Explore the Unexplored
A Downward-Looking Dataset for Differential Wheeled Vehicles with Vision, Event, and Laser

A project of the 2019 Robotics Course of the School of Information Science and Technology of ShanghaiTech University
https://robotics.shanghaitech.edu.cn/teaching/robotics2019

Ling Gao and Kun Huang*

Abstract

We explore the feasibility of estimating the motion of a differential wheeled vehicle with a downward facing event-based camera alongside with a convenition camera that exerts fronto-parallel motion with respect to the ground plane. A basket of multi-channel datasets is recorded and could be used for further research.

1 Introduction and Motivation

Accurate velocity measurements are an essential ingredient to the stable localization and control of ground vehicles. A simple forward integration of vehicle speed can provide an estimate of the vehicle trajectory, a procedure commonly known as dead-reckoning. Although the forward integration will lead to a random walk, fusing its value in a complementary filter with an absolute reference signal such as GPS may already be sufficient to attain

*All the authors are with the School of School of Information Science and Technology, ShanghaiTech University. {gaoling, huangkun1}@shanghaitech.edu.cn.

reasonably accurate localisation. However, while the most straightforward solution to vehicle speed estimation is given by employing wheel odometers, the risk of varying and unknown wheel parameters as well as hard-to-model wheel slippage have ever since motivated the use of contact-less visual odometry to measure incremental vehicle displacements.

Our motivation is inspired by optical mouse sensors. The idea is that for any device that is being moved over a flat, planar surface, the velocity in the plane can simply be measured by an optical sensor that directly faces the plane. As the device is moving, the displacement information is deduced from the apparent motion of brightness patterns in the perceived image. This turns the motion estimation into a simple image registration problem in which we only have to identify a planar homography. Another important motivation for a downward facing sensor is that the depth and structure of the scene will be known in advance, and that—as a result—the motion of patterns in the image under planar displacement can be described by a simple Euclidean transformation. However, even though this estimation problem appears to be simple, it is difficult to find point correspondences between subsequent images as the perceived floor texture often does not lead to distinctive, easily matchable keypoints, especially suffers from the low-light enviroment and motion blur. As a compensation, event camera will survive because of its high dynamic range and low latency.

2 State of the Art

The image displacements for a camera moving in front of a planar scene can be modelled by a simple homography, which furthermore has a closed algebraic form as a function of the relative displacement, the camera intrinsics, and the plane parameters [13, 15]. A homography is an 8 DoF image-to-image transformation, and therefore generally requires 4 points to be solved. The complexity of the solution may be reduced by assumption of known plane parameters, or even constraints on the motion such as for example planar motion. For a downward motion-plane facing camera, these conditions would reduce the problem to a simple 3 DoF Euclidean transformation estimation. Other related work is given by [17], which substitute the Ackermann steering model into the essential matrix or an n-linearity. The imposition of epipolar incidence relationships that do no longer require either structure or scale parameters to be derived makes it possible for those methods to identify

the relative displacement based on a single feature correspondence.

The aforementioned solutions are interesting and related in that they exploit structure or motion related priors to solve the registration problem. However, they require feature correspondences for which at least a majority is consistent with a single dominant camera displacement. The dominant motion is then identified by sampling and testing hypotheses within a heuristic framework [11], which is why such methods cannot guarantee to find the optimal inlier set and an associated image-to-image transformation. Ransac-based methods contrast with globally optimal solutions that search the entire space of possible transformations to identify the optimal inlier set, possibly even without the prior requisite of point correspondences. The most common method here that has been regularly applied in geometric computer vision is given by branch-and-bound optimisation [14].

Another line of research that is related to ours is in globally optimal image matching for sparse [16, 9, 8] or even semantic, region-based [18] features. Such methods proceed by branching in the space of a low-dimensional image-to-image mapping, for example a 4 DoF transformation in the case of [9]. An alternative for Euclidean image registration that does not require the extraction of sparse features is given by the application of the Fourier Mellin transform [12, 10], which provides high computational efficiency but no guarantees of global optimality.

The above reference mainly justify the feasibility of this idea without mathematic proof, which we believe is off topic with respect to this project. Here, we also list some used ROS packages during our dataset recording and data processing.

ROS Package: OpenCV [2]

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

ROS Package: RPG DVS [1]

The ROS DVS package provides C++ drivers for the Dynamic Vision Sensors (DVS/DAVIS). Even if one do not have a DAVS or DAVIS device, one can still use this driver to read pre-recorded event data files. The package also

provides a calibration tool for both intrinsic and stereo calibration. This package is based on libcaer.

