Vision, Modeling, and Visualization (2014)
J. Bender, A. Kuijper, T. von Landesberger, H. Theisel and P. Urban (Eds.)

High-Speed Object Tracking Using an Asynchronous Temporal Contrast Sensor

D. Saner[1,2], O. Wang[1], S. Heinzle[1], Y. Pritch[1], A. Smolic[1], A. Sorkine-Hornung[1] and M. Gross[1,2]

[1]Disney Research Zurich, Switzerland
[2]ETH Zurich, Switzerland

Abstract

Purely image-based approaches to tracking objects in video sequences are more prone to failure the higher an object's speed, as it covers a greater distance over the camera's exposure time and is subject to increased motion blur. These algorithms are also susceptible to changes in the object's appearance and illumination. In this paper, we approach this problem by asynchronously recording local contrast change at high speed, using a type of visual capture device called a Dynamic Vision Sensor (DVS). We use this additional data to determine motion that takes place during the exposure of a single video frame. We present a multi-modal capture system incorporating both a DVS and a traditional video camera, a method to register the different types of sensor information, and finally apply these datasets to challenging object tracking scenarios.

Categories and Subject Descriptors (according to ACM CCS): I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Sensor fusion I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking

1. Introduction

Tracking the motion of objects across frames of a video sequence is at the heart of many applications in computer vision. If the recorded video sequence is the only data available, it involves finding corresponding features between subsequent frames. As each image is exposed over a certain timespan, it integrates an object's motion over that period. The problem thus becomes more difficult the larger the distance an object covers per frame. This is due to larger search spaces required, potential appearance and illumination change, and motion blur.

One way of tackling these issues is the use of high-framerate cameras. However, the additional information to support feature tracking algorithms comes at the cost of a steep increase in the amount of high-resolution data to be stored and processed, as well as putting high requirements on scene lighting.

Our proposed approach involves the application of a novel type of visual sensor called a Dynamic Vision Sensor (DVS) [LPD08]. A DVS registers local contrast change at a very high temporal resolution. These changes are exported as asynchronous events, consisting of a timestamp, pixel coordinates, and the direction of change (i.e., increasing or decreasing intensity). As the DVS does not expose entire frames, events can be emitted immediately with very low latency. The model used in our experiments provides a latency that allows each sensor pixel to fire events at more than 65,000 Hz.

In our experimental setup, we combine a DVS with a traditional video camera in a beamsplitter configuration, registering the coordinate systems of the two sensors. This allows us to correlate events from the DVS with pixels in the full color video. By synchronizing the camera's framerate with the internal clock of the DVS, we can also determine which frame was being exposed as a specific contrast change event was triggered. Thus spatially and temporally calibrated, we receive a point cloud of contrast change events at a very high temporal resolution, during and in between the exposures of video frames (fig. 1). As objects moving through the scene generate a very dense trail of contrast change events, we can use this information to determine where they have moved in between the completed exposure of one frame and the next. Our contributions include insight into fusing the different modalities of data provided by the video camera and the DVS to calibrate the setup, as well as a novel, frameless, spline-based tracking approach suitable for the data provided by the DVS.

DOI: 10.2312/vmv.20141280

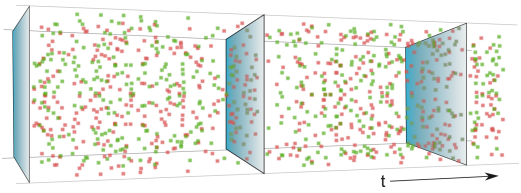


Figure 1: *A video camera records full frames at regular intervals (blue planes). The DVS registers local change as it happens, and delivers events asynchronously to the client. These events (red and green) can fall anywhere within the full 3D space-time volume, at low spatial, but very high temporal resolution.*

Section 3 presents our hardware setup and data capture workflow, while section 4 explains the method we use to register the coordinate systems of the two sensors to a common basis. Section 5 then goes on to explain our approach at object tracking in recorded videos. We show results in section 6, compare them to those of a state-of-the-art image-based tracking algorithm in particularly challenging scenarios, and discuss limitations of our own approach. We explore possibilities for future work and extensions in section 7, before concluding in section 8.

