Semesterthesis – Final Presentation

Welcome everyone to the Final Presentation of my Semesterthesis about Visual-Inertial Odometry. I will show up what is out there and demonstrate where we are towards a generic Visual-Inertial Localization. Which means a Localization which is efficient, accurate and robust.

-

In the future a Task like windmill inspection and maintenance will be done by autonomous mobile robots. One key challenge towards this goal is Single Robot Localization.

The localization needs to be accurate in a global sense: the robot has to fly to the windmill, and in a local sense for example for the inspection itself.

Visual-Inertial Odometry is one framework takling Localization. It combines the rich structure information of a camera with the short-time accuracy of an Inertial Measurement Unit.

Main Advantages against other Localization Approaches are the reliance on lightweight sensors. The use of passive sensors with a small power consumption; and the reliance on solely onboard sensors averting the need of an external localization system.

-

I would like to present 2 main topics today. In the first part I will compare two Visual-Inertial Odometry Implementations by demonstrating their working principles, showing differences and demonstrating their accuracy.

In the second part I will takle the question if Visual-Inertial Odometries can work with non-timesynchronized Hardware.

In the end I will give a short outlook into future work.

-

Visual-Inertial Odometry is part of the larger field of Visual Odometry. On top you can see four main work of this field. The Visual-Inertial Odometries can be divided into 2 principally different approaches – Filtering-based and Keyframe-based.

The Filtering-based Approach estimates the robot pose in 2 steps: Firstly the robot state is propagated between frames based on the IMU measurements. Secondly, as soon as a new image is available, the estimate is updated based on the observation.

**The information of a past observation is transferred into the current one via the prior belief of the filter**.

In the filtering approach a past estimate will never be corrected based on the current observation.

On the other hand the keyframe-based approach estimates the robot pose by performing a nonlinear optimization based on landmark observations taken from a set of past keyframes and the current frame. Like that poses of the past are corrected based on the current observation.

Because of this inherent difference the Filtering-based approach normally shows a lower accuracy as it more prone to drift.

On the other hand the keyframe-based approach is in general more computational expensive.

For this work I focused on ROVIO and OKVIS, two promising Implementation of the two Approachs. Both have been developed here at ETH and are on the way of being published open-source.

-

Rovio stands for Robust Visual-Inertial Odometry and it is using a direct Extended Kalman Filter Approach. What you can see on the slide is a Visualization of Rovio during operation. On the left side we see the current image Rovio is working with.

Rovio detects new features with a fast corner detector and is working with Multilevel Patch Features. It is working with a small number of features – at the moment around 20 – which are reprojected into the image.

On the right side we see a Visualization of Rovios estimates – In the center we see an inertial coordinate frame and, in small, the estimated body frame. The white dots represent the filters belief of the Features in space and the green line denote the 2 sigma bound of the feature uncertainty.

At the initialization all features are initialized with a high uncertainty

PLAY

As we can see, as soon as the camera is moved the uncertainty associated with the features is decreased.

-

Okvis stands for Optimal Keyframe-based Visual-Inertial SLAM. In the Visualization we can see on top the latest keyframe and on the bottom the current frame.

Okvis detects new features with a multiscale Harris-Corner detector and describes them with a Brisk descriptor. In the image, the extracted features are shown as circles, the green lines highlight matching pairs.

Okvis is estimating the robot pose by performing a nonlinear optimization over a few hundreds to thousand landmarks, observed from a few keyframes and the current frame.

The right side of the visualization shows the landmarks.

PLAY

One remark regarding Okvis: We have not been able to run it on our machines until today. The OKVIS results I will present, have been generated by Stefan Leutenegger and Simon Lynen.

DURCHATMEN

I want to compare the two presented algorithms on a dataset Simon Lynen captured for evaluating Okvis.

The data was collected with the VI-Sensor, which is equipped with two global-shutter monochrome cameras with 120-degree wide lenses, and a high-quality MEMS-IMU.

Both algorithms are running on a single camera and the IMU.

The data contains a long trajectory captured indoors with Vicon ground truth. For this purpose approximately 100 handheld loops in the room have been captured.

The 2D overview shows the ground truth trajectory.

Start and Endpoint are lying on the down left corner, for the whole sequence, the camera was facing in direction of movement.

Here you can see a visual impression of ROVIO’s tracking performance. It is not clear out of this graph, but what happened was, that in the beginning it was tracking the circle quite well over time it drifted and in the end it described this circle down here.

If we look at the overview plot of Okvis we see that it is also drifting, but it stays nearer to the ground truth.

For all the upcoming slides I will show ROVIO in RED and OKVIS in blue.

-

In the following I want to differentiate between Global and Local Accuracy.

The most common metric to describe the Global Accuracy is the absolute translation error. It describes simply the norm between the ground truth and estimated position after a certain travelled distance.

I like to show errors with boxplots. You take the data over a certain interval of travelled distance and show the error characteristics. The dot denotes the median, the fat box the 25 and 75 percentils and the thin line the interval with values without outliers.

After 1000 meter we can see that Okvis has a median absolute translation error of below 1 meter, while Rovio shows a median absolute translation error of 2.5 meter.

If we look at the absolute orientation error we see that ROVIO shows a ten times higher error with 45 degrees compared to 4 degree for OKVIS.

If you look at the orientation error at the axes separately, the roll and pitch errors are bounded and below 1 degree for both algorithms over the whole sequence. Thanks to the IMU the gravity vector is observable and therefore errors around axes lying in the horizontal plane are bounded.