ROS Package: Camera Calibration Tool [3]

Camera calibration is the process of estimating intrinsic and/or extrinsic parameters. Intrinsic parameters deal with the camera's internal characteristics, such as, its focal length, skew, distortion, and image center. Extrinsic parameters describe its position and orientation in the world. Camera Calibration Tool allows easy calibration of monocular or stereo cameras using a checkerboard calibration target.

Other ROS Package for all kinds of sensor drivers

- Velodyne Lidar Driver [6]
- Point Grey Camera Driver [4]
- Husky Unmanned Ground Vehicle Driver [5]
- OptiTrack Motion Capture Systems Driver [7]

3 System Description on the Algorithm side

We present a pseudocode algorithm here to provide an overall look on the system description. Our overall algorithm denoted as Globally-Optimal event-based Visual Odometry (GOVO) is outlined in Algorithm 1 and aided by the helper function Algorithm 2 to calculate bound values within a certain interval.

Starting from an initial domain, the algorithm consecutively divides the 2-dimensional intervals into four subspaces, each time evaluating lower and upper bounds and pruning intervals if their upper bound is lower than the lower bound of another interval. We terminate the branching when the intervals lower bound equals the upper bound. Again, this pseudocode algorithm is another justification on the feasibility of the very idea. One has implemented the basic framework and tested its validity only on two consecutive frames, but no actually acceleration on computing time, or memory-saving tricks have been put into action, which we believe is off topic with respect to this project.

Algorithm 1 GOVO: efficient branch and bound algorithm for globally-optimal event-based visual odometry

INPUT: event set $\mathcal{E}$, initial domains V and W for estimated parameters v and w, intrinsic matrix $\mathbf{K}$, and distance between camera and ground d

OUTPUT: optimal solution $(v^\star, w^\star)$, data association among events

Initialize a space S_0 over the domains V and W
Execute Algorithm 2 for S_0
Push S_0 into queue Q
loop
    Pop a space S' from Q
    if $S' \rightarrow c^\star == true$ **then**
        Pop all spaces at same level in Q
        if converged **then** Collect correspondences
        Local refinement over all inlier correspondences
        return refined result $(v^\star, w^\star)$
    else
        Split S' into subspaces S_1, S_2, S_3, S_4
        Execute Algorithm 2 for S_1, S_2, S_3, S_4
        Prune and push into Q
    end if
end loop

Algorithm 2 bound calculation

INPUT: event set $\mathcal{E}$, domains V and W for estimated parameters v and w, intrinsic matrix $\mathbf{K}$, distance between camera and ground d

OUTPUT: convergence signal $c^\star$, lower bound value P_{lb}, upper bound value P_{ub}, data association among events

Calculate lower bound by evaluating the objective at the interval centre
for each event e in $\mathcal{E}$ **do**
    Wrap e back to the image grid on a reference frame
end for
Calculate the covariance of the image as upper bound
if $P_{lb} == P_{ub}$ **then** $c^\star = true$ **else** $c^\star = false$

Figure 1: The husky robot in a garage, with downward-facing sensors, a bumper and assisted lights in the front, Lidar on the top.

4 System Description and Evaluation on the Dataset side

Data acquisition is an essential step among all procedures when conducting a scentific experiment. As suggested and guided, to suit different scenarios, we mount several sensors on the differential wheeled robot, including Velodyne Lidar, the conventional RGB camera with high resolution, dynamic vision sensor integrated with event camera, IMU (Inertial Measurement Unit) and a low resolution monochromatic camera, also aiding by OptiTrack System for high precision ground truth. A photograph of the robot is captured and presented in Figure 1.

Here, we list a basket of recorded datasets under different scenarios with different combinations of sensors in Table 4, and their textures are displayed in Figure 2.

Scenario & Texture	Type of Motion	RGB Camera	DVS	Lidar	OptiTrack System	Duration Time
Lab Floor	Linear	✓	✓	✗	✓	5.4s
Lab Floor	Linear	✓	✓	✗	✓	12.5s
Lab Floor	Rotation	✓	✓	✗	✓	9.9s
Lab Floor	Rotation	✓	✓	✗	✓	11.8s
Lab Floor	Random	✓	✓	✗	✓	14.2s
Lab Floor	Random	✓	✓	✗	✓	41.5s
Lab Carpet	Linear	✓	✓	✗	✓	7.5s
Lab Carpet	Linear	✓	✓	✗	✓	14.5s
Stone-paved	Linear	✓	✓	✓	✗	5.4s
Stone-paved	Linear	✓	✓	✓	✗	8.5s
Stone-paved	Rotation	✓	✓	✓	✗	9.4s
Stone-paved	Rotation	✓	✓	✓	✗	13.4s
Stone-paved	Random	✓	✓	✓	✗	25.3s
Stone-paved	Random	✓	✓	✓	✗	30.2s
Garage	Linear	✓	✓	✓	✗	7.0s
Garage	Linear	✓	✓	✓	✗	20.1s
Garage	Rotation	✓	✓	✓	✗	8.7s
Garage	Rotation	✓	✓	✓	✗	10.1s
Crossing	Linear	✓	✓	✓	✗	5.4s
Crossing	Linear	✓	✓	✓	✗	15.7s
Crossing	Rotation	✓	✓	✓	✗	8.2s
Crossing	Rotation	✓	✓	✓	✗	12.3s