2. Related work

Developments of bio-inspired, event-based visual sensors are driven by the need for data that is available with low latency, free of redundance (i.e. information about static parts of the image), and at extreme lighting conditions [RHK*03]. An early description of the DVS concept is found in [AMSB95]. The DVS used in our experiments is detailed in [LPD08], along with an extensive list of references on similar developments. Our work focuses on applying the unique features of DVS data to augment rather than replace traditional frame-based video.

There has been previous research into applying DVS to tracking coded LED markers for motion capture [Hof12] and pose estimation for flying robots [CSB*13]. Cardinale explored object tracking by tracking clusters of DVS events [Car06]. In contrast to our approach, these experiments rely on DVS data exclusively, while our method supports tracking any object visible in a simultaneously recorded video from a traditional camera.

Extensive research has been performed regarding the alignment of the optical axes of several cameras to capture more data of a single view. McGuire et al. have aligned as many as 8 cameras in an optical splitting tree, increasing the dynamic range or spatial resolution of the captured view, or supersampling data by varying other parameters of the cameras [MMP*07]. In this work, we apply the concept of monocular multi-sensor setups to combine two fundamentally different types of visual sensors.

Setups combining sensors of different types are called multi-modal. Wang et al. combine two video cameras, one sensitive to visible light, the other to infrared, in a beamsplitter setup similar to the one presented in this work [WDC*08]. Their system uses additional infrared lighting in order to improve the illumination and color balance of the main video picture. Joshi et al. add gyroscopes and accelerometers to cameras in order to deblur images shot from a moving camera [JKZS10], and hence address issues related to motion of the viewpoint rather than of the captured objects.

Image registration is the alignment or superposition of data from multiple sensors. Multi-modal image registration is most prevalent in medical imaging, where different scans of the same body area are combined to form a complete picture. These and other use cases are presented in the image registration technique survey by Brown [Bro92]. Both [Sze06] and [ZF03] include extensive reference sections.

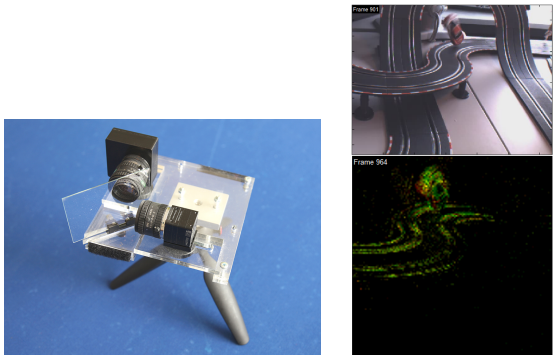
Yilmaz et al. provide a comprehensive survey and classification of image-based approaches to object tracking [YJS06]. We compare our results to a state-of-the-art tracking algorithm by Zhang et al. [ZZY12]. Zhou et al. tackle problems of object tracking systems by fusing features and multiple regular cameras [ZA06]. Our system is based on adding sensor information traditional cameras are not able to capture.

3. Hardware setup and acquisition

The DVS used in our experiments is the DVS128 developed at the Institute of Neuroinformatics, University of Zurich (`http://siliconretina.ini.uzh.ch/`). Its array has a spatial dimension of 128×128 pixels, and changes in intensity can be detected at lighting levels ranging from direct sunlight to as low as 0.1 lux. As contrast change is registered locally for each pixel, the entire dynamic range can be present in the scene at once, whereas a video camera with a fixed exposure time would either underexpose or saturate certain areas. With a latency of $15\,\mu$s, the theoretical upper limit is upwards of 65,000 events per second and pixel [LPD08].

Our setup combines the DVS128 with a Ximea xiQ USB3 color video camera recording up to 60 progressive frames per second at a resolution of 1280×1024 pixels. The camera and DVS128 are mounted on a mounting plate, along with a 50/50 beamsplitter to align their optical axes so that they record the same view of the scene (fig. 2). As the DVS detects contrast change relative to the absolute intensity, it is less dependent on a sufficiently lit scene, and thus a beamsplitter that directs more light towards the video camera could also be used. The two devices are also connected with a cable for temporal synchronization (see section 4).

