{q}_k - \mathbf{p}_k^{i-1}\|,$$

where $\|\cdot\|$ is the Euclidean distance in $\mathbb{R}^3$. Intuitively, E_{motion} captures the amount of motion between $\mathcal{M}_{i-1}$ and $\mathcal{M}_i$. Since the cow stays relatively still during milking, the robot moves slowly, and our camera has a high frame-rate, teats cannot 'jump' from one place to another one between consecutive frames. Hence, for a CMS μ_i^j to be valid, it has to yield a small value for E_{motion}. In practice, we allow only values $E_{motion} < 25$ mm.

3.2.3 TTS estimation

From each CMS μ_i^j given by correspondence finding, we build a potential new TTS $\mathcal{M}_i^j$ for the current frame i: For all points $\mathbf{q}_k \in \mu_i^j$ which have a correspondence to a TTS-quad vertex $\mathbf{p}_k^{i-1} \in \mathcal{M}_{i-1}$, we set the new value of $\mathbf{p}_k^i \in \mathcal{M}_i$ to $\mathbf{q}_k$. For all other vertices $\mathbf{p}_k^i \in \mathcal{M}_i^j$ which have no correspondences in μ_i^j, a situation which occurs when $|\mu_i^j| < 4$, we compute their values by translating their corresponding points $\mathbf{p}_k^{i-1} \in \mathcal{M}_{i-1}$ with the average translation vector

$$\mathbf{v} = \frac{1}{|\mu_i^j|} \sum_{k=0}^{|\mu_i^j|} \mathbf{q}_k - \mathbf{p}_k.$$

3.2.4 TTS optimization

The previous step delivers as many potential TTS models $\mathcal{M}_i^j$ as the number $\|S\|$ of CMS configurations. These are all possible TTS models which can be built by using one or several matches in M_i. We select the best such TTS as the optimal TTS with respect to three metrics which describe geometric constraints observed by watching videos of actual cows during milking, as described below. Let us stress here that we are not searching for an absolute minimum of these metrics, but for a 'best fit', *i.e.*, a TTS which optimizes these metrics over all possible TTSs.

Shape: During milking, the soft udder shape changes as the cow moves. Yet, the *relative* teat positions are quite stable. Thus, the *shape* of our quad $\mathcal{M}_i^j$ should be constrained. While this is partly done by the motion constraint E_{motion}, that allows teats to move only slightly, an accumulation of such small movements over hundreds of frames can yield very different quad shapes. We thus further constrain the quad shape by constraining its area. We could have used other shape metrics here, *e.g.* the quad's aspect ratio. However, the area constraint performs much better during the tracking-initialization stage (see next Sec. 3.2.5). We model the area constraint by the difference between the actual quad-area and the expected quad-area $A_{expected}$ as

$$E_{shape} = \frac{|A(\mathcal{M}_i^j) - A_{expected}|}{A_{expected}}.$$

Here, $A_{expected}$ is a fixed value, computed from actual udder measurements of the cows under analysis. Setting $A_{expected}$ has to be done only once, before the first time the cow is milked, and can be re-used for subsequent milking.

Flatness: We also observed that teat tips stay roughly in the same plane. We therefore want the same to hold for the vertices of the quad $\mathcal{M}_i^j$. We model this by checking how close each vertex $\mathbf{p}_k \in \mathcal{M}_i^j$ is to the plane formed by the other three vertices, *i.e.* by the metric

$$E_{flatness} = \frac{1}{4} \sum_{k=0}^{4} |\mathbf{n}_k \cdot \mathbf{v}_k|.$$

Here, $\mathbf{n}_k$ is the normal of the plane through all quad points except $\mathbf{p}_k$, and $\mathbf{v}_k$ is the normalized vector from any point $\mathbf{p}_{l \neq k}$ to $\mathbf{p}_k$. When our quad is flat, every $\mathbf{p}_k$ lies in the same plane as the other points $\mathbf{p}_{l \neq k}$, so $\mathbf{n}_k$ and $\mathbf{v}_k$ are orthogonal to each other, thus $E_{flatness} = 0$. Higher values of $E_{flatness} > 0$ tell that $\mathbf{p}_k$ do not all lie in the same plane. In particular, note that configurations that include an incorrectly detected point on the cow's tail yield a high $E_{flatness}$, thus are not favored by this metric.