Table 1: A summary on datasets, with texture, type of motion, sensors, and duration time.

Figure 2: Textures are listed in the sequence of (1) Lab Floor (2) Lab Carpet (3) Stone-paved (4) Garage (5) Crossing (from left to right). Please be aware that the actual field of view and actual height between camera and ground in the datasets are much smaller and lower than the ones in display.

Dataset Format

For each data stream, we present in text/png files and binary files (rosbag). While their content is identical, some of them are better suited for particular applications. For prototyping, inspection, and testing, it is recommended to use the text files, since they can be loaded easily using Python or Matlab. The binary rosbag files are prepared for applications that are intended to be executed on a real system.

Here we list the detailed infomation in text/png format as follows.

- *calib.txt* Intrinsic parameters (f_x f_y c_x c_y k_1 k_2 p_1 p_2 k_3).
- *pose.txt* One measurement per line (timestamp p_x p_y p_z q_x q_y q_z q_w).
- *dvs/events.txt* One event per line (timestamp x y polarity).
- *dvs/imu.txt* One measurement per line (timestamp a_x a_y a_z g_x g_y g_z).
- *dvs/image_raw/*.png* Grey images with resolution of $346 * 260$.
- *camera/image_raw/*.png* Grey images with resolution of $2448 * 2048$.

Here we list the detailed infomation in binary format as follows.

- *camera–calibration* contains intrinsic parameters.
- *vrpn* contains pose information of the vehicle as ground truth.
- *velodyne* contains pose information of the vehicle as ground truth.
- *rpg_dvs_ros* contains event streams, IMU streams, and grey images.
- *pointgrey* contains grey images with high resolution.

Notes on Data Acquisition

1. There is an unexpected incompatibility between OptiTrack System and Dynamic Vision Sensor, where OptiTrack uses high-power infrared spotlights at a high frequency. It is invisible and easily-negelected to human beings or conventional sensors, but it has a spectacular interference on event cameras because of its sensitivity on high dynamic

range. This will overflow the buffer with useless noises. We deactivate the LED option on the control board in OptiTrack software to a minimal level as a tradeoff between low noise and accurate ground truth detection.

2. Both the conventional RGB camera and dynamic vision sensor are mounted with the sam type of the lenses, which has a small field of view but high distortion. They were equipped on the same height and pointed to the ground vertically. One needs to do the camera calibration twice, not only because of the accuracy, but also with the fundamental difference on camera resolution.

3. For event stream, fine-tuning on some parameters is neccessary to gain a clean result. Among all parameters one can play with, *background activity filter time* seems to be the most important one. It will reduce the noises generated by strobing LED from OptiTrack, and provide a relatively clean result.

4. Light is also essential in our experiment. For conventional RGB camera, sufficient illumination gurantees a shorter exposure time and a slighter motion blur effects. For Dynamic Vision Sensor, since events are triggered by the change of log intensity of pixels, a brighter enviroment will produces less false-alarm changes (noises) than darker environment.

5. Originally, we plan to mount one more event camera on the robot as a comparsion. The home-made CelePixel Dynamic Vision Sensor has a much higher resolution (roughly 1 million pixels), resulting in an unfortunately situation that skyrocketing number of events blocked the buffer and occupied the processing units. This phenomenon will block other sensors and sometimes even worse, crashing the whole system.

Evaluation and Reproducibility

As for evaluation on the quality of dataset, here we purpose a simple measurement to test the blurriness of the captured images both from conventional RGB camera and Dynamic Vision Sensor, as a comparison to event streams. We believe the ratio of the number of detected features in a blurred image to one in the still image can be seen as a compelling criterion, since the quality and quantity of features are a crucial step in any SLAM algorithm.

Here, we list the evaluation result for all recorded datasets in Table 2. Also, as mentioned above, the promising result from our testing codes based on the collected dataset proves that the event stream is reliable and robust.

