FREIE UNIVERSITÄT BERLIN

Tracking honey bee dances from sparse optical flow fields

Tim Landgraf, Raúl Rojas

B-07-11
June 2007

FACHBEREICH MATHEMATIK UND INFORMATIK
SERIE B · INFORMATIK

Tracking honey bee dances from sparse optical flow fields

Tim Landgraf, Raúl Rojas
Institut für Informatik
Freie Universität Berlin

June 2007

Abstract

Tracking honey bee dances precisely is a hard task since the bees are highly flexible and the background highly distractive and cluttered. We propose methods that cluster optical flow features using a fast hough transform. A bounding box is shifted according to the movement of the underlying optical flow features. Errors made are corrected in a second step by evaluation of coherence of motion for future frames.

1 Introduction

Returning forager honey bees recruit others to remote field locations by performing the "waggle dance" [2,3]. It is a form of communication containing information about the location and value of a food source and the amount of energy a bee needs to spend to reach it [5]. After 60 years of research on the bee dance the questions a) what are the essential parts of the dance that convey information and b) how do the follower bees perceive these parts in the darkness of the hive remain unanswered [4]. Analysis of trajectories of the dancer and follower bees may offer insight into these questions. We recorded hundreds of high-speed video recordings of bee dances and developed a software to track the recorded bees. We extract both the transform parameters and the quality (i.e. the coherence of motion) of the motion cluster consisting of optical flow features and are therewith able to shift a bounding box according to the movement of the bee and correct for possible errors made in the optical flow computation. Interesting results regarding the paradigmatical "figure 8 shape" of the dance are found.

2 Preliminaries

2.1 The Bee Dance

Knowledge about the bee dance goes back to Aristotle [0, 1]. A patient observer is able to find bees returning from a foraging flight and right after arriving in a zone near to the entrance performing vigorous movements of the abdomen to the left and right while moving forward in a rather straight line on the comb surface. After a variable number of oscillations the bee returns to the beginning point of this 'wagging run', alternating between a left and right return path (Figure 2-1).

This dance shape is often idealized in literature as the figure 8. These dance shapes are repeatedly performed a variable number of times and may only be interrupted for food presentation to 'attending' bees (*trophallaxis*). After dancing and unloading the crop the dancer bee will mostly exit the hive to forage again.

Figure 2-1 : The waggle dance of the honey bee. Figure design: J. Tautz and M. Kleinhenz, Beegroup Würzburg.

Nobel prize winner Karl von Frisch was the first who found that this behaviour known for so many years carries specific information (von Frisch, see [25]). He discovered that the bees are communicating (therefore he was calling it "*dance language*") the locations of important food sources (later it was found that also places to get water from or even new hive sites are subject of the dance [19]). The position of such a location relative to the hive location is

communicated as well as the importance or desirability. Haldane [5] says: “[...] its simplest translation into human words is perhaps ‘There is a food smelling of A, requiring an effort B to reach it, in direction C, of economic value D.’”. We observe dances with wagging runs oriented straight up on the vertical surface of the comb when the forager returned from a location that lies directly towards the sun. The entire dance shape is rotated 90 degrees clockwise if the food location lies 90 degrees to the right of the sun’s azimuth. The farther away the location, the longer the wagging run. If detours are taken, the direct vector to the target is communicated, the length of the waggle run is accounting for the longer distance. As humans, we can say (statistically assured) what location a specific dance shape implies. But how do bees perceive the dance in the complete darkness of the hive? There is no answer yet. One intriguing way of finding out which parts of the bee dance carry the information would be to use a robotic bee (Haldane, [5]).

But there are also questions that may be answered just by looking. High-speed cameras deliver sharp images of the dancer and its followers. To investigate the role of antennal contacts and position of the followers relative to the dancer’s body we already profit from the tracking program (current work with Rodrigo. De Marco).

But still there are doubts and critics about the bee dance language (Wenner, [14]). To ultimately bring light to this long survey we need the means to have a closer and sharper look at the dance. The tracking is taking up part of this job.

2.2 Technical equipment

For the video recordings we used two cameras. In 2005 we used a high-speed camera (Redlake Motionscope PCI 500) that was able to record 2048 frames into an internal ring buffer. At a frame rate of 125 fps we obtained only 16 seconds of continuous video data. Since bee dances may even last minutes we decided to buy a new camera that is not restricted to the size of a proprietary internal memory. In summer of 2006 we utilized a firewire camera (Basler A602f) that we equipped with a ring like array of 8 red light LED clusters (10 LEDs each cluster). A program was written for recording with this camera. Using a big RAM buffer in the memory of the notebook, we were able to obtain high frame rates (around 90 fps) for recording for a considerable long time, although the write speed of the hard disk was limited to approx. 13 MB/s (that would allow only 44 fps). Using this system we recorded ca. 800 dances. All implementations are based on Intel’s OpenCV library (see [26]).

2.3 Related work

There is a manageable number of works published about tracking honey bees. Recent work [11] proposes a combination of exact geometrical modelling and a behavioural model, enforced by a particle filter [12]. Others [10] propose an ansatz using eigenimages (‘eigenbees’), again combined with a particle filter.

Both proposals need a previously defined description of the appearance of the target (templates, eigenimages) and are therefore not conveniently suitable for changing ‘phenotypes’, as e.g. resulting from different videos (different scales and lighting, color or black and white images). Furthermore there was no information available about the user interface of these algorithms and the usability for computer laymen.

There are several commercial tracking programs such as from qubits systems (see [17]) or the probably most popular ‘Ethovision’ (noldus, [18]) that are able to track arbitrary objects – but they track the position only in the x-y-plane and need highly contrasted objects on clean backgrounds.

Tracking honey bees is highly specific, the scene is highly distractive, and the target is highly flexible. Therefore specialized and customized algorithms are needed.

3 Preliminary Work

3.1 High-speed video recordings

Before being able to begin the design process of algorithms that accomplish the above given tasks, high speed videos of bee dances had to be recorded. Therefore a camera (Redlake Motionscope PCI 500) was borrowed. Like mentioned above, the length of the videos obtained was not suitable for tracking an entire bee dance. Furthermore the image quality was poor. We used a normal white light for lighting the hive – which produced reflections on the wings and the glass window of the hive (see Figure 3-1). Though, the resulting videos were the first testing basis for the development of tracking algorithms.

Figure 3-1 : Sample frame. Highly distractive reflections on wings, motion blur of a wagging bee, occlusions. Note also the flexibility of the two bees near the center.

Then, a few months later, we were able to buy a new camera, a Basler A602f, that records at VGA solution up to 120fps. The camera connects through the IEEE 1394 interface (firewire), so we were not limited except by the hard disk memory available.

For the new camera we improved the lighting conditions. An array of red light LEDs that was arranged ring-like around the camera was built (see Figure 3-2).

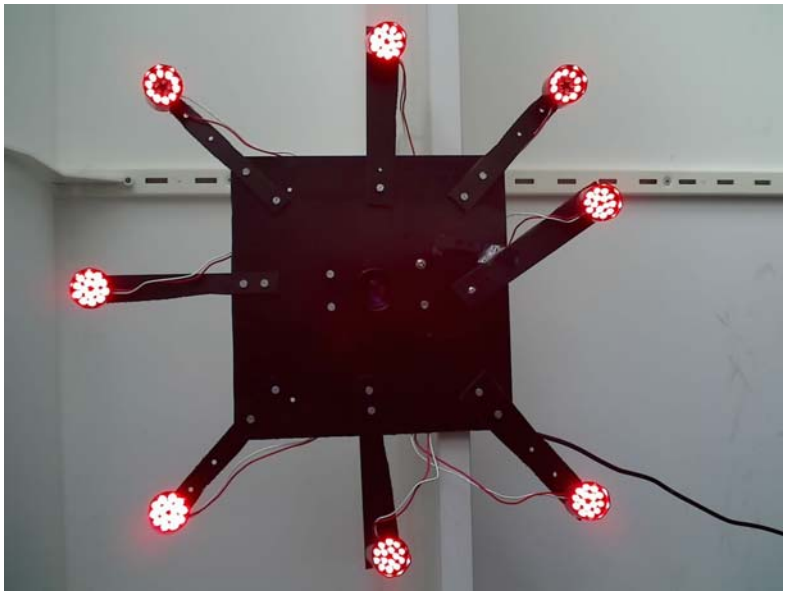

Figure 3-2 : Ring-like LED array. High speed camera in the center.

Since honey bees do not have receptors for red light, these LEDs enabled us to keep the disturbance of the bees at a minimal level. A recording program was written. Using an appropriately big ring buffer we were able to record at 90 to 100 fps many minutes before the buffer overflew. After buying a newer and faster laptop we are now able to record as long as it is desirable. Image quality has increased remarkably, see Figure 3-3.

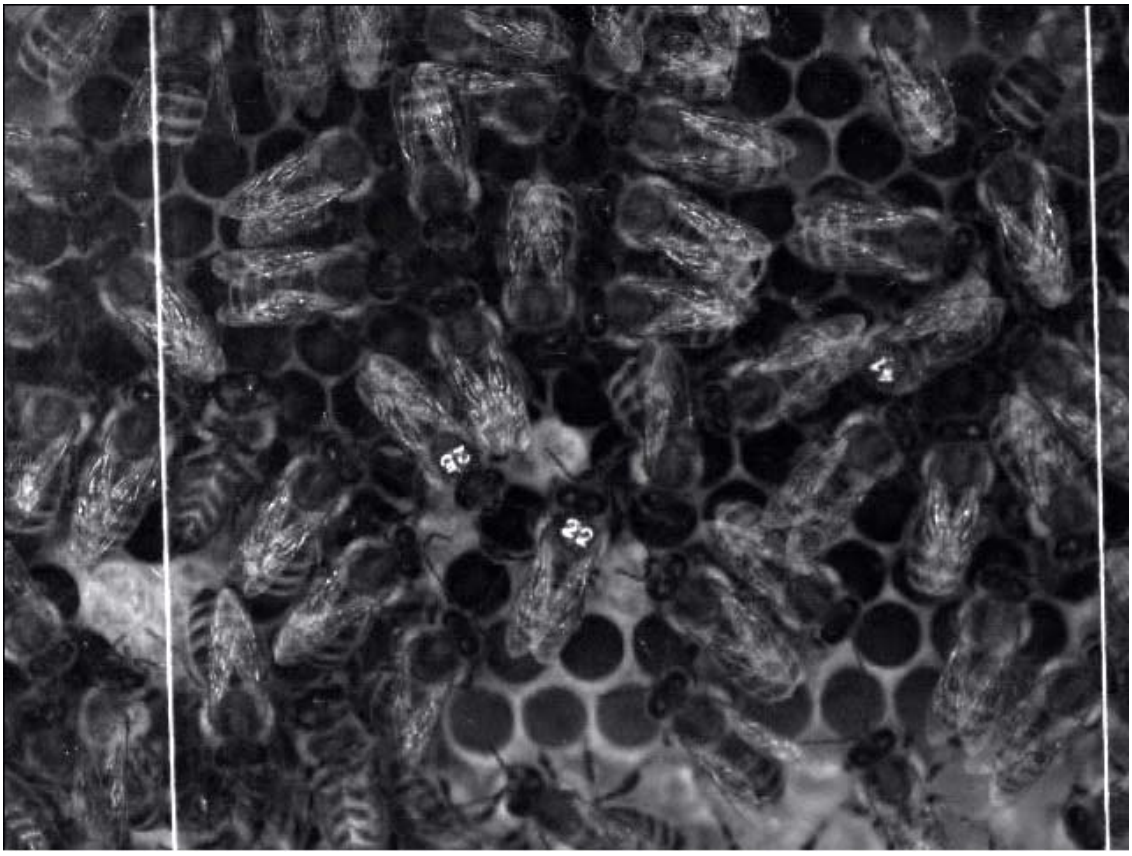

Figure 3-3 : Example frame taken from a sequence recorded using the new camera and lighting device. Note fewer reflections on the wings. Vertical stripes are hanging threads for camera roll normalization.

3.2 Template based matching

A straightforward idea for finding subsequent positions of a before known model was to use correlation techniques. Since we work on high speed recordings the bee's new location can not be anywhere in the image. Given an initial position, $P = (x, y, \alpha)$, we expect the movement vector $dP = (dx, dy, d\alpha)$ to be small. Depending on the zoom factor of the camera, the maximal movement speed of honey bees and the recording frame rate this will be in the range of only some pixels and degrees. If one would randomly distribute hypotheses around the initial bee position and then compare the underlying image with a previously defined template, one should easily be able to find the new position.