Orientation: Finally, we note that teat tips are in a plane roughly parallel to the ground surface on which the cow stands. We encode this prior by measuring the orientation-deviation between the quad vertex-normals $\mathbf{n}_k$, computed as for the flatness criterion, and the vertical direction $\mathbf{u}$, by

$$E_{orient} = \frac{1}{4} \sum_{k=0}^{4} |1 - \mathbf{n}_k \cdot \mathbf{u}|.$$

In the ideal case, all normals $\mathbf{n}_k$ are parallel to $\mathbf{u}$, so $E_{orient} = 0$. Values $E_{orient} > 0$ indicate deviations from the desired orientation. Similar to the flatness metric, the orientation metric typically produces higher values for incorrectly oriented vertices and therefore also favors the correctly oriented configurations, even when the corresponding value for E_{orient} is not optimal in an absolute sense.

To jointly optimize for TTS shape, flatness, and orientation, we use the total geometric error

$$E_{geom} = w_{shape} \cdot E_{shape} + w_{flatness} \cdot E_{flatness} + w_{orient} \cdot E_{orient}$$

where the weights w sum up to 1. The first TTS $\mathcal{M}_i^j$, in the testing order given by CMS finding (Sec. 3.2.1), that scores $E_{total} < \varepsilon$, is considered a good-enough fit, and yields the new value for the TTS $\mathcal{M}_i$ for the current frame i. Here, we use $\varepsilon = \frac{1}{3}$, meaning that only one of the three error metrics can be at its acceptable maximum, while all other error metrics should be zero for us to accept this configuration.

3.2.5 Initialization

To start tracking, we must initialize our TTS $\mathcal{M}$. Also, re-initialization is needed when we cannot track $\mathcal{M}_{i-1}$ to the current frame i. This happens when (a) the current match-set M_i is empty, *e.g.* due to a bad camera angle, too large distance to the cow, complete occlusion of teats in frame i, or limitations of our teat-detection algorithm; (b) no correspondence between $\mathcal{M}_{i-1}$ and M_i exists which satisfies the motion constraint E_{motion} (Sec. 3.2.2), *e.g.* because of accidental robot jumps due to collisions with the cow; (c) no candidate TTS $\mathcal{M}_i^j$ having a sufficiently good geometry E_{geom} is found, *e.g.* due to the same reasons as for (a).

In all such cases, we must build $\mathcal{M}_i$ afresh, using only data from M_i. For this, we first find all CMS sets μ_i^j having *at least* three points, by the same method as for tracking (Sec. 3.2.2). We regard each μ_i^j as a potential TTS $\mathcal{M}_i^j$, and compute its E_{geom}. The TTS yielding a minimal E_{geom} value below our threshold ε becomes our new $\mathcal{M}_i$. If no such TTS is found, we set $\mathcal{M}_i = \varnothing$, *i.e.* mark that tracking is lost in the current frame, and try to re-initialize in the next frame.

Let us further detail the difference between tracking and initialization. During tracking, we optimize for the TTS that (a) fits the most matches found in the current frame, (b) has the best geometric quality, and (c) has a small motion with respect to the previous TTS. In contrast, at initialization we only optimize for geometric quality and number of matches. Indeed, we cannot optimize for motion, since the previous valid TTS may have occurred many frames ago or there was no such TTS (at the video stream start). To track, we only need a *single* valid match in each frame. For initialization, we need minimally *three* valid matches in a frame (to be able to evaluate the geometric constraints). As we shall see in Sec. 5, our tracking is robust enough to require re-initialization only very seldomly, and thus deliver a high overall quality of the proposed solution.

4 Results

Our tracking-and-detection system, implemented in unoptimized C#, achieves tracking at 4…8 fps on a 3.0 GHz Windows PC for an input video stream provided by the SR4000 API. For an image resolution of N pixels, both computational and memory complexities of detection are $O(N)$; for tracking, these are both $O(1)$, since the match-set sizes are not a function of the image size. This strongly suggests that an optimized implementation, *e.g.* in embedded C, can run at real-time rates on a low-cost ARM processor such as available on the milking robot, which supports our claims for practical industrial applicability and low cost.

Figure 7 shows the interaction between detection and tracking by showing the TTS results for 3 sequential frames selected from a longer video. The first frame (a) is an initialization frame. Here, five matches are found (red rectangles). Of these, the correct four corresponding to teats are selected by the initialization procedure (Sec. 3.2.5) to create the current TTS $\mathcal{M}_a$, as using any of the other two false-positives would create tilted quads which yield a high error E_{geom}. The obtained TTS is shown in Fig. 7d atop of a rendering of the point cloud zoomed in on the udder area. As can be seen, the TTS approximates the actual teat positions quite well. In the second frame (Fig. 7b), we find only three true-positive matches on the teats, and two false-positives. However, as seen in the corresponding cloud rendering (Fig. 7e), tracking correctly estimates the position of the fourth teat. In the final image, we only detect one true-positive and one false-positive (Fig. 7c). Here again, the tracking succeeds in creating the correct TTS (Fig. 7f).