A pixel of the DVS generates a signal spike whenever it



(a) (b)

Figure 2: (a) *Capture rig with DVS128 on the left, video camera in front, and a 50/50 beamsplitter. For recording, the front side is covered with a black screen so as to capture only light entering from the left.* (b) *The image at the bottom shows a visualization of all DVS events registered during the exposure of the video frame above. Positive intensity changes contribute to a pixel's green channel, negative ones to the red channel. The race track exhibits change because it has been moved from the impact of the falling car.*

detects a change in intensity that exceeds a certain threshold. These spikes are forwarded to the sensor's output asynchronously and combined with a precise timestamp. Rather than fully integrated frames representing a complete view of the scene, the client receives a stream of events indicating local changes in pixel intensity. This encoding is similar to the way biological retinas transmit visual information to the brain, giving the sensor its nickname "silicon retina" [LPD08].

A software-controlled trigger starts the recording of data from both devices. In our experiments, we recorded both the full color video and the DVS event log to disk, performing any calculations as a post-processing operation. With the exception of the image registration step (section 4), which has to be re-performed only after the physical setup of the capture rig has been modified, the system could be extended to perform online processing.

4. Image registration and synchronization

In order to relate the events captured by the DVS to what is visible in the full color video, the two sensors have to be both spatially and temporally calibrated. Spatial calibration establishes a relation between moving scene objects visible in the camera images, and changes in contrast detected by the DVS. For this purpose, a mapping between the two coordinate systems has to be found. Furthermore, temporal synchronization enables us to assign DVS events to the video frames during the exposures of which they occured.

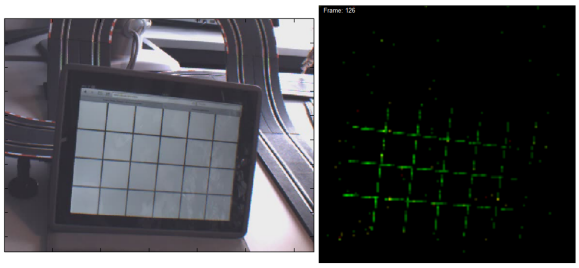


Figure 3: *The calibration pattern as captured by the video camera and the DVS. Events visualized in green are positive intensity changes, hence these events were generated as the black grid lines were switched back to white.*

4.1. Image registration

Because of the fundamentally different nature of the data recorded by the DVS, we cannot use traditional approaches at multi-camera calibration. While calibrating a video camera works best when using a static, sharply defined pattern, such a pattern is invisible to the DVS, which only registers parts of the scene that exhibit change. Introducing movement into the calibration pattern will make it visible to the DVS, but lead to a less precise calibration for the video camera due to motion blur accumulated over a frame's exposure.

We have approached this problem by displaying a calibration pattern on a screen or tablet computer, while flashing it on and off at regular intervals. The pattern is sharpely defined in any video frame that depicts it, as it doesn't move. At the same time, the DVS registers the full calibration pattern every time it is turned on or off on the screen. We render all events registered by the DVS during one camera frame exposure into an image, to get a full representation of the pattern in both datasets (fig. 3). In our experiments, we have used a tablet computer to display a flashing rectangular grid of black lines against a white background. It has to be noted that due to the DVS's low spatial resolution, the grid lines should have a certain thickness in order for them to trigger events at all.

Our semi-automatic image registration method involves finding the grid line intersection points in both the camera image and renderings of the DVS events. The user determines the areas depicting the calibration pattern. A standard line detection algorithm based on the Hough transform as described in [Hou59] is then used to locate the grid lines and find their intersection points. We pre-process the images with a Laplacian of Gaussian edge detector and detect the edges on either side of the grid lines, using their mean to find more accurately centered positions of the intersection points. The grid points in both images are then used to calculate a homography that maps coordinates from the DVS coordinate system to that of the video camera. The homog-

raphy is found using the MATLAB *Functions for Multiple View Geometry* by Hartley [HZ03] (http://www.robots.ox.ac.uk/~vgg/hzbook/code). The normalized DLT algorithm is used to arrive at an initial estimate, further refined using the non-linear MLE minimizing the Sampson estimate of the geometric error [HZ03, pp. 109, 114].