Accordingly an algorithm was implemented. A bounding box was used to designate the position of the dancer bee. The template was extracted from the video's first frame (initialization) and normalized to a zero degree orientation. Then randomly distorted copies (normal distributed) of the initial bounding box were scattered around the previously set position and evaluated by computing a correlation score with the template. The best scored model was selected to represent the next position.

Unfortunately this did not work satisfyingly. There are some task-inherent problems. First of all, honey bees are extremely flexible and versatile. One can observe rolling movements around the bee's longitudinal axis, showing parts of the sides. Then the abdomen of the animal can be retracted and extended again. The three body parts (head, thorax, abdomen) are not fixed; they can be moved and bent so that the appearance of the bee can differ extremely compared to the template image taken only a few frames before. Then the lighting produced many reflections on the wings which degraded the correlation score even more.

Figure 3-4 shows the same bee in different frames of one short video sequence. It that particular case the bee is marked. Tracking unmarked bees showed even less accurate tracking.

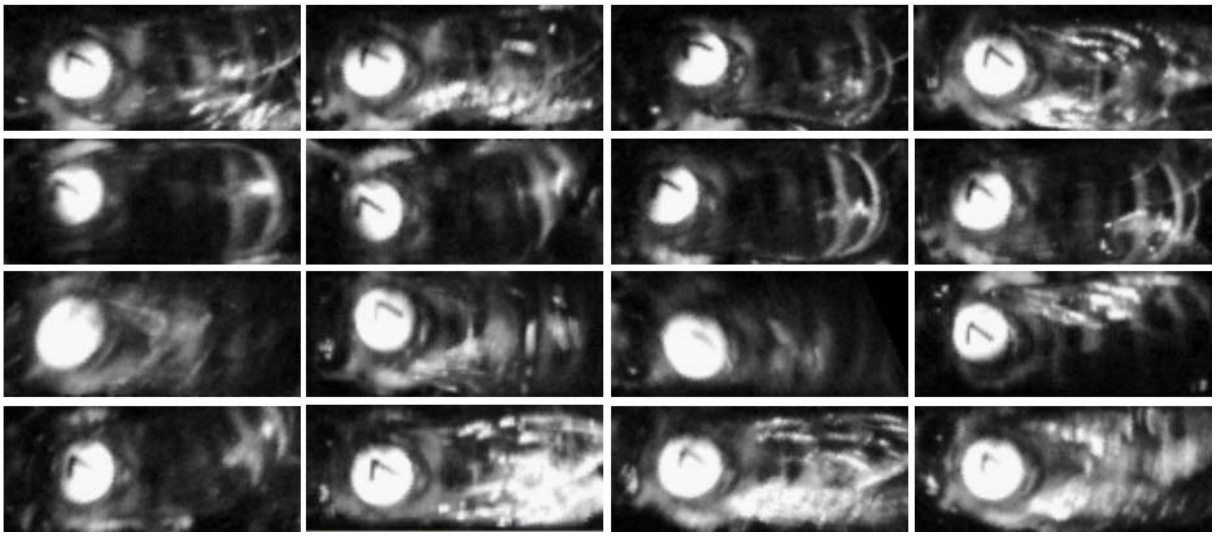

Figure 3-4 : The very same bee taken every 10th frame of a 5 second video clip. Note reflections on the wings (last row), bending of the abdomen (1st image of last row) and different lighting / shadows. There are even more extreme changes in the bee image in the set of videos we have.

So correlation based matching, at least the way it was tried here, did not work out. Modelling shape with ellipses (as in [11]) would probably result in better qualitative performance. Anyhow, we decided to leave the field of correlation simply because the speed of computing 100 or 1000 correlations, even using down sampled images was way too slow to give

biologists a handy tool for tracking the bees[1]. Furthermore would it have been necessary to define a template image for every video file. It would be desirable to have an algorithm that works independently of appearance size. [2]

3.3 Edge based model fitting

Another strategy may be to model objects using more detailed shape representation and fit this model onto the outlines of the object to be tracked. Tracking cars can be done by matching a 2D projection of a 3D car model onto the image edges. In hand tracking for example, active contours (see [16]) are matched against the edges of the hand.
To account for the flexibility of the bee's abdomen a deformable template with 9 parameters was formulated (Figure 3-5).

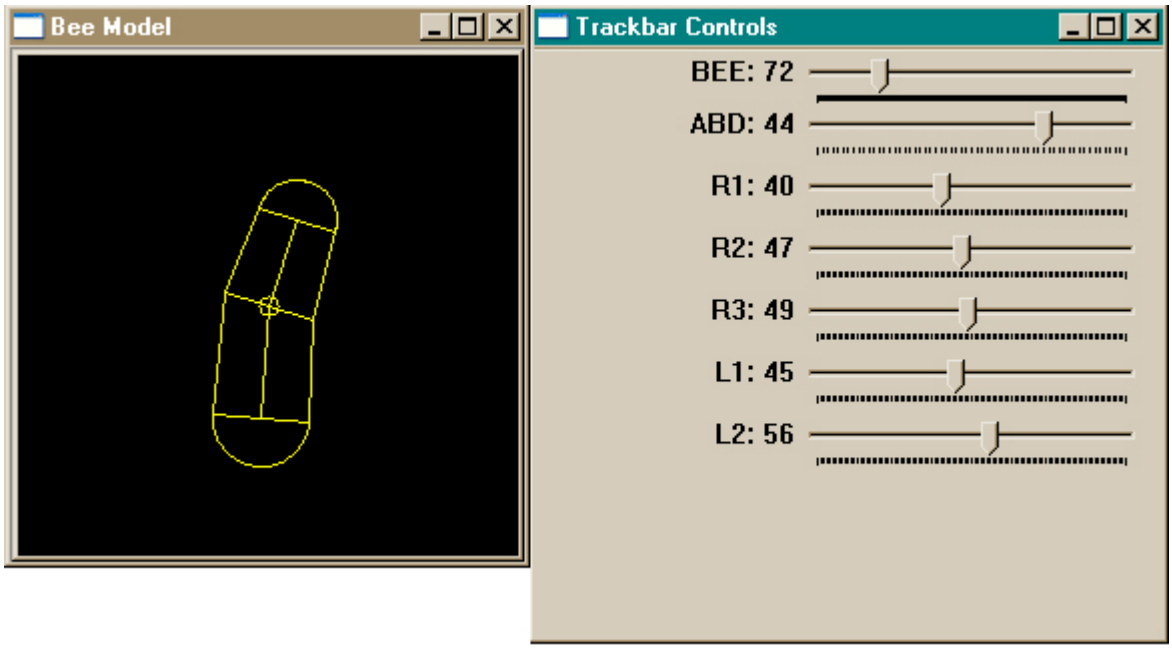

Figure 3-5 : Deformable template with 9 parameters. The right window shows trackbars to alter all parameters except x and y coordinates. Namely these are (top to bottom) : main orientation angle, angle of abdomen, width of the head, width of the body, width of the abdomen, length of first half, length of second half.

Again a high number of hypotheses may be generated randomly (this time 9 parameters are to be perturbed) and then scored against the edge image of the current frame. Therefor

[1] It would be possible to use the GPU (graphics processing unit) to speed up correlation to a acceptable level. This has not yet been tried.
[2] Later when the new camera was available and better videos were recorded it turned out to be still a good idea to use correlation for finding at least the head and thorax of the bee. Nevertheless we will continue the report. The concept of correlation will be used again in the following sections.

orthogonal lines with fixed distance to each other are created along the contour of the model (see Figure 3-6).

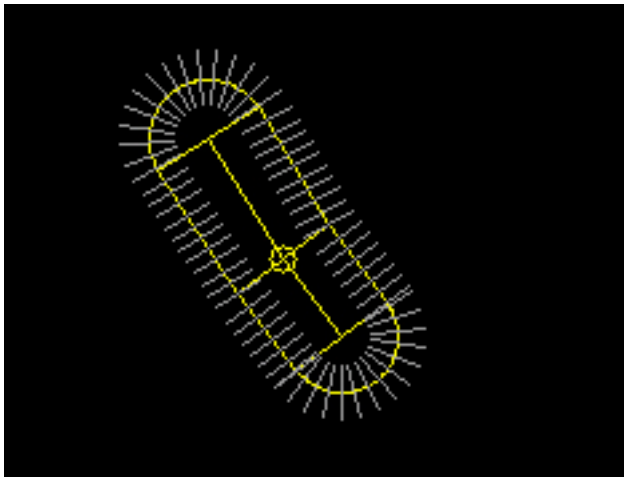

Figure 3-6 : Deformable template with contour orthogonals.

'Walking' these lines pixel wise and summing the closest occurrence of edge pixels (weighted with weight `w = 1/s(x)`, `s`: sigmoid function, `x`: distance to contour) gives an overall score of the fitting of the model to the contours in the image. Again this was implemented. Testing this algorithm with artificially generated bees resulted in success. The speed of computing the scores of a few hundreds of hypotheses was pleasurably high compared to the former template based methods.

But unfortunately again, the images of the high speed videos did not make the algorithm work. The textured images resulted in highly distractive edge images, and the model was fitted onto edges that very seldom were associated to the real bee's outline. Figure 3-7 shows a collection of edge images coming from the very same bee. There are some that clearly show the outline of the bee, some that show some missing parts of it and some that really show nothing bee-like. There are mainly two reasons why the edges vanish. Firstly, the motion blur coming from the waggle run and secondly, direct body contacts of bees. Since the waggle run has the main focus edges alone could not be the answer. Furthermore the threshold values for the edge detector were manually chosen. Due to changes in image quality and a missing heuristic for adjusting them to the very situation the edges may be considered mediocre.

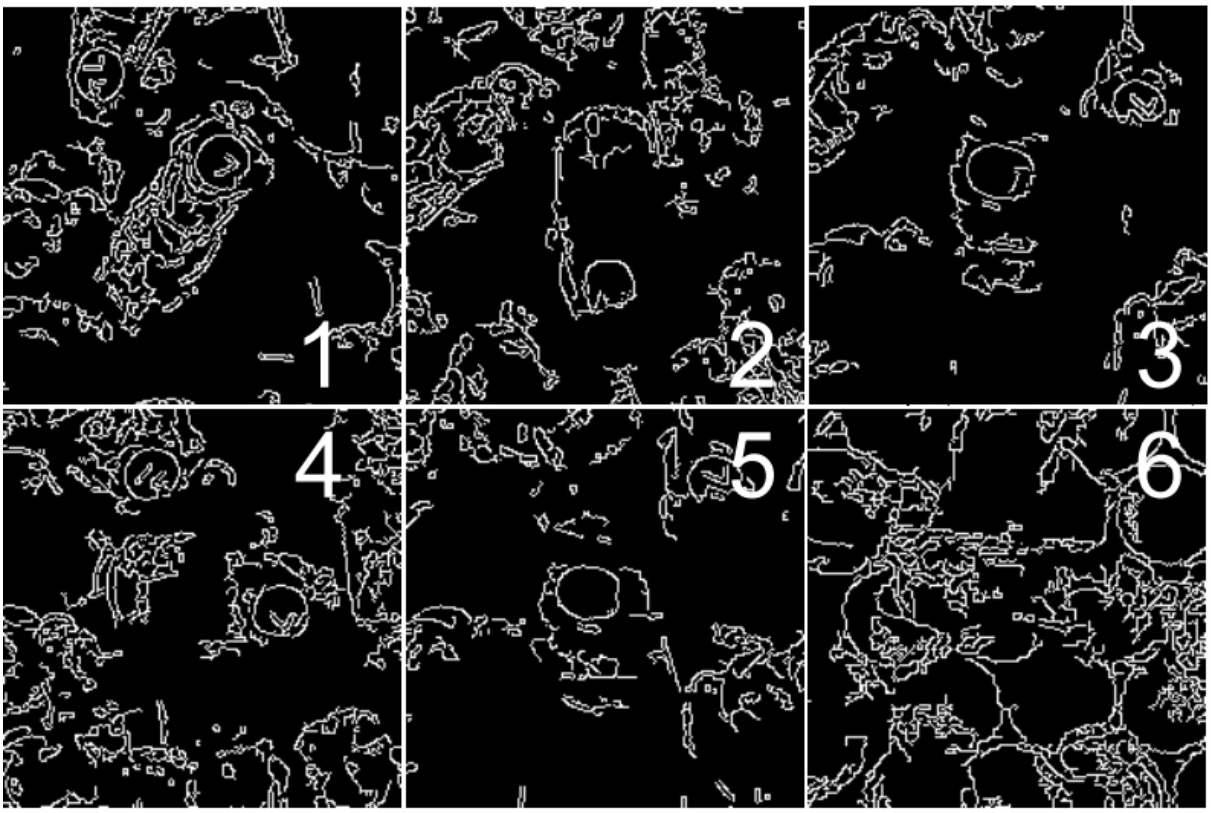

Figure 3-7 : Edge images obtained using the Canny algorithm (using the same threshold values). 1, 4 and 6: normal situation, 2: moving bee with missing edges due to shadows, 3 and 5: waggle dancing bee

3.4 Enhanced model fitting

Adding different terms to the score function of the models might improve the algorithm. Until now the score s of a model m was computed as $s(m) = s_{edge}(m)$, where s_{edge} computes the abovementioned edge score. The goal now was to find different markers that help to *pull* the model onto the bee by using an extended score function $s(m) = a_1 * s_{edge}(m) + a_2 * s_2(m) + a_3 * s_3(m) + \ldots + a_k * s_k(m)$, with coefficients a_i and score functions s_i $(i = 2, \ldots, k)$ that evaluate the current frame regarding different aspects.