To reproduce the very same dataset as ours with identical trajectory seems impossible, but with the 3D-printing holder to contain and fix the cameras with same height, the already installed driver on the Raspberry Pi, one can easily reproduce a similar dataset with ease. We write a README file to guide anyone who want to conduct the data acquisition again.

5 Conclusions

Given the small depth-of-scene, it is necessary to process images at a very high frame rate if elevated vehicle speeds are to be estimated. Therefore, our efforts consist of improving computational efficiency, accuracy, robustness and employing event-based cameras to obtain very low-latency perception, and unlocking the unexplored potential towards accurate visual odometry in dynamic scenarios in conjunction with our globally optimal registration approach. The dataset we collected will be a solid foundation to conduct further research on visual odometry for differential wheeled vehicles.

References

[1] `https://github.com/uzh-rpg/rpg_dvs_ros`.

[2] `https://opencv.org`.

[3] `http://wiki.ros.org/camera_calibration`.

[4] `http://wiki.ros.org/pointgrey_camera_driver`.

[5] `http://wiki.ros.org/Robots/Husky`.

[6] `http://wiki.ros.org/velodyne`.

[7] `http://wiki.ros.org/vrpn_client_ros`.

[8] Jean-Charles Bazin, Hongdong Li, In So Kweon, Cédric Demonceaux, Pascal Vasseur, and Katsushi Ikeuchi. A branch-and-bound approach to correspondence and grouping problems. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1565–1576, 2012.

Scenario Texture	Type of Motion	Duration Time	# of features (still image)	# of features (blurred image)	ratio
Lab Floor	Linear	5.4s	28	2	7.1%
Lab Floor	Linear	12.5s	19	2	10.5%
Lab Floor	Rotation	9.9s	19	1	5.2%
Lab Floor	Rotation	11.8s	14	3	21.4%
Lab Floor	Random	14.2s	12	1	8.3%
Lab Floor	Random	41.5s	19	2	10.5%
Lab Carpet	Linear	7.5s	44	0	0.0%
Lab Carpet	Linear	14.5s	52	5	9.6%
Stone-paved	Linear	5.4s	5107	49	1.0%
Stone-paved	Linear	8.5s	709	84	11.8%
Stone-paved	Rotation	9.4s	213	41	19.2%
Stone-paved	Rotation	13.4s	5219	46	0.9%
Stone-paved	Random	25.3s	4174	45	1.0%
Stone-paved	Random	30.2s	4417	83	1.9%
Garage	Linear	7.0s	184	13	7.1%
Garage	Linear	20.1s	121	18	14.9%
Garage	Rotation	8.7s	198	2	1.0%
Garage	Rotation	10.1s	181	7	3.9%
Crossing	Linear	5.4s	7	4	57.1%
Crossing	Linear	15.7s	9	1	11.1%
Crossing	Rotation	8.2s	9	2	22.2%
Crossing	Rotation	12.3s	8	2	25.0%

Table 2: An evaluation on datasets, with number of features on both still and blurred images.

[9] Thomas M Breuel. Implementation techniques for geometric branch-and-bound matching methods. *Computer Vision and Image Understanding*, 90(3):258–294, 2003.

[10] H. Bülow and A. Birk. Fast and robust photomapping with an unmanned aerial vehicle (uav). In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[11] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[12] X. Guo, Z. Xu, Y. Lu, and Y. Pang. An application of fourier-mellin transform in image registration. In *The Fifth International Conference on Computer and Information Technology*, 2005.

[13] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[14] Ailsa H Land and Alison G Doig. An automatic method for solving discrete programming problems. In *50 Years of Integer Programming 1958-2008*, pages 105–132. Springer, 2010.

[15] Yi Ma, Stefano Soatto, Jana Kosecka, and S Shankar Sastry. *An invitation to 3-d vision: from images to geometric models*, volume 26. Springer Science & Business Media, 2012.

[16] David M Mount, Nathan S Netanyahu, and Jacqueline Le Moigne. Efficient algorithms for robust feature matching. *Pattern recognition*, 32(1):17–38, 1999.

[17] Davide Scaramuzza, Friedrich Fraundorfer, and Roland Siegwart. Real-time monocular visual odometry for on-road vehicles with 1-point ransac. In *2009 IEEE International Conference on Robotics and Automation*, pages 4293–4299. IEEE, 2009.

[18] Pablo Speciale, Danda P Paudel, Martin R Oswald, Hayko Riemenschneider, Luc Van Gool, and Marc Pollefeys. Consensus maximization for semantic region correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7317–7326, 2018.