For a homography mapping to be precise, the principal points of the two optical systems must be aligned [AJN05]. In our setup we have only aligned the optical axes, and approximated the principal points manually. Due to the low resolution of the DVS when compared to the video camera, a DVS pixel will always map to an area several pixels in diameter in the video image. Thus, we have found a homography mapping to be sufficiently precise for our purposes.

4.2. Camera intrinsics

The video camera's intrinsic parameters and distortion have been determined using the *Camera Calibration Toolbox for* MATLAB by Jean-Yves Bouguet (http://www.vision.caltech.edu/bouguetj/calib_doc/). We use this calibration information to undistort the camera's images before processing them. The DVS's low spatial resolution presents a problem when trying to determine the intrinsic parameters of the optical system, as no precise enough calibration pattern can be made visible. However, we found that in our setup, the distortion of the optical system doesn't exceed one DVS pixel at any point, making an undistortion of DVS event coordinates unnecessary. It also has to be noted that due to the sparse, binary nature of contrast change events, as opposed to a camera's color information, the undistortion itself would pose a challenge, as interpolating discrete events would essentially introduce new, spurious events. Intrinsics calibration of a DVS system remains an open problem.

The nature of DVS data also poses a challenge when setting the focus of its lens, as discrete events cannot be out of focus. Depending on the contrast, they will either register or not. The flashing calibration grid can be used to support manual focussing, by choosing the distance of the display and the thickness of the pattern lines so that they are just wide enough to still register, as long as the image is in focus.

4.3. Temporal synchronization

For the purposes of temporal synchronization, the DVS128 supports injecting special events when a falling edge on an input pin is detected. We connect it to the output pin of the video camera, which is set to ground whenever the exposure of a new frame is started. These synchronization events are timestamped by the DVS to microsecond precision, and injected into the event stream. Thus, we can determine which events took place between the exposure of one video frame and the next, by finding all events which fall between two synchronization events.

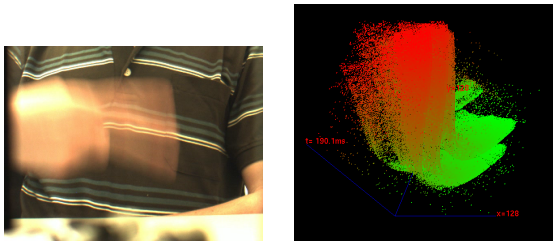
As the camera's image capturing process has to be started before the actual recording, there is no guarantee that the first exposure synchronization signal corresponds to the first video frame that is actually written to disk. This makes necessary an additional step to compute the offset between video frame numbers and synchronization events. We record a temporal synchronization event that is clearly visible on both sensors, such as a head slate being shut, at the beginning of each recording. The offset is determined manually, by rendering all DVS events between two synchronization signals to an image, and corresponding this visualization to the video frame depicting the same state of the synchronization event (such as the falling race car in fig. 2(b)).

5. Object tracking

In order to gain information about object motion from the DVS data, we make use of the fact that an object generates a dense trail of events along its edges and contrast features as it moves through the scene (fig. 4). By following these events, we can track an object from its initial position throughout the video, with very high temporal precision. Traditional tracking methods use appearance based models, that try to find regions of similar appearance over subsequent frames. One could apply these methods by simply creating artificial "exposure" times and binning events into frames. However, due to the low resolution, binary nature, and noise characteristics of the DVS data, appearance-based methods perform very poorly. Objects can undergo rapid changes, and the pattern of events they form in time is unpredictable, as it depends both on object appearance as well as background appearance. Therefore, we introduce a new method to track objects that takes advantage of the fact that DVS events are centered around regions of motion, and performs a geometry-based fitting of events in the spatiotemporal volume. Our method works in a few stages. We compute piecewise linear trajectory estimates for each object by fitting a 3D line in the spatio-temporal volume using RANSAC. We then fit a spline to the events that are inliers to this set of linear estimates.

We start by supplying the number of objects to be tracked along with a bounding box as an input to the algorithm. We then move a sliding temporal window along the time axis of the DVS event log, and fit a linear approximation of the motion for each object in this window. To do this, we use RANSAC to find the optimal pair of events such that the line connecting them has the most inliers. We define inliers to be events that are within the size of the user provided bounding-box from the line in question. We note that while the motion in each window is linear, we will only use the inliers from these trajectories, and as such do not require that the motion be exactly piecewise linear, only that events from the motion of the object exist within a region around this linear trajectory.