The major contribution to the absolute orientation error is coming from the yaw error – the rotation around the world-z axis.

The most important Result regarding Global Accuracy is therefore that, for this dataset, Rovio shows a yaw drift of 45 degrees over 1000 meter, which is 10 times than for OKVIS.

-

In addition to the Global Accuracy it is of great interest how well tracking is working on short distances. The relative translation error compares the true and the estimated Travelled Distance between two frames.

Here again I am showing the relative errors with boxplots. The “Bubbles” in the plot are outliers.

We can see that the relative translation error is a bit higher for Rovio, but in the same order of magnitude for both algorithms. For both algorithms we see an error of approximately 2 cm after 1 meter travelled distance.

The second metric to describe the local accuracy is the relative orientation error. It compares the true change in attitude with the estimated change of attitude. Here we see similar performance for both algorithms. Both algorithms show a median error of 0.6 degrees after 1 meter of travelled distance.

-

Let us summarize the results on this dataset.

1) Regarding Global Accuracy we see a clear gap between Rovio and Okvis.

2) Locally we saw that both algorithms show similar results.

Regarding Computational Complexity Rovio is quite lightweight with a processing time per frame of 10 ms on my laptop with settings I used to generate the results. For Okvis we do not have a value to compare but we know that it is computationally more demanding but able to run in real time.

Beside of these results there is an additional “soft” argument for Rovio: It is working robustly for many people in here – live with the visensor, in simulation and on other different datasets I was working on.

DURCHATMEN

I would now like to go over to the second topic, where I want to show you an analysis I have done on the way towards non-timesynchronized hardware. The results I presented so far have been based on data collected with the VI Sensor. The Vi Sensor is hardwarewise time- synchronized which means that the IMU measurements are triggering the camera image capture. Like this you get your IMU and camera information really timesynchronized into your algorithm.

An important question when we want to apply Visual-Inertial Odometry generically is, if an algorithm is also able to work with a sensor setup that does not perform such a hardware time-synchronization.

-

As a first step towards this question I did an analysis, where I took data collected with the VI-Sensor and added an artificial bias on the IMU-Timestamp.

You can see in the plot the Translation and Orientation Errors averaged over the whole timesequence as a function of the artificially added IMU timestamp bias. What we observed with this analysis was, that, for the given data, Rovio is able to do quite a good job until a bias of 50 ms, and eventually starts diverging at 100 ms.

A second result out of this analysis was, that we can get satisfying results at even higher Biases by fixing the Camera-IMU extrinsics. This means that normally Rovio is constantly estimating the camera-IMU extrinsics over time. If we initialize Rovio with a good set of extrinsics and reduce this degree of freedom in the filter state, higher artificial timestamp biases can also be tracked.

One remark: This plot is highly dependent on the dynamics in the dataset. For this analysis I used data with slow motions.

-

This analysis encouraged us to collect a real dataset to compare Rovio once working with the VI sensor data and once with a non time-synchronized setup.

I attached a Bluefox camera with the same characteristics as the VI-Sensor camera to the VI-Sensor, making sure that they capture the same field of view.

With that I collected two datasets within the Vicon room and analysed Rovio performing once with the VI sensor data and once with the data from the Bluefox camera and the IMU data from the VI-Sensor. For both sets I collected data over a distance of 60 meters which was 3 loops in the vicon room.

-

Here you can see a short impression on the “slow dataset”

PLAY

I collected it in the LEO vicon room, where a lot of landmarks are in the field of view. The camera stream of the VI sensor camera and of the Bluefox camera look very similar. Running now Rovio once with the visensor data and once with the non-timesynchronized setup got the following result.

The most important result of the analysis is, that the non-timesynchronized setup does not break Rovio and it still does a good job in estimation.

The second and not too surprising result was, that the tracking performance of Rovio working with the VI-Sensor was more accurate.

-

In the following I tried to go towards the dynamic limits and collected the following shaky dataset.

PLAY

Running now Rovio on the VI sensor data still resulted in non-diverging tracking, while Rovio running on the non-timesynchronized data was diverging. We reached a limit for Rovio working with non-timesynchronized hardware.

The last step I tried now was to fix the camera-imu extrinsics. By reducing this degree of freedom in the filter state Rovio is not diverging anymore. It is an important result that fixing the extrinsics can improve performance when applying Rovio on non-timesynchronized hardware.

-

DURCHATMEN

Summerizing the presented work we saw, that

1. Rovio and Okvis are both promising Visual-Inertial Odometry Implementations
2. Rovio is prone to drift – especially in Yaw
3. Rovio and Okvis showed similar and good performance regarding local accuracy

and

1. Rovio is able to run on a non-hardware-timesynchronized Sensorsystem

This brings me to the outlook after this Semesterthesis.

To tackle the global drift of Rovio it has to be combined with a “Backend”, which is a secondary module taking the output of Rovio and performing global optimization. This backend could than feedback its results to Rovio like a GPS input.

As a next step Rovio will be run on an embedded system with lower-quality IMU and non-timesynchronized hardware.

And as a third point, the presented results encourage even more to run Okvis and evaluate it’s performance on different datasets and on the data I collected for my analysis towards non-timesynchronized Hardware.

-

Coming back to the Task of Windmill inspection and maintenance I would like to highlight again, that both Visual-Inertial Odometries showed very promising results. I strongly belief, that Visual-Inertial Odometry is one key towards efficient, accurate and robust Localization.