First of all we tried to profit from one observation. Even if the contours of the bee were not found by the edge detector, taking the absolute difference of two consecutive images (diff) revealed high values in the vicinity of the contour at least if there was movement at all (Figure 3-8).

Figure 3-8 : Taking the absolute difference of two consecutive frames yields high values in areas where movement occurred – if the moved object is textured. Above 'diff image' shows a wagging bee.

So the fit onto this diff image was scored by walking along the abovementioned orthogonals but now incrementing the score if the corresponding value in the diff image was higher than a certain threshold.

Then because of the texturedness of the bee's image there are also high values inside the bee's contours in the diff image. Maximizing the amount of pixel values greater than a threshold that lie inside the model will attract the model onto the bee, again, at least if it is moving.

Using the information coming from the diff images improved the tracker noteworthy. But still there were failing tracks when the dancer bee was close to others, which generally is often the case. Furthermore misleading influences occur stronger when the bee does not move.

So there is the need of something different to keep track.

The observation that the image of the head and thorax region does not change as much as for example the image of the abdomen does (because of wing reflections) led us back to correlation. Using an image template for the head and thorax (Figure 3-9) will at least hold the model over the thorax region of the bee; at least as long as the ai in the score function are well chosen. The alignment then may be found using the edges and diff information.

Manually, values for the coefficients we found that produced results that were better than the results obtained with the template matching or using the edges alone for fitting.

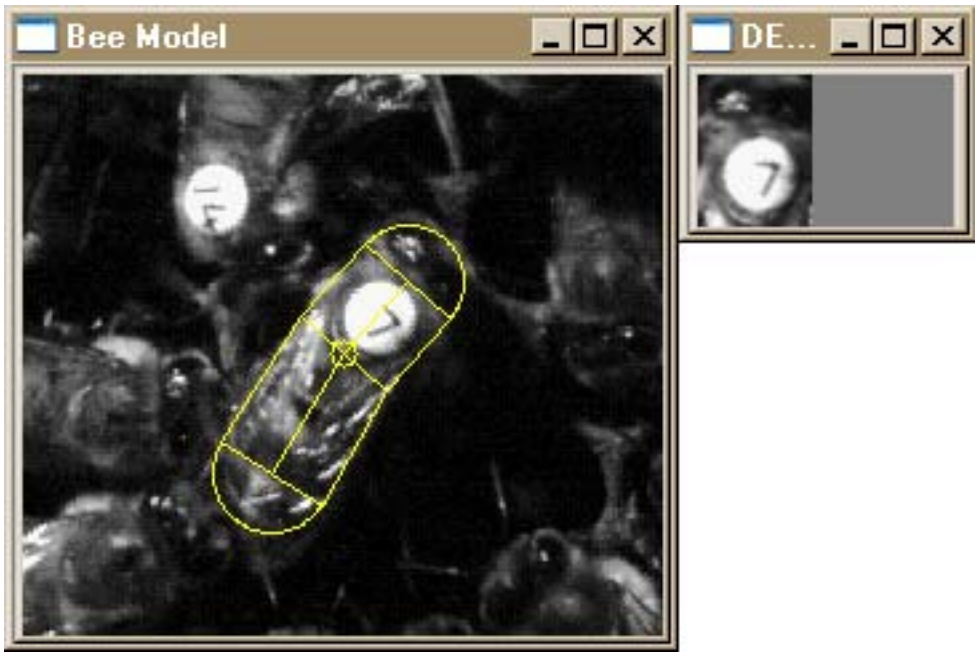

Figure 3-9 : Deformable template with extracted correlation template from head and thorax region. Tracking using only this template is more robust than tracking using a full template but exhibits alignment failures in the abdomen region because of missing scoring influences.

As long as the tracked bee did not move close to other bees the tracking was precise and working (see Figure 3-10). But there were other problems that relate to the score function. Since the coefficients were fixed only one weighting of the parts of the score function was used. But since the bee can also stand still one would have liked to have an increased coefficient for the edges and a lower one for the diffs. It is imaginable to formulate the score function as $s(m, t) = a_1(t)*s_{edge}(m) + a_2(t)*s_2(m) + \ldots + a_k(t)*s_k(m)$, such that the a_i are functions depending on the frame at time t, therefor evaluating the image at time t (and time $t-1$ for the diffs) and adapting the score to the current situation. This has not been tried yet.

The usage of template correlation improved but also slowed down the tracking.

The idea to look for the amount of movement in the expected area of the bee led us to a different tracking paradigm.

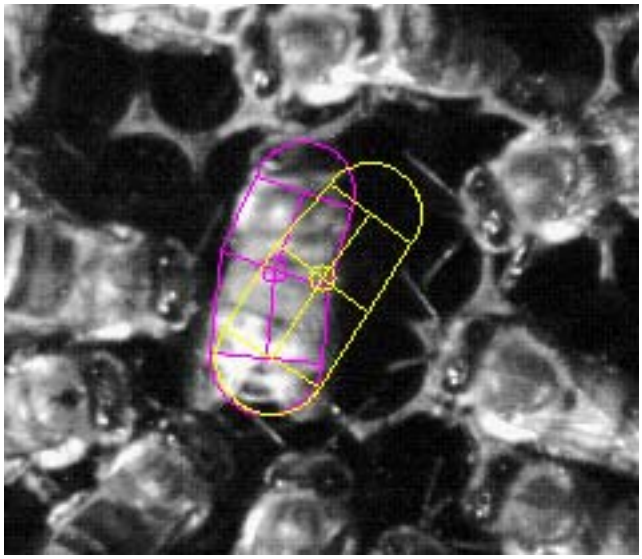

Figure 3-10 : Example of a working model fit. Given an erroneous initial position (yellow) the correct position is found (purple). Note the clear contrast-rich outline of the bee. No body contacts are established with other bees.

4 Optical flow based approach

Since the appearance of the honey bee is highly textured while the outline is hardly the same each frame it seemed compelling to eliminate the previous approach and switch to a different paradigm in computer vision: S*tructure from Motion*.

Computing the optical flow field of two consecutive images of a video yields the approximation of the motion field. We assume that due to the parallelism of the camera plane and the hive plane the projection of the 3D scenario is warped only a little. Under these assumptions we are able to find the bee by only aggregating regions that show similar movements and lie in relative vicinity of each other. This idea is the main cornerstone of the following final approach.

4.1 Sparse optical flow fields

While the computation of optical flow generally means to compute the flow vector of every pixel (or pixel patch) of the image and thus obtain an exhaustive global flow field, it is possible to do this computation only for an exclusive number of features. This results in a sparse optical flow field. The selection of pixels as features to be tracked can be done in a straightforward way ([8]).

Using a pyramidal implementation of the Lucas-Kanade tracker ([9], [7]) we obtained very promising point tracks of these features (Figure 4-1).

Figure 4-1 : Sparse flow field of a 5 frames sequence. A waggle dancing bee in the center.

After a number of frames these features lose quality, expressed in the similarity of the pixel patch (using a fixed window size) compared to a reference patch (in the first frame). These features are sorted out. Typically one observes the loss of a rather big portion of the features after 15-25 frames. Although there were features that lost track and moved off the bee the general impression was very positive. Watching the movements of these point tracks without displaying the originating video it is easy for a human observer to recognize the shape and movements of the bees. The optical flow reduces the amount of information to the essential parts. Only some pixel patches are trackable. All others are neglected. A second level of information is created on which the signal to noise ratio is increased. Therefore the tracking based on sparse optical flow fields is highly robust. From now on the tracking will be based on this second layer.

4.2 Overlapping optical flow

The aforementioned loss of quality and precision of the optical flow features or even the complete loss of track makes refreshing of the features necessary.

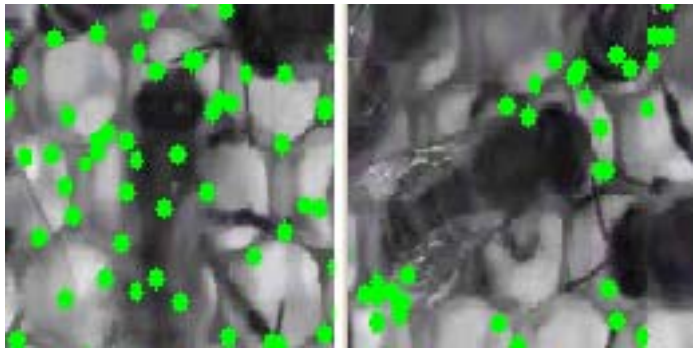

Figure 4-2 : Loss of optical flow features. Left: The first frame with features all over the bee. Right: After 50 frames no feature has survived. Note the point clouds at the tip of the abdomen and near the head. The features cumulate there or leave the bee entirely.

Assuming that the quality of an arbitrary optical flow feature decreases with time independently of the nature of the feature. Obviously a feature that has been found on the corner of a reflection on the wing will be highly probable to lose track after a small number of frames. But due to the flexibility and agility of the bees even features that are located on a stable corner (e.g. on the thorax) will lose track eventually. Thus we need a re-initialization of optical flow features. It is imaginable to erase single features which's qualities fall below a certain threshold and to find the same features again (or others in vicinity). Another way of accounting for the decay of the quality is to re-initialize the whole set of features after a number of frames being tracked. The problem would then be the loss of movement information between frames `i` and `i+1` where `i+1` is the index of re-initialization[3] (see Figure 4-3). To retain this information one would need to either re-initialize only one portion of the feature set while the rest is still being tracked to the following frame.

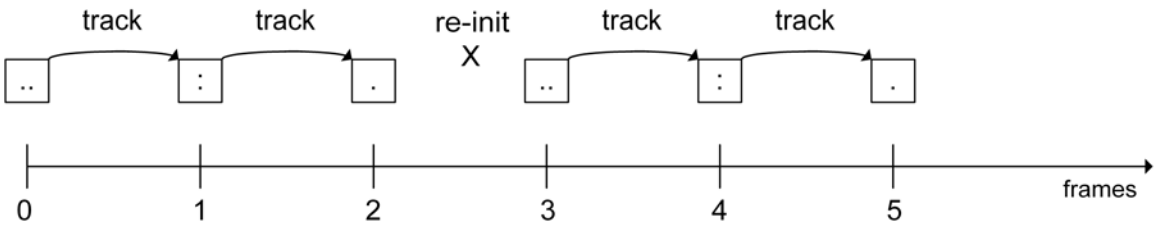

Figure 4-3 : Optical flow tracks. One box denotes a set of optical flow features. Suppose a set of points is tracked from frame 0 to frame 2. New points are initialized for frame 3. Movement information is lost. If re-initialization would take place in frame 2, one would need to store the points separately to not lose the results from tracking the points from frame 1.

Equivalently it is imaginable to track more than one set with shifted re-initialization times.

[3] Whereas this loss would not be critical. One could track the features to frame i+1, move the bounding box accordingly and then re-initialize the features on the same frame. The problem arises when we need to have a look at several future frames to move the bounding box (e.g. for scoring, see next sections).