The temporal extent of the sliding window in content-dependent, and is based on the *number* of events, rather than



(a) (b)

Figure 4: *The 3-D plot* (b) *shows DVS events registered wile recording the scene depicted in the video frame* (a)*. While it is hard to tell how many fingers are extended from the video frame due to strong motion blur, the data in* (b) *clearly shows two extended fingers moving through the frame, generating contrast change events along their edges. (In this rendering, colors relate to the timestamp, not event polarity.)*

a fixed amount of time. In other words, at each step, the window is grown until τ events have been observed. This approach has two benefits. First, it is adaptive in that it accounts for the speed of the object; slow moving objects create fewer events and piecewise linear trajectory estimates will be computed over a longer amount of time, while fast moving objects will be modeled by piecewise linear trajectories that are much shorter in time, giving reliable estimates without excessive computation. Secondly, this approach naturally handles occlusions; a moving object stops generating events when it is occluded, and so the rolling window will grow past that portion of the video until it is visible again (fig. 5).

In a single temporal window, we start by computing a RANSAC trajectory estimation for each object. To handle multiple objects, points are iteratively removed until all objects are tracked. To do this, we estimate trajectories for each object *near* the end point of the track in the previous temporal window. We define valid start points as those within the spatial bounding box of the end of the trajectory in the previous temporal window, and then pick random end points until we find the line with the most inliers. Once estimated, we remove the events for the object with the most events in its trajectory, and then refit the remaining ones. This approach provides robustness when objects cross, as RANSAC will tend to fit towards the trajectory with the most points.

By using this iterative fitting approach, we are able to track objects that cross, as long as the trajectory is maintained in the crossed region, as shown in fig. 6.

As a final step, we fit a spline through all inliers of an object's determined motion path. This yields an accurate, smooth motion path (with arbitrary sub-frame temporal precision) for each object for the entire duration of the video segment. We use piecewise cubic splines, which we found to have sufficient accuracy and robustness.

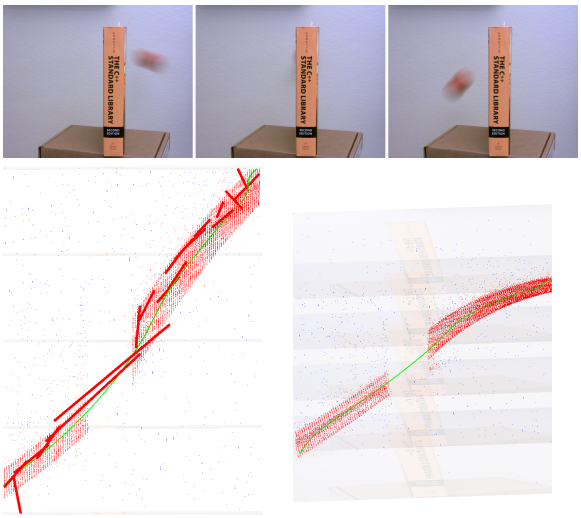


Figure 5: *For a recording of an object becoming temporarily occluded (top), the rolling window grows longer during the occlusion, until the object is visible again. This leads to a longer linear trajectory being estimated, and the occluded movement interpolated. In the plots at the bottom, the red line segments are the RANSAC fits of each rolling window, which determine the set of inliers (red points). The spline (green curve) is fit based on these control points. The transparent, horizontal planes represent frames of the video. The ball is only fully occluded in one frame.*

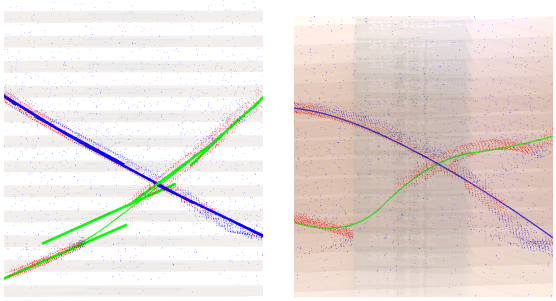


Figure 6: *Objects crossing paths are handled correctly, as each trajectory's inliers are removed from the point cloud before the trajectory of the next object is determined for the same temporal window. The inliers determined by the piecewise linear trajectories on the left act as support points for the spline fit, as shown in the plot on the right. Problems can arise if two objects generate a similarly dense trail of events, in which case their identities might be switched after the flip, as described in section 6.*