5 Quality Analysis

Analyzing the full tracking process is crucial to validate the robustness and correctness of our proposed solution. The video data we use is unlabeled, *i.e.*, has no ground-truth for the correct teat positions. Labeling it would cost a huge effort (manually marking 3D teat positions in thousands of frames for several videos). Thus, we base our validation on (a) the visual inspection of the tracked teat-set $\mathcal{M}$ (Sec. 3.2), and (b) on a data-analysis tool for the tracking process. This analysis tool was crucial in helping us find an optimal set of parameters, metrics, and heuristics for our problem. The analysis tool is described below.

During tracking, we record all computed information: input and derived images, match locations, tracked teat positions, error metrics, and system state (tracking, initializing, or tracking lost). Our analysis tool next aims to show such data to (a) allow validation of the tracking quality; and (b) help finding reasons for sub-optimal tracking, potentially leading to algorithm and parameter improvements.

Our analysis tool consists of several linked views (Fig. 8). Its set-up follows the overview and details-on-demand design common for visual analytics tools (Shn96), showing both overall tracking performance, but also finer-level details that explain this performance. The analysis tool is connected in a feedback loop with detection-and-tracking (Secs. 3.1,3.2) so that the analyst can spot sub-optimal results in the overview, examine details to find their causes, adjust the responsible algorithm parameters, see the effects (*e.g.* improvements), and repeat the process until an optimal algorithm and parameter-set is found.

We next detail the views of our analysis tool. The *model state* view shows a timeline overview of the TTS model state (initializing, tracking, or tracking lost). States are shown by color-coded bars – blue=tracking, yellow=re-initialization, and red=tracking lost. This gives an easy-to-follow global overview of the entire tracking process, and allows quickly spotting frames whose state changes from neighbor frames, *e.g.* frames where tracking fails and which occur in a sequence of correctly tracked frames. After spotting such frames, we can use the views described next to find causes of the respective state-change.

The *tracking view* refines the overview information from the model state view by showing graphs of all model variables as functions of time. Correlating values of these signals with state values (or state changes) in the model view allows tracing back the cause of the respective states one step back, *i.e.*, to the components of the error metrics E_{geom} or E_{motion} (Sec. 3.2). For instance, in the model-state view in Fig. 8, we see a suspiciously large amount of red (tracking lost) frames. At first sight, this suggests that our tracking is not working optimally. Let us focus on the largest red block, marked A in Fig. 8. We see that this block correlates to a zero value for the $E_{flatness}$ metric (Eqn. 3.2.4) in the tracking view. This tells us that tracking is lost because this metric had a too large value, which in turn caused E_{geom} to exceed the allowed threshold ε. Showing other model variables in the same view allows back-tracing the cause of a large $E_{flatness}$ error to earlier data, such as the number and locations of found matches. Using this procedure, we found out that, for the time-range of block A, the cause was that there were no correct matches found in the image, due to the robot drifting out of the udder area. As we expect tracking to be lost in such cases, this does not flag a problem of our tracker, but of the robot's steering.

The *TTS view* shows the trajectories of the four tracked teats over the entire analyzed video, both as 2D camera-view projections (TTS view, top images) and also as 3D world space positions (TTS view, bottom images). Given the assumed smooth motion of both the tracked shape (cow) and camera (robot), such trajectories should be smooth curves. Also, these curves should have a relatively similar overall shape, given the geometric constraint that limits the relative motion of teats from each other (Sec. 3.2.4). Spotting large line-segment jumps in the TTS view allows us to find time-ranges when tracking performed incorrectly. Such a jump is marked B in the figure, and is visible for all four teats. Clicking on such a jump brings the data for the