Because these sets overlap this idea is called '*overlapping optical flow*'.
Suppose we have s_n sets of optical flow features. Each one of the sets is constructed to contain a number of features for the whole image (or a predefined region of interest, ROI). Let `i` be the current frame index of a video sequence. Then let the (re-)initialization be performed for set $s = i \bmod s_n$ using cvGoodFeaturesToTrack[4] on the i-th image (see Figure 4-4). All features contained in all other sets are tracked using the pyramidal Lucas-Kanade tracker. For $s_n = 3$, each set is re-initialized after 3 video frames, thus we obtain point tracks of two sets (the 3rd set is re-initialized).
The information obtained from the tracking of the feature points contained in these sets is highly redundant and could be subject to optimization (e.g. erase double points).
But since the number of feature points is high it seems to be possible to extract shape information from these point tracks as well.
C++ classes were implemented for computing, holding, exporting, displaying, searching and iterating these overlapping optical flow feature tracks.

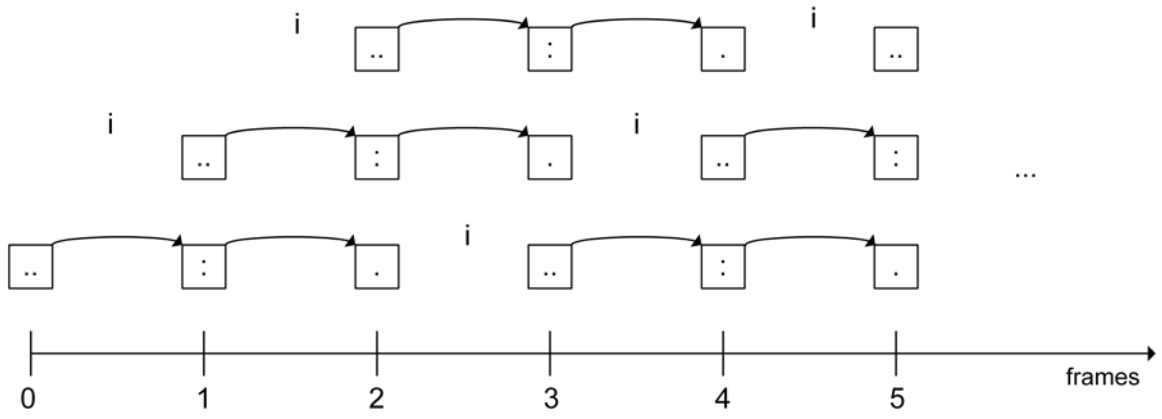

Figure 4-4 : Overlapping optical flow. Each row stands for a set of optical flow features. Each set is (re-) initialized every 3rd frame. The 'i' denotes an initialization step for the following frame.

The question now is: how can one extract shape and movement from these point tracks? Since the user should give us a hint where the respective bee is located, we have knowledge about the points that are initially on the bee. But after a number of frames these points are erased and others are added to the set of points. How can one cluster a subset of points so that they are likely to represent shape (or at least orientation) of the bee?

[4] The OpenCV implementation of [8].

It is obvious that clustering using only the motion vector (the direction of movement of the feature) is not enough. Since the bees turn on the comb surface we have to account also for rotations. Suppose the bee rotated around its center. There will be points located at the bee's head moving in one direction while others on the abdomen will move in the opposite direction. So computation can not focus only on single points. For obtaining knowledge about an (affine) transformation we would need at least three points.

4.3 Clustering by similar affine transformation

The idea is to cluster points that exhibit the same movements and then to compute the bounds, position and orientation of this point cloud to obtain the bee's position.

Selecting three points is possible using an initial mouse click on the bee. The points to be selected have to be near the input point. To increase the confidence that these three points do really belong to the bee some heuristics are imaginable. One can for example only take points that form a triangle (preferably close to equiangular) that undergoes only small shear (e.g. approximately by comparing the area). Obviously the three points must have a smaller distance to each other than a bee length (depending on the size of the bee image).

Let $p^1_{\{t,\ t+1\}}$, $p^2_{\{t,\ t+1\}}$, $p^3_{\{t,\ t+1\}}$ three points at times t and $t+1$. Let A be the transformation of these points (can directly be computed using least squares). Now iterate through all available points (that means all points that are not initialized at time $t+1$) and check if their movement $p^i_t \rightarrow p^i_{t+1}$ is conform to A (e.g. $|A \cdot p^i_t - p^i_{t+1}| < e$, where e is an error – or a function of a measure of the overall movement $e(\|p^i_{t+1} - p^i_t\|_2)$). If the above expression is fulfilled this point is classified to belong to the transformation (therefore to the bee) and other points can be checked iteratively. Using the added points it is possible to refine A.

The above circumscribed algorithm has been implemented and tested. Good results were found if there was sufficient movement. It turned out to be difficult to find a good error range e to exclude similarly little moving points outside the bee in situations where there is generally low movement. Also there was missing a shape model for the bee. Computing the bounds of the classified point clouds often failed to represent the bee (see Figure 4-5).

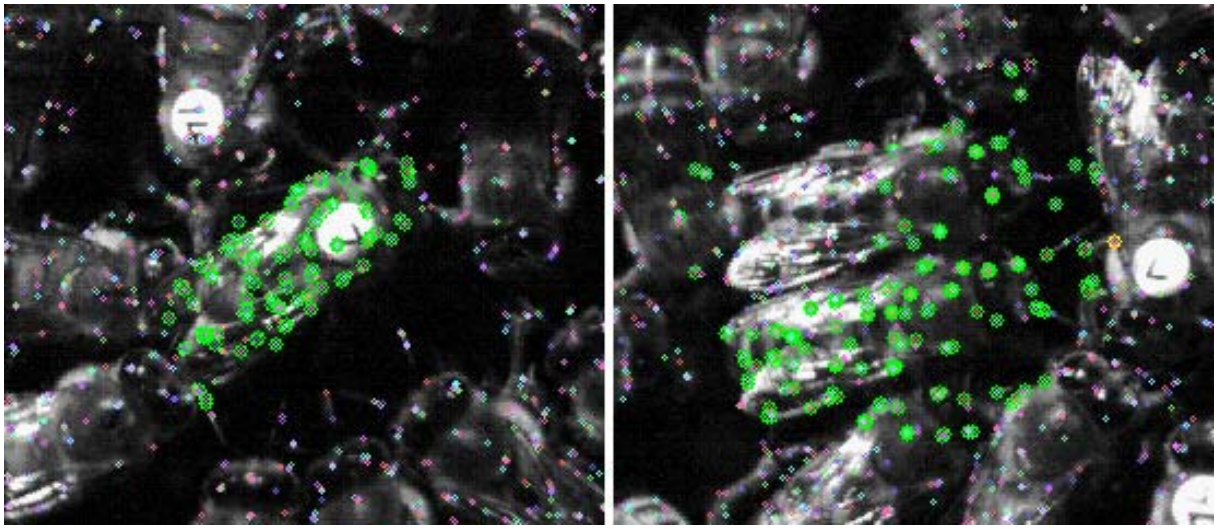

Figure 4-5 : Result of clustering by similar affine transformation. Small dots are optical flow features, big dots (green) are clustered features given 3 preselected points on the bee. Left: Almost all points on the dancer bee are selected to move similar. Right: Due to little movement also points on other bees are selected.

4.4 Clustering using a bounding box

One straightforward method to model the shape of the bee is to simply use a rectangle bounding box that is rotatable. The most important shape features (length, width, orientation) are inherent and computations such as the selection of points lying inside or outside a bounding box is possible using fast integer arithmetics (unlike using ellipses). Corresponding C++ objects and functions for the work with bounding boxes (BBs) were written.
Using BBs it is comfortable for the user to select an area of the image that largely is occupied by the bee. Now all optical flow features can be selected easily (improvements possible, see section 6.5). This selected subset of points (actually point tracks) can be used to compute the transform of the bee. Since we now have a rather safe set of points (we are at least for the very first frame confident that almost all points belong to the bee) we can try to access the transform directly. Assuming only small amounts of non-rigid point movements it is possible in a straight-forward way to directly compute the rigid transformation $T = \{A, t\}$ that transforms the point p^i_k into p^i_{k+1} as $A \cdot p^i_k + t = p^i_{k+1}$ (least squares, pseudo-inverse)
For artificial test data this gives the expected results.
But least-squares is known to be sensitive to outliers. It fails to compute the right rigid transform on real optical flow data coming from our videos and also on disturbed artificial data. Using more robust methods (e.g. RANSAC, [21]) it is possible to account for the noisy data and extract the right transformation parameters.

But still there might be errors in that computation (errors that come from the Lucas-Kanade tracker and errors coming from the assumption of a rigid transformation) and the generated shift of the BB leads sooner or later to the selection of slightly wrong points (some points on the bee are missing in the selection and some selected points are outside the bee area) and the following computation will be less accurate - resulting in an accumulation of errors. It is therefore desirable to have a measure of how coordinated a point-subset is moving. The tracking has to be split up into two phases. The first phase should use the optical flow tracks to shift the bounding box accordingly without taking into account the erroneous information coming from outliers. This shift can be performed 'blindly' a variable number of frames (may depend on sum of movement length). The second phase should correct the accumulated error. It has to correct the position of the bounding box by evaluating the co-ordination of point-cloud movements. Let's imagine the bounding box has been shifted according to the extracted parameters of the suspected rigid transform. After a number of frames there will be a slightly wrong orientation or position of the bounding box. Selection of the optical flow features inside the box to extract the next movement parameters would lead to wrong results. If we randomly distribute a small number of bounding boxes (hypotheses) around the original one we will most likely 'hit' the real position of the. The way of scoring this particular box better than all the others by evaluating only the optical flow data is described next.
The following section shows a maximum likelihood method that combines all requirements into one algorithm.

4.5 Hough Transform

A well known algorithm, called Hough transform, robustly finds the parameters of a given problem ([10], see also [22] for an introduction on Hough transform). Usually the task is to find the occurrence of geometrical shapes such as lines, circles, ellipses, etc. It is possible to use the same method to extract the three parameters of a rigid transform even under noisy conditions.
In this section idea of using Hough to find the right transform *and* to score the abovementioned random hypotheses will be outlined. Sequently an implementation is presented.

4.5.1 General Idea

Let $Q = \{p^1_z, p^1_{z+1}, \ldots, p^1_{z+m-1}; \ldots; p^n_z, p^n_{z+1}, \ldots, p^n_{z+m-1}\}$ be the global set of n point tracks of length m beginning at time z. Let $B = \{w, h, x, y, \varphi\}$ be an arbitrary bounding box of width w, height h, position (x,y) and orientation φ. Let

then be $Q(B) = \{p^{1'}_z, p^{1'}_{z+1}, \ldots, p^{1'}_{z+m-1}; \ldots; p^{k'}_z, p^{k'}_{z+1}, \ldots, p^{k'}_{z+m-1}\}$ the subset of Q with k point tracks where the condition "$p^{i'}_t$ *lies in B*" is met for all $i = 1..k$. Suppose $Q(B)$ holds the movement field of an object that is moved rigidly between times z and $z+m-1$. We first like to find the rigid transformation $T = \{A, t\}$ that explains best the movement of all $p^{i'}_z$ to $p^{i'}_{z+1}$; A is a rotation matrix, t is a translation vector.

Given a rotation angle α one can compute $A = (\cos\alpha\ \sin\alpha; -\sin\alpha\ \cos\alpha)$. Since the points of one point track are known before and after the transformation one can directly compute the translation vector $t = (t_x, t_y)$ given $A: (t_x, t_y) = pi'_{t+1} - A \cdot pi'_t$. Following the idea of Hough we compute all translations for all possible (feasible) rotations and increment a counter for a group of transformation parameter triples. Suppose a rotation matrix A is given for an arbitrary angle α and t_x and t_y are calculated as given above. The movement of one single optical flow feature results in a 'vote' for the triple (α, t_x, t_y). By creating a 3D matrix M of integers (called accumulator) we are able to count the votes for all rotations of all point moves by indexing M with rounded values of α, t_x and t_y. The result is an approximation of the density function that describes the probability of the (rigid transform-) parameters given the point movements. Finding the index of the maximum of M gives the most probable transform triple $T^{max} = (\alpha^{max}, t_x^{max}, t_y^{max})$.

$s = \max(M)$, the maximum itself (the number of votes for T^{max}), reflects how many point movements 'overlap' into the same accumulator cell. Given an overall number of k points that were selected using the bounding box the quotient $q = s/k$ would give the relative portion of coherently moving points. Since q is a function of the bounding box B (as well as the subset $Q(B)$) we can maximize $q(B_i)$ for a set of bounding boxes B_i $(i=1..n_b)$. The maximal scored bounding box contains the most consistent point movements and is therefore likely to contain the bee[5].

So the Hough transform gives us the means of elegantly solving the above mentioned tasks of extracting the transformation parameters as well as a measure of coherence of movement. But improvements can still be made and an efficient implementation has to be found.