This approach has few parameters and produces high quality tracks. However, one important parameter is τ, the size of the window to use for linear trajectory estimates. Numerous factors influence the optimal choice for the threshold. A moving object triggers a certain amount of events based on its size, texture, and speed of motion. In general, this number should be such that the motion of the object can be roughly approximated by a line in the spacetime volume. In practice, we set this manually for each dataset, however the number can be set quite roughly. For all results shown here, we use τ values between 5000 and 15000.

6. Results and evaluation

We have applied our object tracking method to footage of several scenes that are particularly challenging for purely image-basesd techniques.

Figure 7 shows footage of an object (a DVD) being moved at very high speeds and in constantly changing directions, leading to very strong motion blur in the video frames. The heavy blur leads to feature-based tracking failing to detect the object and losing its track, as shown in fig. 8. The temporally dense information provided by the DVS during the exposure of just one video frame lets us reliably track the DVD's motion, even at very high speeds when the disc is only very faintly visible in the actual video. A large number of events generated leads to shorter extents of our rolling window, and robust estimation over shorter piecewise trajectories.

Figure 9 shows three identical rubber balls moving in different directions in front of the capture rig. Their high speeds cause them to cover large distances between subsequent frames (fig. 10). The fact that they have a very similar appearance presents an additional challenge to feature-based tracking algorithms, which have difficulties distinguishing between them. Our method is more easily able to tell them apart, as inlier events from the first object being tracked are removed before determining the trajectory of the second object. It has to be noted, however, that it is still possible for our algorithm to fail if objects of similar size and speed pass each other very closely, and their event point clouds cross. If the temporal window for a new piecewise fit happens to start at the point in time where the objects are very close, their identification can flip, as no previous trajectory information is considered. A higher value for τ can lower the probability of this problem occuring, but in turn limits our ability to track more complex, non-linear motion paths.

Such a failure case is depicted in fig. 11. One of the two LEDs occasionally moves in front of the other, or very close to it, causing their DVS event point clouds to coalesce. As events generated by one LED fall into the bounding box of the other at the start of the new rolling window, RANSAC can end up returning the end point of the wrong LED's trajectory. Hence, the object markers jump from one LED to

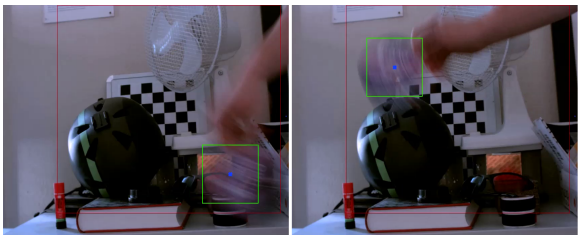


Figure 7: *Our method accurately tracks the position of the DVD even in the case of very strong motion blur in the video image. The red rectangle shows the overlap region of the two sensors, i.e. the area of the video frame covered by the DVS.*

Figure 8: *Motion blur is too severe in this scene for feature-based tracking to detect the object.*

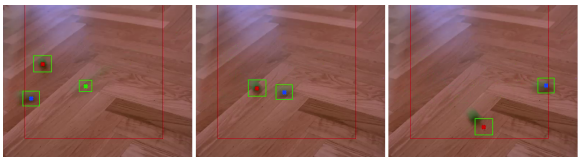


Figure 9: *Our method tracks the three moving rubber balls reliably, even the bouncing one (green marker) which is blurred so heavily as to be almost invisible in the video.*

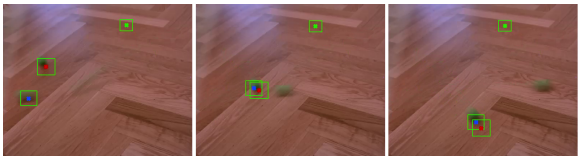


Figure 10: *The feature-based object tracker fails to detect the bouncing ball (green marker). The tracker also fails at distinguishing the two other balls, because of their similar appearance. From the moment they pass each other close-by, the two tracking positions stick together for the rest of the footage.*

the other in our result. As we don't include data from the color video in our algorithm, this can happen even for visually dissimilar objects. The repeated switching of object identities would not be the case though if the density of event trails generated by each LED were sufficiently different, such as due to one of them being considerably bigger or faster. RANSAC will always tend towards fitting the first object to the trajectory with the most points, before finding the second object's trajectory among the remaining outliers. The removal of the set of inliers before handling of the next object is also the reason why our object trackers will not stick together for objects crossing paths momentarily, as has happened for the feature-based tracker in fig. 10).