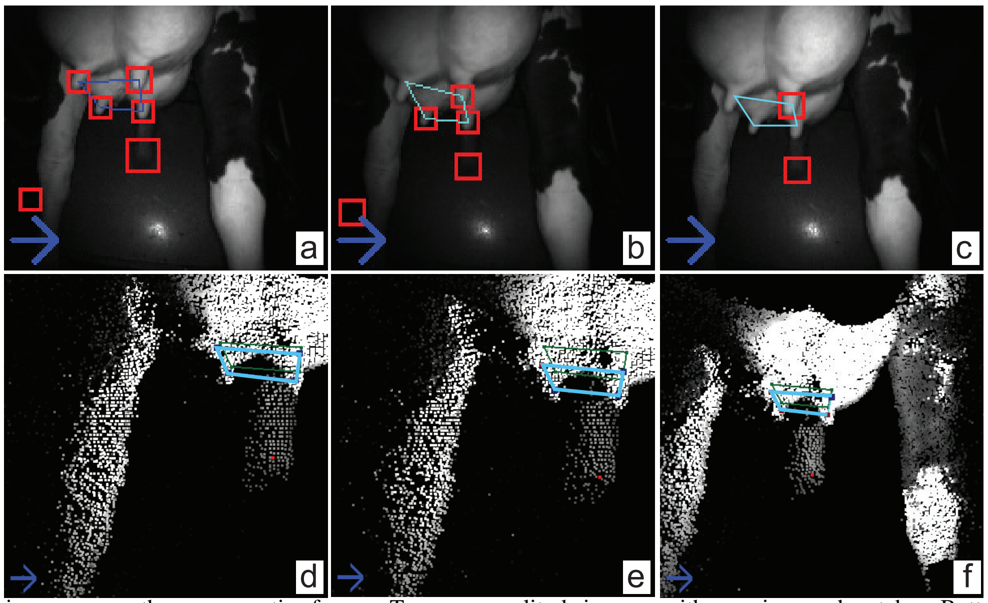


Figure 7: Tracking sequence, three consecutive frames. Top row: amplitude images, with superimposed matches. Bottom row: zoom-in on the point cloud around the tracked TTS. The bottom-left arrow icon indicates that the system is successfully tracking (see Sec. 4).

respective time moment(s) in focus in the other views. The current time is shown in the tracking view by the dot marker labeled *C*. We now see that this moment corresponds to the beginning of the first large red block in the tracking view. Hence, we know that the jump is caused by a tracking-lost event (which is expected and correct). If, however, the jump corresponded to a tracking state (blue in the model state view), this would have shown severe tracking problems, as the tracking would have created jumps (not in line with our knowledge of the studied phenomenon) *and* would have marked these as valid tracked frames.

The *frame data* view shows the amplitude, depth, and point cloud data acquired from the ToF camera for the frame selected in the other views, as well as numerical statistics on this frame (number of matches and values of the model metrics). These 'details on demand' allow refining the insight obtained from the overviews. All views are linked by interactive selection – clicking on a time-instant or position in the overviews shows details of the selected frame in the frame data view. For instance, the frame data in Fig. 7 corresponds to the moment *C* discussed above. As visible in the amplitude image, the two back teats are now connected to the suction cups of the milking robot. In such cases, tracking is expected to be lost (due to the robot being too close to the udder). Hence, we have explained that the tracking-lost event observed in the TTS and model-state views is expected and not due to a tracker problem.

The analysis tool allows browsing a video both frame by frame or playing it in real-time, so that correlations between tracking performance and algorithm variables can be easily seen. Using this tool, we have been able to refine our proposed detection-and-tracking algorithms, fine-tune their parameters, and also validate the end-to-end tracking performance of our system. Overall, we have tested over 15 real-life videos of several minutes each acquired in actual stables in a production-process environment, that cover a wide range of camera-to-subject distances, angles, and motion paths. Average tracking performance amounts to **over 90%** of the frames being successfully tracked. This clearly exceeds the documented performance of comparable systems (LMI12; Sco13; MES14; Wes09; Hun06).

6 Conclusions

We present an end-to-end system for the detection of cow teats for automatic milking devices (AMDs) in the milk industry. We present several techniques and algorithms that make this detection robust and fully automated when using a very low resolution time-of-flight camera, which renders classical computer vision algorithms not applicable. By combining depth and point cloud information analysis with observed model priors, we achieve a simple and robust implementation that can successfully track over 90% of the frames present in typical AMD videos, which exceeds the performance of all known competitive solutions in the area. In contrast to these solutions, our proposal is also fully automated, allows large relative camera-subject motions and orientation changes, and accounts for occlusions. We present a visual analytics tool that allows tracker refinement and result validation.

Several extension directions are possible. Different teat detectors can be designed to find teats more accurately under extreme zoom-out conditions, *e.g.* based on 3D template matching. Secondly, using a more complex model including both teats and udder shape should render our tracking performance even higher in contexts of high occlusion. Such refinements will lead to a more effective solution for the next generation of AMD robots for the dairy industry.