Given a set of n_b bounding box hypotheses $\{B_i \mid i=1..n_b\}$. Each B_i corresponds to a score $q_i = q(B_i)$ and to a triple of transformation parameters T_i. To sharpen the assertion coming from the score q_i we expand the scoring for future movements. If we apply T_i to B_i, for $i=1..n_b$, we obtain the new positions of the B_i. Then the scores and transformations for

[5] It is better to use $q(Bi) = s^2 / n$ to weaken the score of bounding boxes containing only a few points.

the next movement z+1 to z+2 can be computed and used to shift the B_i again and so on. The resulting scores are added to an overall score which is more meaningful than a single score that reflects only one movement. As an addition to the algorithm we regard points that lie outside the bounding box (inside an area around the bounding box) as negative votes. Thus points outside the bounding box that are moving similarly to the points inside the bounding box weaken the score of their transformation. The following pseudo code outlines the idea. The function Hough computes the transformation parameters, the function correct corrects the bounding box position and is invoked after a predefined number of frames.

```
function Hough(BB)
      select points inside and close to BB
      for each selected point pi_t and its successor pi_t+1
            for all angles a from –max_angle to max_angle
                  A = rotation matrix given α
                  t_x, t_y = pi_t+1 – A*pi_t
                  if pi_t lies inside BB
                        increment Accumulator at pos (α, t_x, t_y)
                  else
                        decrement Accumulator at pos (α, t_x, t_y)
                  end
                  if Accumulator(α, t_x, t_y) > max_score
                        max_score = Accumulator(α, t_x, t_y)
                        max_transform = (α, t_x, t_y)
                  end
            end
      end
      return max_score, max_transform

function correct_position(BB)
      generate random hypotheses
      for each bounding box hypothesis b_i
            for f = 0 to number of future frames - 1
                  score, transform = Hough(b_i)
                  add score to current bounding box score
                  shift b_i according to transform
            end
      end
      BB = get_the_max_scored_bounding_box()
```

Obviously this is not feasible since the size of the accumulator has to be large and only a small percentage of the memory would be used[6]. Making small changes to the algorithm saves more than 95% of the memory.

4.5.2 Implementation

Generally the accumulator `M` is sparsely filled with entries unequal 0. That means we waste lots of memory by using an explicit accumulator.

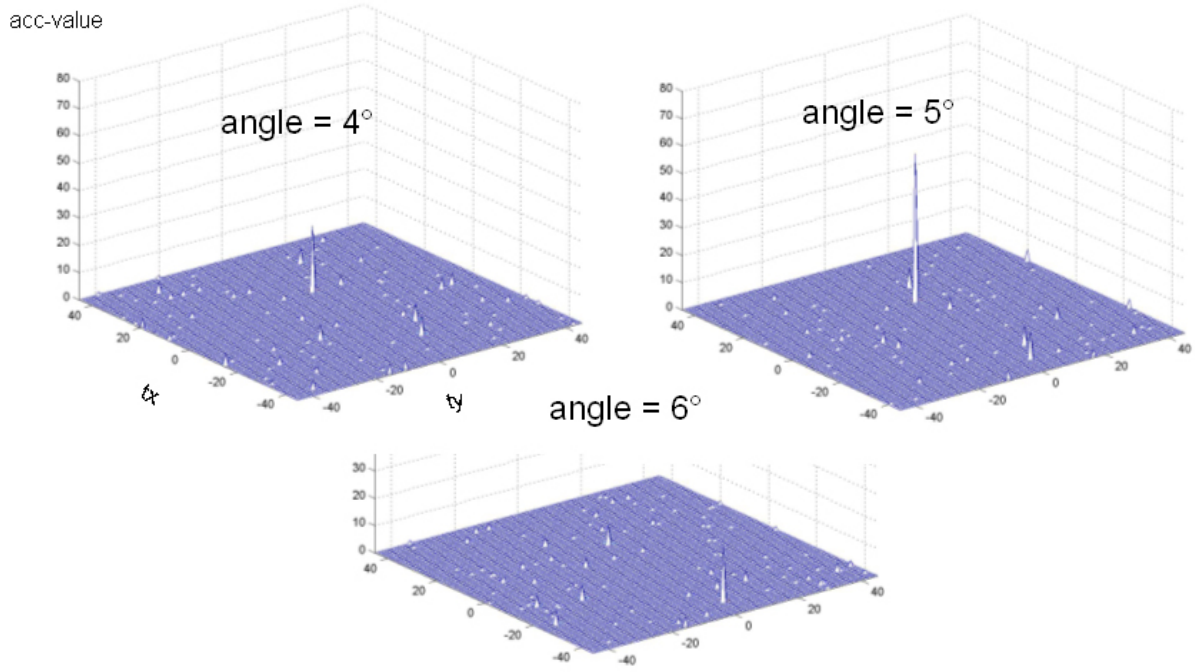

Figure 4-6 : The Hough accumulator for three angles. The maximum is very sharp, the accumulator is only sparsely filled.

To reduce the size of `M` one can transform the points (that are addressed in the image coordinate system) to the local bounding box coordinate system which reduces the resulting t_x and t_y to smaller absolute values and therefore the size of `M` (see Figure 4-6 for an example accumulator using this method) to 5% of what was used with the former method[7].

The final ansatz however is using a hash table to count / score the resulting transformation parameters (α, t_x, t_y). By rounding α, t_x and t_y to a suitable level of precision it is

[6] Suppose a point track p^1_t = (400,300), p^1_{t+1} = (401, 300). Then the resulting translation vector for a given rotation of 35 degrees would be t = (tx, ty) = (205, -159). That would require a size of of the accumulator of 35 MB.

[7] Maximal translation vectors using features in the bounding box coordinate system range between -40 to 40 degrees. Therefore the size of the accumulator may be reduced to 1.7 MB.

possible to create a 64 bit key[8] that consists of 21 bit signed integer values for each the parameters. One bit remains unused (see Figure 4-7).

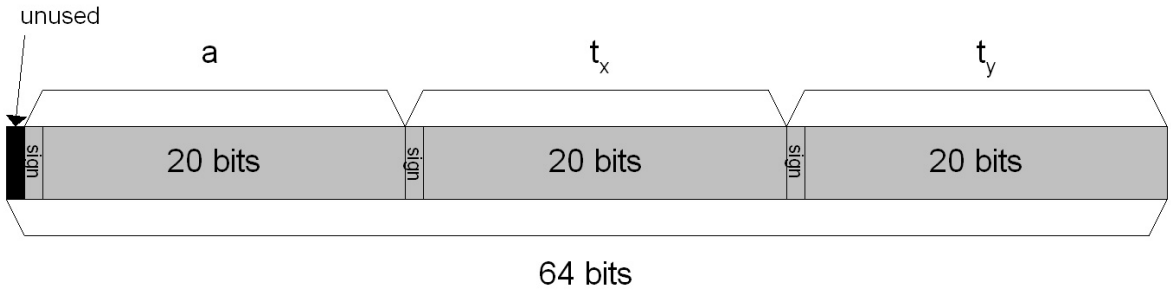

Figure 4-7 : The three parameters are put into one 64 bit key.

21 bits can display numbers up/down to $\pm 2^{20} - 1$ (which is a range of approx. 1 million each direction from 0). This is enough to fit also values rounded to 1/100[9].
The hash function used is the well known sdbm function (used also in BDB, Berkeley Database) – a fast hashing algorithm.
Since the hash value of the 64 bit key is still 32 bit long, we have to reduce the range of it for being able to use the hash as an index for the hash table. Usually one looks for a feasibly big prime number and takes the rest of the integer division as the index for the hash table. But since this is slow, it is also possible to bitmask only one part of the 32 bit hash. The bit mask has to be big enough to avoid collisions in the hash table but also small enough to still save memory compared to the previous method. For the implementation we used a 16 bits bit mask. Compared to the first implementation ansatz we save again 80% of the memory for the accumulator. The speed of computation is approximately equal.

5 Results

The above described algorithm has been implemented in Visual C++. A GUI has been written that enables the user to load video files, set a bounding box variable in size on an arbitrary bee, track it manually, semi-automatically or automatically. The trajectories can be displayed, altered by simple mouse input and saved to a text file. Previously saved trajectories can be loaded into the program to be viewed or changed. All input such as seeking, playing, stopping the video file or moving the bounding box can either be made by keyboard input or using the

[8] Using 32 bit keys yields 2 unused bits and 3 10-bit integers for the parameters. This is too small to fit the values for the tx and ty – depending on the image size they might get bigger than 2^9 = 512.
[9] We may want to avoid this high precision since we need the resulting values for t_x and t_y to fall into same accumulator cells and not to be separated.

mouse. A video with overlaid bounding box and trajectory can be exported to an avi file of many compression methods. The current program version can be downloaded from http://page.mi.fu-berlin.de/landgraf.

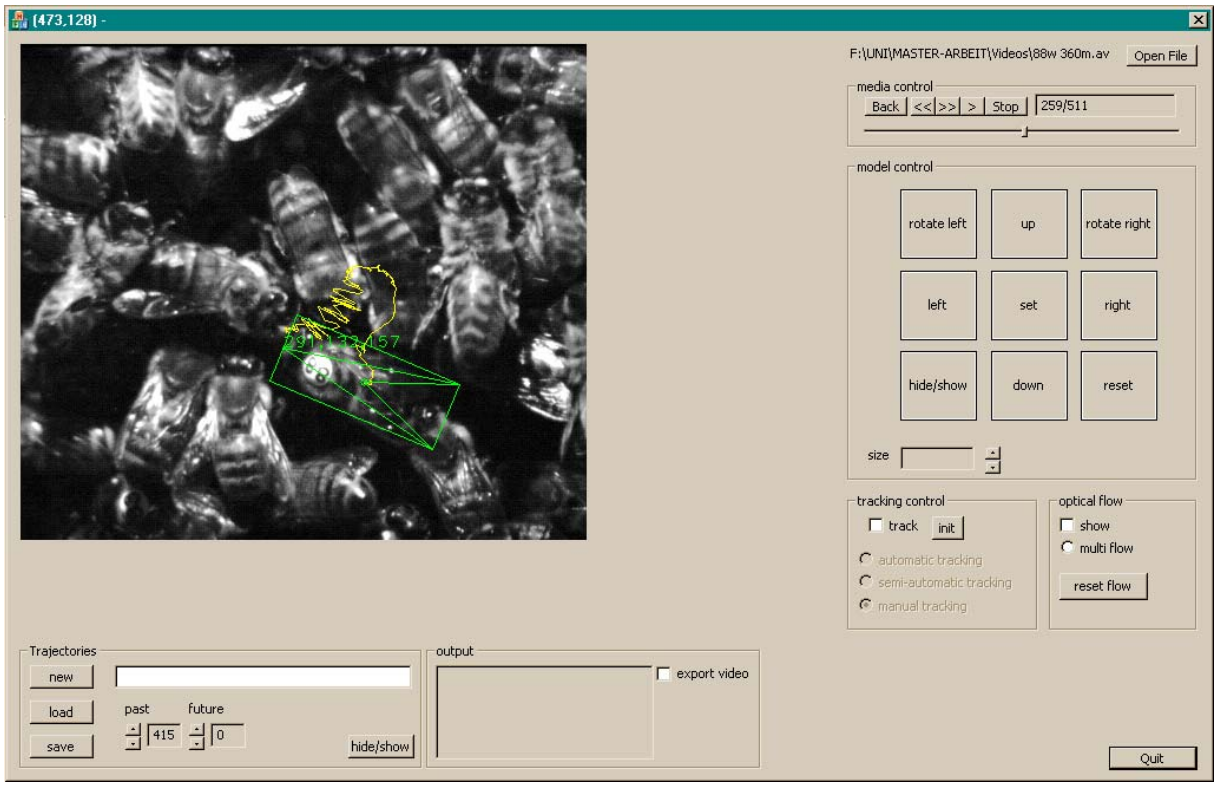

Figure 5-1 : Screenshot of the tracking program. The trajectory of a waggle dancing bee is displayed.