Figure 11: *When objects move past each other closely enough, the algorithm is susceptible to confusing them. Here, the green and blue markers switch hands multiple times.*

7. Future work

While our experimental setup has delivered promising results, there is potential in increasing the precision of calibration and registration. Alternate calibration patterns could help calibrate the DVS to sub-pixel precision, alleviating to some degree the problems of the large disparity in sensor resolutions. Calibration could also be simplified and improved by integrating both sensors in a single-lens system, or even on a single chip. Sensors that combine a DVS with APS (active pixel sensor) video recording functionality are currently in development.

Our method requires that the number of events in a temporal window be predefined. However, this is in essence a parameter of the expected motion of the scene, as well as the objects' texture, size, lighting conditions, and size of occlusions. One possible direction for future work would be to dynamically and automatically adapt the threshold based on information from the scene and currently tracked objects, improving reliability particularly in the case of multiple moving objects, objects entering and leaving the scene, or considerable changes in size. Related to this, approaches to determine a threshold not globally but per object, in the vicinity of their last known position, could make our method more robust in situations where only some of a scene's moving objects are occluded, while others continue to generate events. In our current algorithm, this can lead to relatively long rolling windows in cases where the remaining objects' movements would benefit from fitting shorter trajectory segments.

We assumed in our experiments that the capture rig is always stationary, and all motion is due to objects moving through the scene. This limitation could be alleviated by determining which events correspond to camera motion by fitting a plane through them in the space-time volume, removing them from consideration, and subsequently determining actual object motion from the remaining points. Robust handling of scenes with non-static backgrounds is an area open for future work.

Of particular interest is also the adaptation of our methods and algorithms to real-time applications, rather than post-processing. It is conceivable that the DVS point clouds could be processed on the fly, and splines fit in stages on a certain window of previously determined inliers. The asynchronous nature of the DVS output lends itself inherently well to low-latency online processing.

Another interesting direction is the incorporation of color information from the full video frames into the motion estimation process. We found that the discrepancy in sensor resolutions makes it difficult to glean useful information from the video frames, as the area each DVS pixel covers in a video frame is rather large. Additionally, in the case of objects moving at high speeds, motion blur makes even simple color estimates very inaccurate. However, the additional visual information could support object tracking in cases where objects are visually easily discernible and of sufficient size, while the amount of contrast change events they generate based on their speed and appearance is similar.

Finally, this work explored just one possible application of our combined DVS and video camera setup. The DVS's local contrast change events, registered with full, high-resolution color images from a video camera provides valuable additional information about the dynamic contents of a scene. This information could potentially find numerous interesting applications such as supporting traditional image processing tasks, or taking advantage of the DVS's sensitivity to pixel-local contrast change to enhance video shot in poor lighting conditions.

8. Conclusion

In this work, we have presented a multi-modal visual capture system consisting of a traditional video camera and a DVS, a sensor which asynchronously registers local contrast change. We have shown that the unique information provided by a DVS can support tasks that are hard or impossible to solve if the only input is fully exposed video frames. Our experimental application to the tracking of moving objects in video sequences has delivered reliable results in challenging situations where traditional approaches fail.

The fundamentally different nature of the two sensors creates its own set of challenges when the two datasets are

to be registered. Our setup and framework takes first steps towards a reliable calibration method, which allows to associate events generated by the DVS with specific areas in video frames.

More generally, we believe that this type of capture setup opens up various exciting paths of research into new scene analysis and video processing applications, while also showing potential towards improving traditional algorithms. We see our work as an initial exploration into the possibilities presented by combining video cameras with temporal contrast sensors, while discussing some of the inherent challenges that arise due to the unique nature of the data the latter generate. We therefore hope that this first work inspires vision researchers to apply these sensors in their own areas of interest, and to find new practices and applications.