REFERENCES

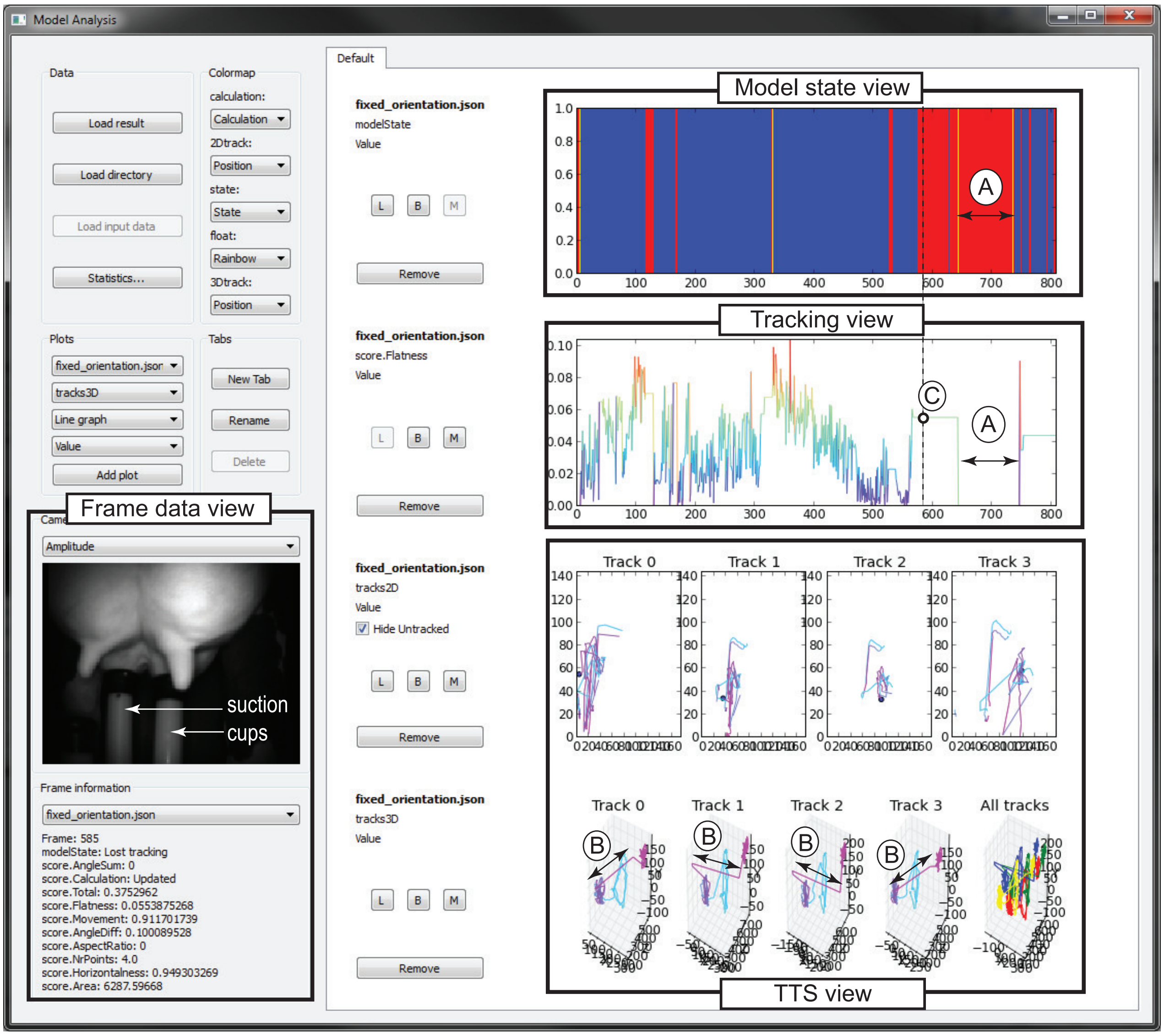
AGARWAL A., TRIGGS B.: Recovering 3D human pose from monocular images. *IEEE TPAMI 28*, 1 (2006), 44–58.

BAYA H., ESSA A., TUYTELAARS T., GOOL L. V.: Speeded up robust features. *CVIU 110*, 3 (2008), 346–359.

CHEN D., FARAG A., FALK R., DRYDEN G.: A variational framework for 3D colonic polyp visualization in virtual colonoscopy. In *Proc. IEEE ICIP* (2009), pp. 2617–2620.