5.1 Tracking Quality

Videos (old and new ones) have been subjected to the automatic tracking. The initial position was set manually and the tracking was initiated. Mostly 10 to 15 future frames were chosen for the optical flow computation and each correction step was performed after 5 – 10 frames. The duration of computation was measured and the resulting trajectories saved.
In parallel the same videos were tracked manually[10] using the program to create a reference target. Reference trajectories and automatically computed trajectories were compared.
The tracking quality might be expressed in the Euclidian distance of the bounding box vector to the reference[11]. For the sake of clarity we will nonetheless keep apart the calculation of the x-y-error and the orientation error. The former will be expressed in percentage of the

[10] Semi-automatic tracking was used. That means that the movement of the bounding box was computed for a variable number of frames until the human observer recognizes a wrong lay of the box on the bee. Then the correction was made manually.
[11] Therefore one would have to translate the orientation angle into a unit vector.

bounding box diagonal[12], the latter in degrees. This will keep the results comparable since different videos exhibit different bee - and therefore bounding box sizes.
Following Figures 7-2 to 7-6 relate to results obtained using one of the new videos. Figures 7-7 to 7-9 show results from tracking one of the old videos.
Figure 5-2 depicts the x-y-trajectories obtained from automatic and manual tracking. They roughly look the same. To visualize the error made by automatic tracking Figure 5-3 and Figure 5-4 show the error of the planar positions and orientations, respectively. Figure 5-5 and Figure 5-6 compares the bounding boxes at the time of maximum errors for the automatically tracked and reference trajectories to give an impression of how bad the tracking becomes at worst.
The mean error of position of that particular video sequence is approximately 10% of the length of the diagonal of the bounding box. The bounding box never left the body of the bee. The mean error of the orientation angle is about 7°. This does not sound a lot. In coaction with positional errors this nevertheless can become a subjectively not viable result.
Generally the position and orientation of the bee was successfully followed until the end of the sequence when the bee left the frame borders.

The tracking results of the old video show slightly lower positional errors which may be explained by the bigger bee appearance due to a smaller distance of the camera to the hive. Thus the number of optical flow features on the bee in average is higher. Oppositely the error of the orientation angle is slightly higher which may relate to the higher crowdedness of the hive in that particular video. The correction step of the algorithm has its problems of finding the correct position if nearby bees move similarly to the tracked bee. The tracker might have correctly identified the location of the bee but was distracted only in finding the correct orientation.

Generally the results in the whole sets of videos are similar to what is shown here. Mean positional errors of 5%-13% and mean orientation errors of 7° - 12° were found for the videos that were tracked throughout the full sequence. There are indeed videos that let the tracker fail. Mostly this is due to occlusions, motion blur in the waggle phase and similar serious distractions. There are also some sequences that do not show above described difficulties and nonetheless let the tracker fail. Obviously wrongly corrected positions lead to the wrong selection of optical flow points that in turn deliver the wrong parameters – exactly what we

[12] The ratio width/height of all sizes of bounding boxes is fixed.

wanted to avoid. But these situations are rare. The videos have to be sharp and recorded sufficiently close to the hive to obtain reliable flow fields and therewith a reliable automatic tracking. Furthermore for the correction step the search space was rather sparse, meaning that the generation of more hypotheses or the use of a particle filter would have led to better results.

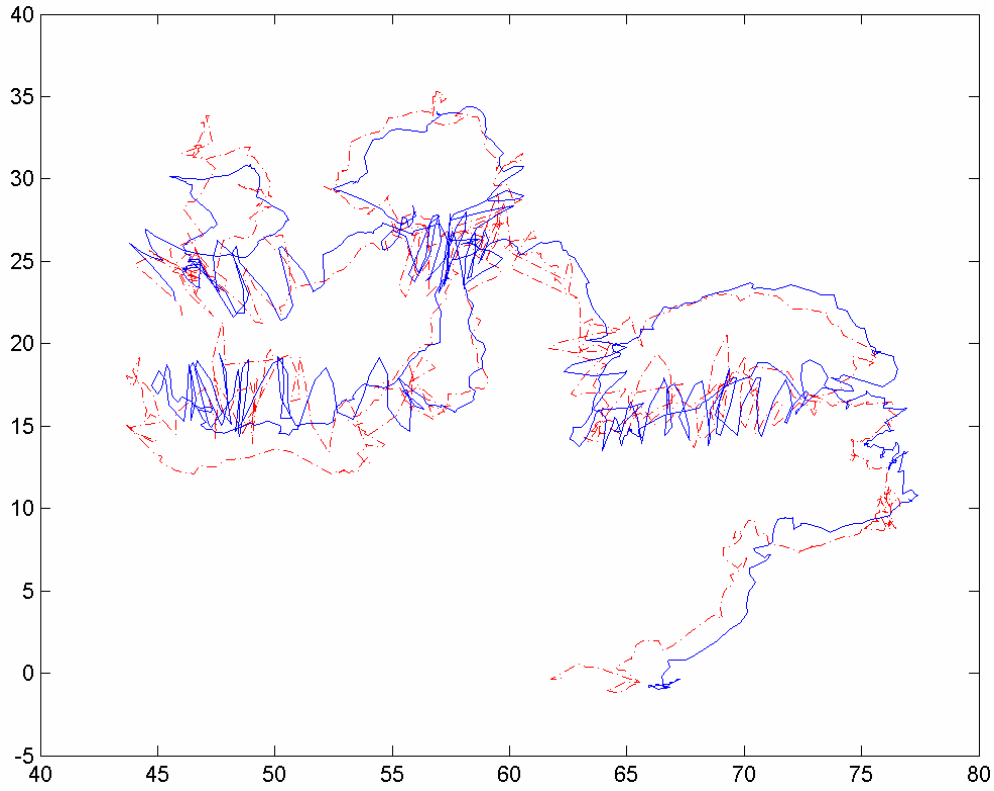

Figure 5-2 : Automatically (red, dashed) and manually (blue) tracked trajectory. Axis units in millimetres. Length of video : 1383 frames. At the end of the sequence the bee leaves the frame borders.

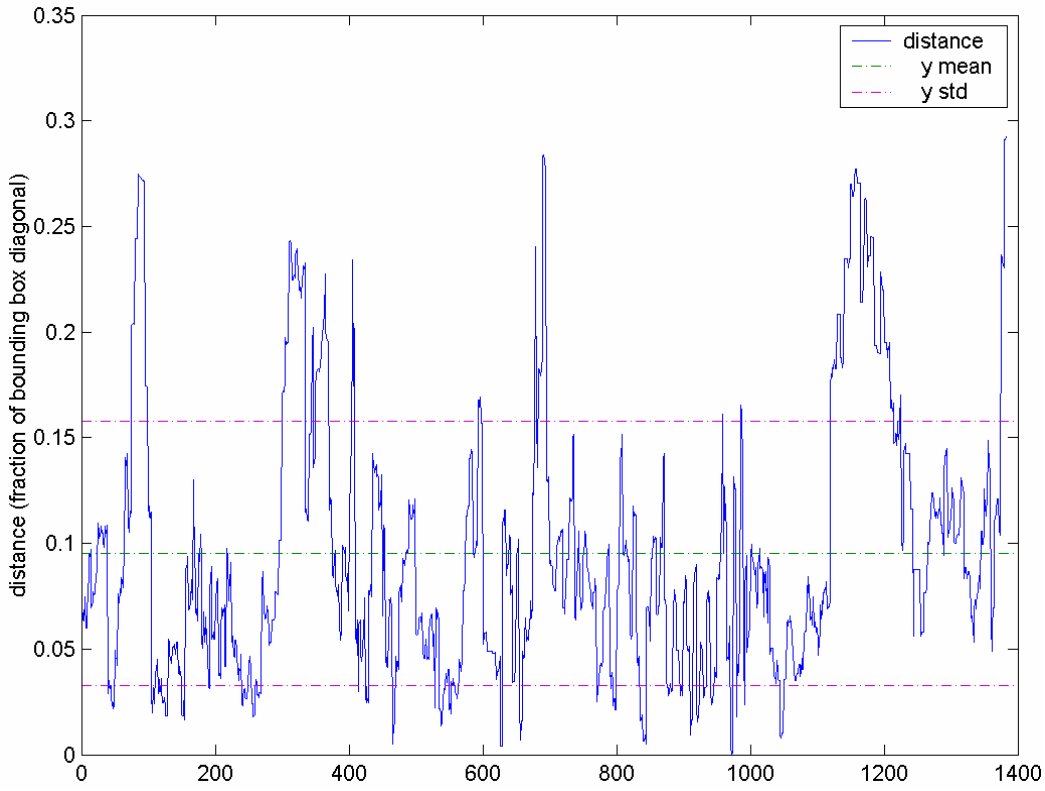

Figure 5-3 : Relative distance of automatically found bounding box positions to manually set reference (fraction of bounding box diagonal) for first video (1383 frames, new video, 89 fps). Statistics: Min = 0, Max = 0.3, Mean = 0.1, Std = 0.06 (rounded).

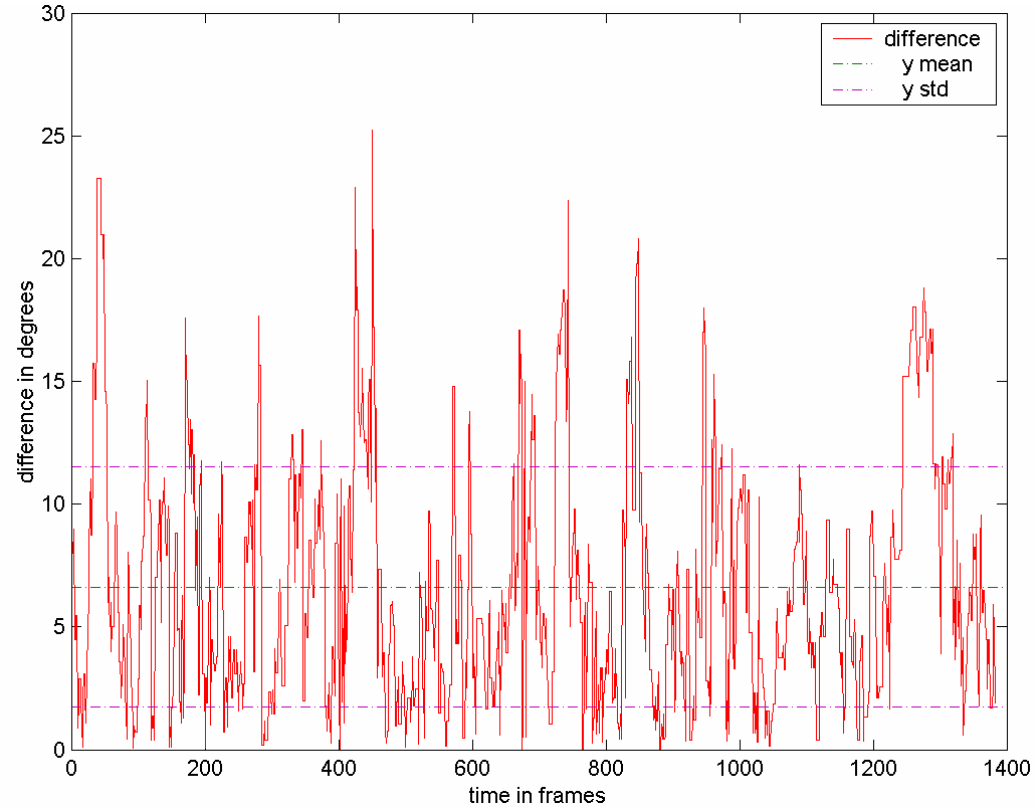

Figure 5-4 : Absolute differences of the orientations found automatically to the reference bounding box orientations for first video. Statistics: Min = 0, Max = 25, Mean = 7, Std = 5 (rounded).

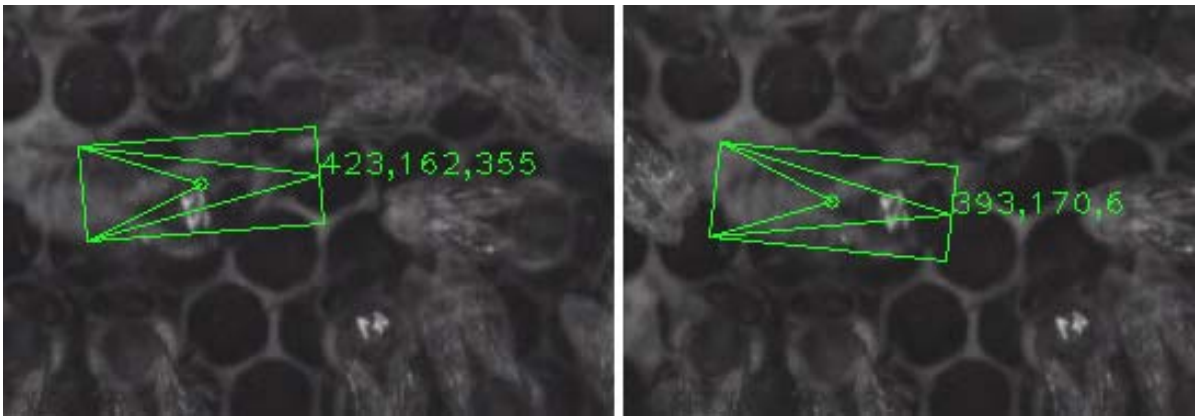

Figure 5-5 : Bounding box positions (left : automatic tracking, right : reference) at t = 690 resulted in the maximum error of approximately 30 % of the bounding box diagonal. Automatic tracking recovered the erroneous position in the next correction step.