References

[AJN05] AGARWAL A., JAWAHAR C. V., NARAYANAN P. J.: *A Survey of Planar Homography Estimation Techniques*. Tech. rep., Centre for Visual Information Technology, International Institute of Information Technology, Hyderabad, 2005. 4

[AMSB95] ANDREOU A., MEITZLER R., STROHBEHN K., BOAHEN K.: Analog vlsi neuromorphic image acquisition and pre-processing systems. *Neural Networks 8*, 7-8 (1995), 1323 – 1347. 2

[Bro92] BROWN L. G.: A survey of image registration techniques. *ACM Comput. Surv. 24*, 4 (Dec. 1992), 325–376. 2

[Car06] CARDINALE J.: Tracking objects and wing beat analysis methods of a fruit fly with the event-based silicon retina, 2006. Semester thesis, Universität Zürich. 2

[CSB*13] CENSI A., STRUBEL J., BRANDLI C., DELBRUCK T., SCARAMUZZA D.: Low-latency localization by active led markers tracking using a dynamic vision sensor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Tokyo,Japan, November 2013), pp. 891–898. 2

[Hof12] HOFSTETTER M.: *Temporal Pattern-Based Active Marker Identification and Tracking Using a Dynamic Vision Sensor*. Master's thesis, Universität Zürich / ETH Zürich, 2012. 2

[Hou59] HOUGH P. V. C.: Machine analysis of bubble chamber pictures. In *International Conference on High Energy Accelerators and Instrumentation* (CERN, 1959). 3

[HZ03] HARTLEY R., ZISSERMAN A.: *Multiple View Geometry in Computer Vision*, 2 ed. Cambridge University Press, New York, NY, USA, 2003. 4

[JKZS10] JOSHI N., KANG S. B., ZITNICK C. L., SZELISKI R.: Image deblurring using inertial measurement sensors. *ACM Trans. Graph. 29*, 4 (July 2010), 30:1–30:9. 2

[LPD08] LICHTSTEINER P., POSCH C., DELBRUCK T.: A 128×128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits 43*, 2 (Feb. 2008), 566–576. 1, 2, 3

[MMP*07] MCGUIRE M., MATUSIK W., PFISTER H., CHEN B., HUGHES J., NAYAR S.: Optical-splitting trees for high-precision monocular imaging. *IEEE Computer Graphics & Applications, Special Issue on Computational Photography 27* (March-April 2007 2007), 32–42. 2

[RHK*03] RUEDI P.-F., HEIM P., KAESS F., GRENET E., HEITGER F., BURGI P.-Y., GYGER S., NUSSBAUM P.: A 128×128 pixel 120 db dynamic range vision sensor chip for image contrast and orientation extraction. In *Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC. 2003 IEEE International* (2003), pp. 226 – 490 vol.1. 2

[Sze06] SZELISKI R.: Image alignment and stitching: a tutorial. *Found. Trends. Comput. Graph. Vis. 2*, 1 (Jan. 2006), 1–104. 2

[WDC*08] WANG O., DAVIS J., CHUANG E., RICKARD I., DE MESA K., DAVE C.: Video relighting using infrared illumination. *Computer Graphics Forum (Proceedings Eurographics 2008) 27*, 2 (Apr. 2008). 2

[YJS06] YILMAZ A., JAVED O., SHAH M.: Object tracking: A survey. *ACM Comput. Surv. 38*, 4 (Dec. 2006). 2

[ZA06] ZHOU Q., AGGARWAL J.: Object tracking in an outdoor environment using fusion of features and cameras. *Image and Vision Computing 24*, 11 (2006), 1244–1255. 2

[ZF03] ZITOVÁ B., FLUSSER J.: Image registration methods: a survey. *Image and Vision Computing 21* (2003), 977–1000. 2

[ZZY12] ZHANG K., ZHANG L., YANG M.-H.: Real-time compressive tracking. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part III* (Berlin, Heidelberg, 2012), ECCV'12, Springer-Verlag, pp. 864–877. 2