DISTANTE C., DIRACO G., LEONE A.: Active range imaging dataset for indoor surveillance. *Ann. BMVA 21*, 3 (2010), 1–16.

DEY T., GOSWAMI S.: Provable surface reconstruction from noisy samples. In *Proc. SCG* (2004), pp. 428–438.



Colormap
calculation: Calculation
2Dtrack: Position
state: State
float: Rainbow
3Dtrack: Position

Plots: fixed_orientation.json, tracks3D, Line graph, Value

Camera: Amplitude

fixed_orientation.json
modelState
Value

fixed_orientation.json
score.Flatness
Value

fixed_orientation.json
tracks2D
Value
☑ Hide Untracked

fixed_orientation.json
tracks3D
Value

Frame information
fixed_orientation.json
Frame: 585
modelState: Lost tracking
score.AngleSum: 0
score.Calculation: Updated
score.Total: 0.3752962
score.Flatness: 0.0553875268
score.Movement: 0.911701739
score.AngleDiff: 0.100089528
score.AspectRatio: 0
score.NrPoints: 4.0
score.Horizontalness: 0.949303269
score.Area: 6287.59668

Figure 8: Visual analysis tool for our teat detection-and-tracking system. The analysis tool consists of three overview views (model state, tracking view, and TTS view) and one detail view (frame data). All views are bidirectionally linked by interaction (see Sec. 5).

DEY T., LI K., RAMOS E., WENGER R.: Isotopic reconstruction of surfaces with boundaries. *CGF 28*, 5 (2009), 1371–1382.

DORRINGTON A., PAYNE A., CREE M.: An evaluation of time-of-flight range cameras for close range metrology applications. *ISPRS J. Photogramm. 38*, 5 (2010), 201–206.

HOVINEN M., AISLA A., PYÖRÄLÄ S.: Visual detection of technical success and effectiveness of teat cleaning in two automatic milking systems. *J. Dairy Sci.*, 88 (2005), 3354–3362.

HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. *Proc. ACM SIGGRAPH 26*, 2 (1992), 71–78.

HUNT A.: Teat detection for an automatic milking system. In *MSc thesis, Univ. of Dublin, Ireland* (2006). doras.dcu.ie/17194/1/aidan_hunt_duffy_20120703135817.pdf.

KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proc. SGP* (2006), pp. 61–70.

KUSTRA J., JALBA A., TELEA A.: Robust segmentation of multiple intersecting manifolds from unoriented noisy point clouds. *CGF 33*, 1 (2014), 73–87.

LMI TECHNOLOGIES: Time of flight imaging enables automated milking, 2012. www.lmi3d.com.

LOWE D.: Distinctive image features from scale-invariant keypoints. *IJCV 60*, 2 (2004), 91–110.

LAZAROS N., SIRAKOULIS G., GASTERATOS A.: Review of stereo vision algorithms: From software to hardware. *Int. J. Optomechatronics 2* (2008), 435–462.

MESA IMAGING: SR4000 user manual, 2010. www.mesa-imaging.ch/prodview4k.php.

MESA IMAGING: Automatic milking systems, 2014. www.mesa-imaging.ch/applications/automatic-milking-systems.

OPENCV: OpenCV library, 2014. www.opencv.org.

SCOTT MILKTECH LTD.: World's first automatic milking system, 2013. scott.co.nz/scott-milktech.

SHNEIDERMAN B.: The eyes have it: A task by data type taxonomy for information visualizations. In *Proc. IEEE Symp. Vis. Lang.* (1996), pp. 336–343.

SMINCHISESCU C., TRIGGS B.: Covariance scaled sampling for monocular 3D body tracking. In *Proc. IEEE CVPR* (2001), pp. 447–454.

SUSSMAN M., WRIGHT G.: The correlation coefficient technique for pattern matching. In *Proc. ISMRM* (1999), p. 203.

WESTBERG M.: Time of flight based teat detection. In *Tech. Report LiTH-ISY-EX-09/4154-SE, Univ. of Linköping, Sweden* (2009). liu.diva-portal.org/smash/get/diva2:224321/FULLTEXT01.pdf.

WESTIN C., PELED S., GUBJARTSSON H., KIKINIS R., JOLESZ F.: Geometrical diffusion measures for MRI from tensor basis analysis. In *Proc. ISMRM* (1997), pp. 17–42.

YUILLE A., HALLINAN P., COHEN D.: Feature extraction from faces using deformable templates. *IJCV 8*, 2 (1992), 99–111.