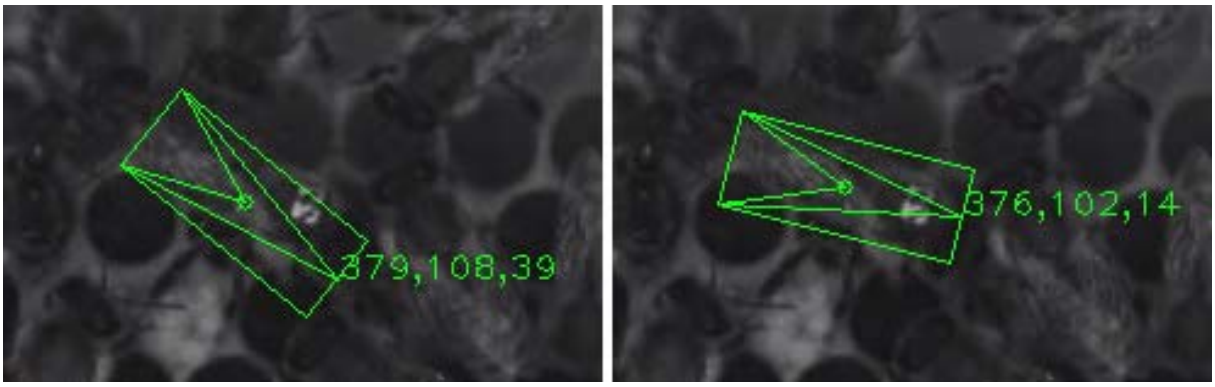

Figure 5-6 : Bounding box orientations (left : automatic tracking, right : reference) at t = 451 resulted in the maximum error of approx. 25°.

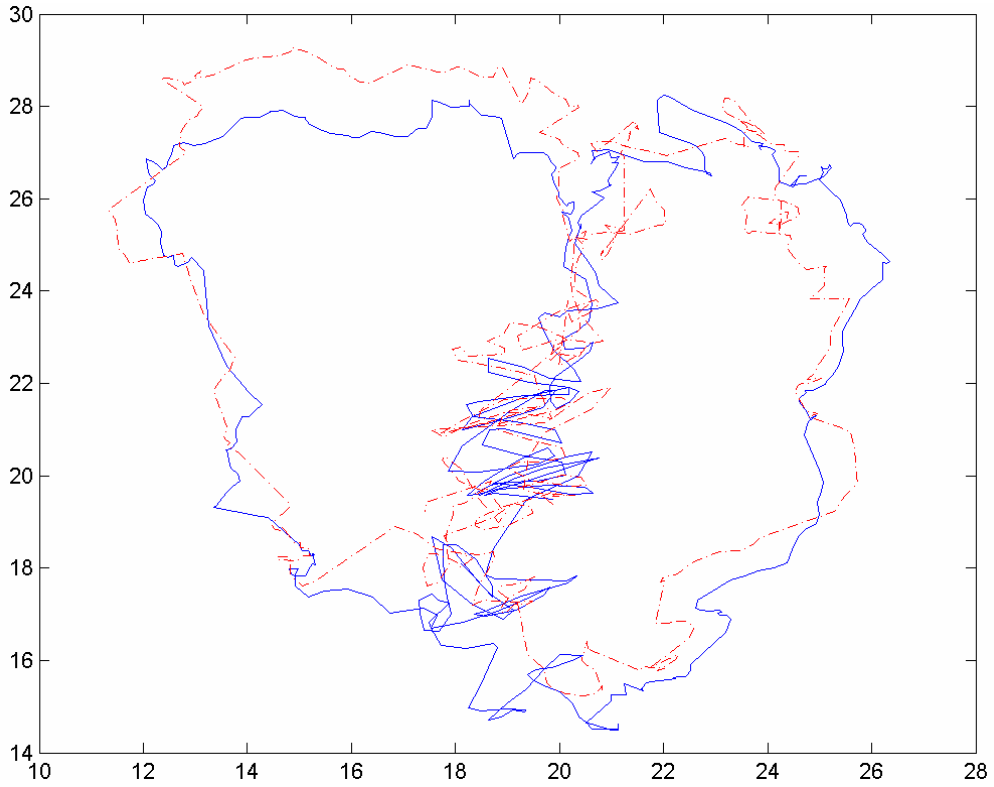

Figure 5-7 : Automatically (red, dashed) and manually (blue) tracked trajectory. Axis units in millimetres. Length of video : 512 frames. Frame rate : 125 fps. Note the clear figure 8 shape that was missing in the previous dance plot.

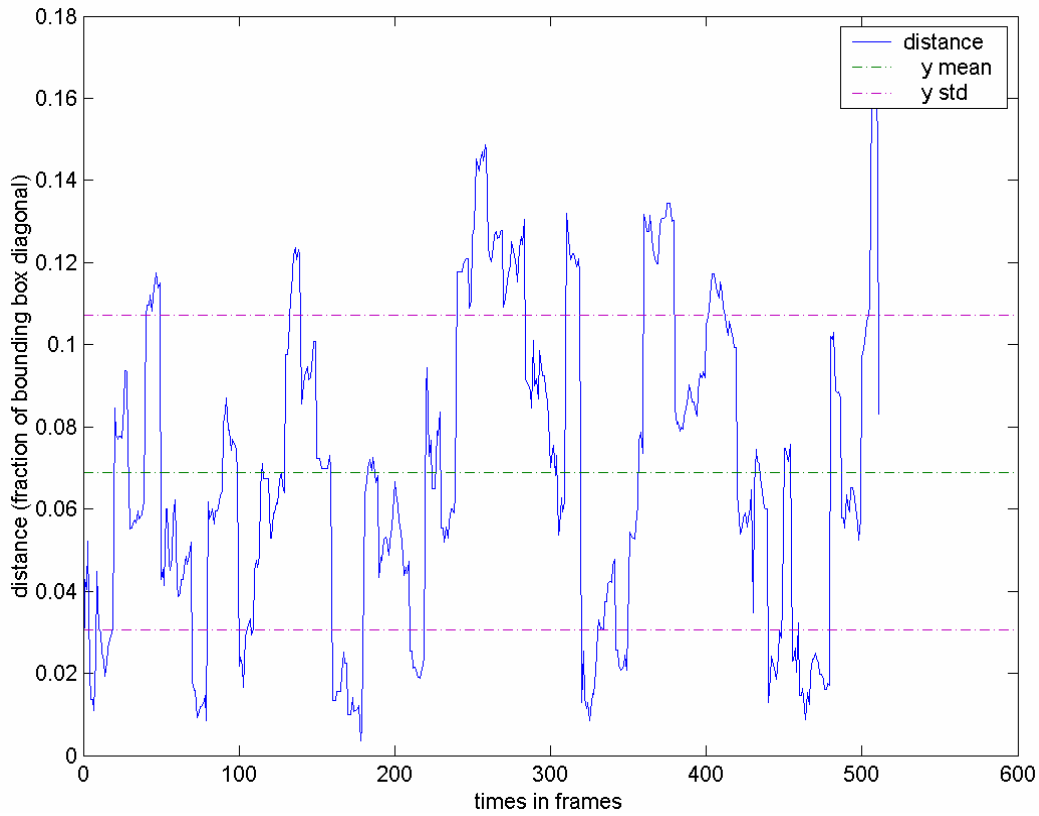

Figure 5-8 : Relative distance of automatically found bounding box positions to manually set reference (fraction of bounding box diagonal) for second video (512 frames, old video). Statistics: Min = 0, Max = 0.18, Mean = 0.07, Std = 0.04 (rounded).

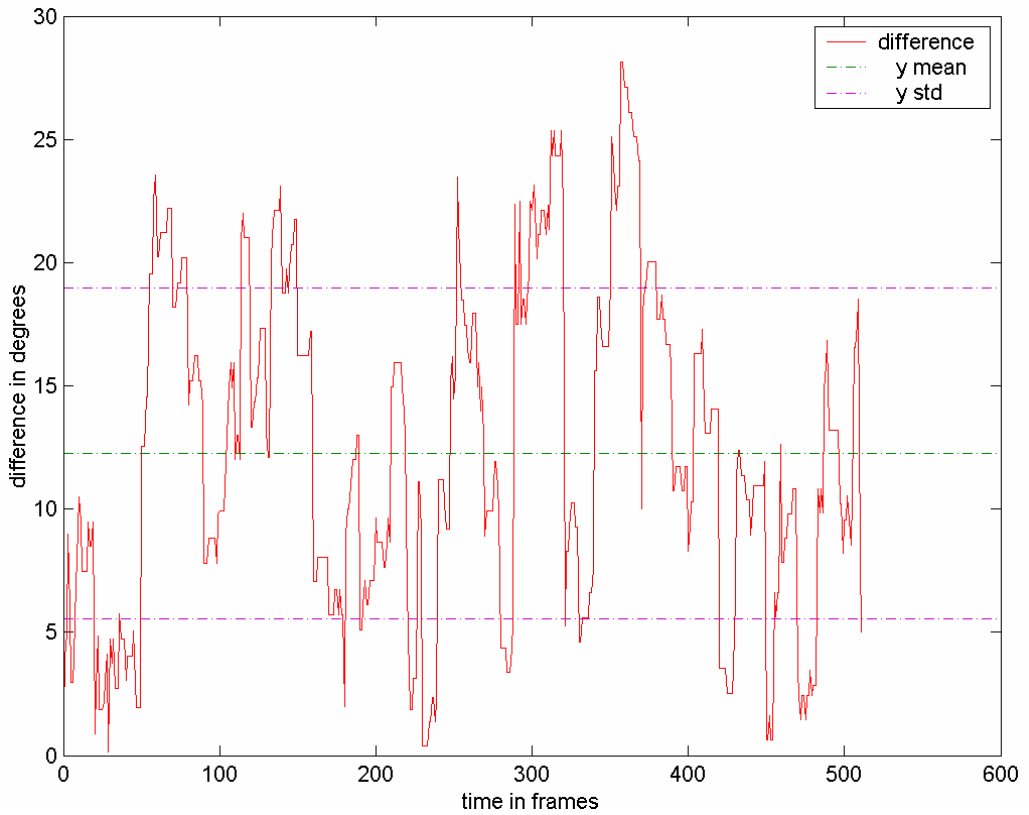

Figure 5-9 : Absolute differences of the orientations found automatically to the reference bounding box orientations for second video. Statistics: Min = 0, Max = 28, Mean = 12, Std = 7 (rounded).

5.2 Tracking Speed

Tracking a 512 frames sequence using 10 future frames and correction steps each 5th frame takes 5 min 41 sec on a Pentium 4, 2.6GHz, 1GB RAM. This yields a frame rate of 1.5 fps. The correction step takes almost 3 times as long as one normal step (average values). The speed depends also on the number of hypotheses that are created and, if used, the number of iterations of the particle filter. The more hypotheses the more precise the result but the more time is consumed.

5.3 Error Analysis

Regarding the functioning of the algorithm there are different origins of errors made. Firstly the computation of the transformation can be erroneous because of errors coming from the optical flow tracker. Secondly the assumption of rigidity of movements performed by the bees is not true always. Thus the extraction of the transformation parameters might generate erroneous results. Thirdly the correction step can produce erroneous shifts of the bounding

box. Here errors coming from the optical flow tracker and errors coming from the assumption of rigidity may converge.
Rolling or wagging movements are often introducing errors to the flow computation.
There are, especially in the beginning and ending of a return phase, (non-rigid) bending movements when the bee is turning. Moreover there are wing movements that can induce distractions of the tracker[13]. If the number of equally moving points on the wing is higher than on the abdomen the wing movement is interpreted as a turn-like movement.
Then, if follower bees in close vicinity exhibit the same movements as the dancer, the correction step shifts the bounding box onto the two bees if the number of optical flow features generates a higher score than the features on the dancer alone. Fortunately these situations are seldom.
Sometimes the lighting of the scenery leads to shadows on one side of the bee body inducing a feature distribution only in the half that is lighted. Here the assumption of equal distribution is not given and the bounding box is shifted in the correction step onto the lighted half only.

6 Enhancements

Regarding the speed of computation it is clear that there has to be added more effort to make the automatic tracking faster. Furthermore there are possible enhancements that expand the scope of the program and might increase tracking quality. Some ideas are presented in the following sections.

6.1 Motion and Error Model

One of the biggest parts of the computation time is occupied by scoring the random hypotheses. Since no information of the previous movements is integrated in the generation of the hypotheses there are many redundant and futile bounding box positions that consume time for scoring. Using a motion model one would be able to limit the scatter area to a range that can be chosen by evaluation of the previous motion. Then one can use fewer hypotheses which in turn reduces time consumption. But a motion model has to take into account that there are different classes of motion patterns (see previous section on error analysis). The waggle run has some distinct properties as the more or less fixed frequency and the lateral movements that are performed with a slight roll. This roll induces wrong results from the computation of transformation parameters as well as in the correction step. If the waggle run is detected (simply by detecting alternating lateral movements) the error resulting from roll

[13] Sometimes the bees spread only one or even both wings for a few miliseconds.

movements may be modelled and equalized. The same idea can be applied to the class of turning movements.

6.2 Particle filters

Particle filters (such as the CONDENSATION algorithm) have been proposed recently. The goal of using these is to be able to track multiple hypotheses throughout the sequence. The probability of hypotheses representing the bee's position given the observation (image) can have more than one (local) maximum - especially in the case of a crowded bee hive. Using particle filters it is possible to manage this multi-modal distribution.
Suppose only one maximum has been tracked and due to errors this maximum 'washes out' throughout tracking due to the above described difficulties. Then in the correction step a set of hypotheses would be generated randomly and wrong bounding boxes that lie partly on a nearby bee with better scoring flow features will be chosen as the next maximum.
We would have lost track. The particle filter in turn would still retain the former maximum (that now may be lower than other surrounding maxima) and continue tracking it. We could choose which local maximum is more likely to be the bee.
Particle filters have been implemented but not integrated into the program yet. The use of particle filters is highly probable to make the tracking much more precise and robust.

6.3 Automatic bee recognition

It would be desirable to find bees automatically to select for tracking. Imagine a marked bee returns from foraging, enters the hive and begins dancing right away – researchers would only have to record the whole experimental period and start the tracking at the end of the day. The marked bee is detected, the tracking is initiated automatically and stopped if the bee exits the hive.
Or the biologist marks a bee already in the hive using a laser pointer and the program first finds the laser spot and then refines the position and orientation of the bee.
The first task would require a greater viewport of the camera. If this would have been established a region near the entrance may be selected where arriving bees are subjected to the recognition algorithm.
Tracking only the bee's position in a greater area may have different requirements than tracking the bee dance in full detail. For this thesis the focus was set on highest precision possible to provide insights into the dance.

6.4 Automatic path labeling

It might be desirable to segment the dance trajectory into different parts, such as waggle run, left return phase, right return phase, etc. The respective information can be obtained by online evaluation of the trajectory obtained. It might be used for generalization and visualization of the dance in further work steps. Figure 6-1 shows the orientation angles for a waggle run. Although the waggle consists also of strong lateral movements of the bee body it might be identified by evaluating only the orientation angles.

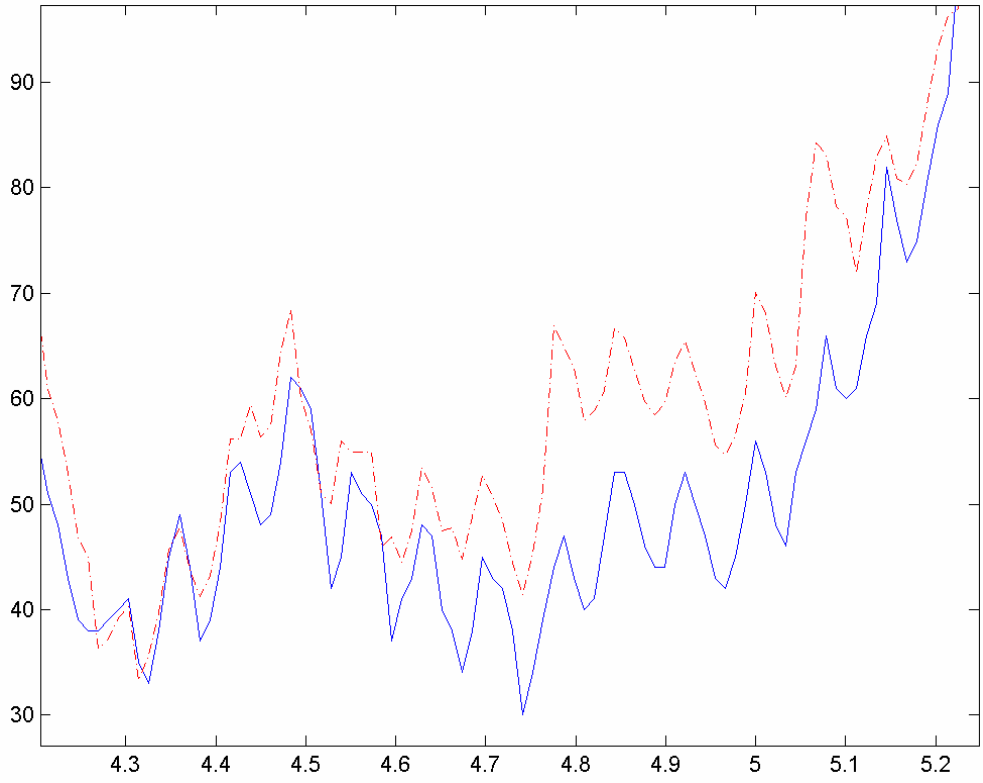

Figure 6-1 : Orientation angles of the automatically tracked bounding box (red, dashed) and the reference (blue) for a waggle run over time (in seconds). It might be easily identified by recognition of oscillations.

6.5 Tree representation of optical flow points

The work with optical flow points includes some time consuming operations such as searching of a selection of points lying in vicinity of others. To effectively execute such commands it may be of great advantage to manage the points not linearly but in a tree structure where the points are sorted by 2D location and the search runs in a logarithmic time scale.

7 Discussion

There are 2 findings in this thesis.

First, it proved to be sufficient using only optical flow to track bees throughout the whole video. The tracking is robust and independent of additional information, such as template images. Videos of low image quality were tracked resulting in equally low errors as using higher quality sequences.

Secondly, an algorithm was proposed that successfully tracks the movement of approximately rectangle shaped objects from optical flow data. The transformation parameters are extracted as well as a measure of co-ordination of movement of the optical flow features - in the same algorithm. The results show that the automatic tracking is able to track honey bees throughout a full dance and longer.

But since the assumption of rigid movement is not true in every situation there are errors to cope with. The automatic tracking can correct errors made. But to gather as exact coordinates of the bee as possible, one has to either re-view and correct manually the trajectory found or use the semi-automatic tracking. Herein the correction step is not statically performed after a fixed number of frames – it is done by the user, corresponding to human rating of the bounding box position.

For biologists it is advised to use the semi-automatic tracking which already saves a lot of time and gives high precision. It is usual in any way that the tracking result is reviewed and corrected optionally by a human. This can be done right while extracting the data – with no great time consumption.

The tracking depends highly on the quality of the video. In many cases it is not possible to automatically track the position and orientation of the bee with satisfying quality because of many different factors. The two sets of videos, the older ones recorded 2005 and the newer ones from 2006 are similarly working with the algorithm. We have got a notion that the extraction of transformation parameters from optical flow works better with the former ones since the frame rate and the size of the bee are high (~125 fps and ~ 60 pixel x 135 pixel). The newer videos have a slightly lower frame rate (~90 fps) and furthermore a lower size of the bee image[14]. But the image quality is higher and the reflections of the wings – one of the main reasons of the failure of correlation – are less numerous and pronounced. That may also favor the use of correlation techniques in the new videos. A correlation tracker that uses information from the optical flow to reduce the search space of the correlation may probably work best on all new videos, but still would need a template definition though. Particle filters will probably improve tracking results for both sets of videos.

[14] Talking in mean values. The videos have not been recorded from always the same distance to the hive (indeed there are some videos with bigger bee images than 60x135 pixels).

The Optical-Flow-Hough algorithm can be improved. There are still many redundant computations. It is imaginable to mix the correction step with the transformation extracted from the feature movement in every frame. Doing this the error from the parameter extraction is adjusted and the 'jumps' coming from the correction step would be smoothed. But then one needs to increase speed of computation. Then it may be thinkable to use the quality measure of the optical flow features in the scoring of the Hough algorithm. Features may have higher votes the higher the quality of the feature track is. Furthermore the observation of motion blur in the waggle run leads to the notion that only features at the sides of the bee might have good enough quality – thus only these features may be selected for the computation of transformation parameters.

It might be also a good idea to use another feature selector, such as SIFT or SURF (see [23], [24], respectively) or even define a set of only a few distinctive features we expect to occur on bees. These pre-stored custom features can be represented in an eigenspace (see [27]) and serve as template patches. Since methods exist that can use these features scale-invariantly no template definition would have to be made by the user. These features are expected to be much more robust. The bounding box can be associated to each custom feature in a predefined way. The features lose their 'anonymity'. They will be related to only a particular region on the bee such as "the corner right above the right eye" or "the region where thorax and abdomen touch". Their position relative to the bounding box is defined in the first frame and subsequently re-established in the tracking.

For now, the program is already being used to extract trajectories. We are questioning the statistical properties of the waggle run such as the variance of amplitude of the waggle oscillation to quantify the precision with which the dance is conveying information.

8 References

[0] J B S Haldane. Aristotle's Account of Bees' 'Dances'. The Journal of Hellenic Studies, Vol. 75. 1955

[1] B G Whitfield. Aristotle and the Dance of Bees. The Classical Review. Vol. 8 (No. 1.). 1958

[2] Donald R Griffin. Animal Minds (chapter 10: symbolic communication). 2001.

[3] Hauser and Konishi. The Dance Language of Honey Bees: Recent Findings and Problems. The Design of Animal Communication. 1999.

[4] A Michelsen, B Andersen, J Storm, W Kirchner, M Lindauer. How honeybees perceive communication dances, studied by means of a mechanical model. 1992.

[5] J B S Haldane, H Spurway. A statistical analysis of communication in 'Apis Mellifera' and a comparison with communication in other animals. 1954.

[6] B Horn, B Schunk. Determining Optical Flow. 1981.

[7] B Lucas and T Kanade. An iterative image registration technique with an

application to stereo vision. 1981.

[8] J Shi, C Tomasi. Good Features to Track. 1994.

[9] J Y Bouguet. Pyramidal Implementation of the Lucas Kanade Feature Tracker. 2000.

[10] P V C Hough. Methods and Means for Recognizing Complex Patterns. US patent. 1992.

[11] Z Khan, T Balch, F Dellaert. A Rao-Blackwellized particle filter for EigenTracking. Computer Vision and Pattern Recognition. CVPR 2004.

[12] A Veeraraghavan, R Chellappa. Tracking Bees in a Hive. 2005.

[13] M Isard, A Blake. CONDENSATION – conditional density propagation. IJCV. 1998.

[14] A M Wenner. The Elusive Honey Bee Dance "Language" Hypothesis. Springer. 2002.

[15] CMU 1394 Digital Camera Driver. http://www.cs.cmu.edu/~iwan/1394

[16] Active Contours. A Blake, M Isard. Springer. 1998.

[17] VIDEO TRACKING SOFTWARE. http://www.qubitsystems.com

[18] EthoVision. http://www.noldus.com

[19] M Lindauer. Schwarmbienen auf Wohnungssuche. Z. vergl. Physiol. 37, 263-324. 1955.

[20] J Canny. A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1986.

[21] M A Fischler. R C Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. 1981.

[22] T Landgraf. Bachelor's thesis: Detection and tracking of antennal movements for the analysis of sleep like behaviour in the honey bee. 2003.

[23] D G Lowe. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision. 2004.

[24] H Bay, T Tuytelaars, L V Gool. SURF: Speeded Up Robust Features. Proceedings of the ninth European Conference on Computer Vision. 2006.

[25] K von Frisch. Die Tänze der Bienen. Österr Zool Z 1:1-48. 1946

[26] OpenCV. Open Source Computer Vision Library. http://www.intel.com/technology/computing/opencv/index.htm.

[27] Y Ke, R Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. Proc. CVPR. 2004.