2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Stereo Vision and IMU based Real-Time Ego-Motion and Depth Image Computation on a Handheld Device

Korbinian Schmid and Heiko Hirschmüller
Department of Perception and Cognition
Robotics and Mechatronics Center
German Aerospace Center (DLR)
Oberpfaffenhofen, Germany

Abstract— **Real-time environmental depth perception and ego-motion estimation is essential for all mobile robotic systems. We present a system that computes high quality depth images with 0.5 MPixel resolution using Semi-Global Matching (SGM) and estimates the ego-motion by key frame based visual odometry fused with the data of an inertial measurement unit (IMU). The hardware includes a pair of cameras, a small Intel Core2Duo CPU board, a Spartan 6 FPGA board, an OMAP3530 ARM processor board as well as an IMU. The total weight of the experimental setup is 830 g and is, thus, also feasible for hand-held or flying platforms. Experiments show that the vision system runs at 14.6 Hz with a latency of around 250 ms and produces high quality depth images as well as reliable 6D ego-motion estimates. In the fusion algorithm of visual odometry and IMU data, time delays of the vision system are compensated and a system state estimate is available at the full data rate of the IMU which is important for system control. This paper presents the integration of different techniques into a fast, light weight, real-time system and validates its performance by experiments on real data.**

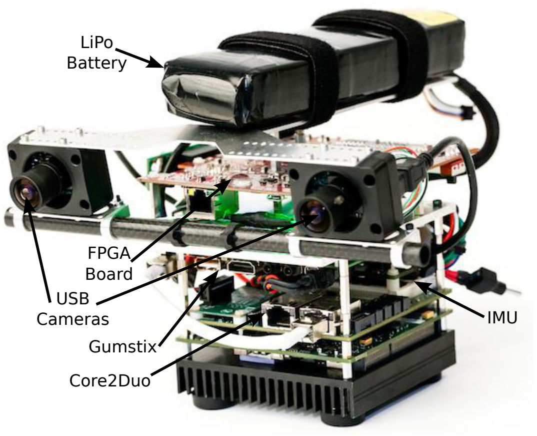


Fig. 1. Mobile robot perception device

I. Introduction

Perception of the environment is essential on all system levels for autonomously navigating robots. On the lower level an accurate estimate of the current system state (i.e. pose, velocity and sensor biases) is required for control. On a higher level, surrounding obstacles have to be recognized to avoid collisions. But also high level navigation and interaction tasks need to be aware of the robot's surrounding.

On autonomous mobile robots all sensor data should be processed directly onboard the system as communication to a base station can in general not be guaranteed. Furthermore, mobile robots have strong limitations in power consumption and payload and, therefore, limited computational resources. Thinking of micro robots or hand-held devices, the onboard processing of high amounts of exteroceptive sensor data is well known to be a challenging task.

Considering the mentioned restrictions, a suitable sensor equipment has to be chosen. Cameras have suitable characteristics for mobile robots as they are light weight and passive. Using a stereo system the 6D-motion of the robot which is the basis for navigation can be determined. Additionally, a dense stereo (depth) image can be used for obstacle avoidance and map creation. Compared to active depth sensors like Kinect, stereo systems are not limited to indoor environments, due to its passivity and the possible adaptation to different lighting conditions.

However, cameras produce a large amount of data that has to be processed in real-time. A trade-off between image resolution and data rate has to be found. A high resolution improves the depth information which is the basis for several other tasks like obstacle detection, whereas a high frame rate allows for a better motion estimation especially on highly dynamic systems like Micro Air Vehicles (MAVs). We also need to consider that data processing and transmission introduces time delays which can be critical if the calculated pose is directly used for control of agile systems. Nevertheless, relying only on an exteroceptive sensor like a camera, we could not guarantee a fail-free motion estimation. Visual pose estimation can fail suddenly, due to bad lighting and texture conditions of the scene or fast movements that cause image blur. Supporting the camera measurements by a proprioceptive sensor like an IMU should close the gaps where the image-based motion estimation fails and allow also for an increased update rate of the motion estimation.

By such a combination we get a perception unit for mobile robots that can be used for system control, obstacle avoidance and 3D environment modeling. We use an FPGA implementation of the Semi-Global Matching (SGM) stereo algorithm [1] for calculating depth images in real-time. Based on the

3D structure of the tracked image features we determine the 3D pose of the robot which is fused with the inertial measurements of an IMU. We obtain a system state estimate at a high rate, compensated for time delays and possible short-term failures in the visual odometry measurements.

We will give an overview of related work to be followed by a system description including details to the hardware, image processing and state estimation. Thereafter, we analyze the performance of our system considering runtime and accuracy with and without fusion. Finally, we discuss results and future work.

II. Related Work

Feature based visual odmetry [2] and simultaneous localisation and mapping (SLAM) [3] have a long history. Modern approaches aim at matching all pixels, which require graphic card implementations due to the high computational load [4] and partly rely on active sensors [5], [6] that do not work outdoors due to sunlight.

For mobile and especially flying systems, only low power CPUs and FPGAs are available for processing. Honegger et al. presented an FPGA implementation of correlation based stereo matching and optical flow computation on images of 376×240 pixels that runs at 127 Hz [7]. The system aims at highly dynamic flying platforms, which is the reason for the high frame rate. Goldberg and Matthies show an implementation of stereo matching and visual odometry, supported by an inertial measurement unit (IMU) on an OMAP3530 processor [8] for use on small robots. They report a processing speed of 46 Hz at a resolution of 320×240 pixels with 32 pixels disparity range.

In one of our earlier publications [9] we discussed that a more advanced stereo algorithm like Semi-Global Matching (SGM) [1] delivers denser depth images with higher structural details as correlation based stereo methods, which is advantageous for navigation. For the same reasons, SGM is also used for driver assistance applications [10] in the 6D Vision system of Daimler[1]. Our approach for visual odometry [2], [9] is not based on local tracking and can, therefore, handle large motions at low frame rates. The implementation for a crawling robot performed 4-5 Hz frame rate at a resolution of 640×480 pixels with 128 pixels disparity range on a desktop computer with SGM running on a graphics card [11].

In this paper, we close the gap between systems that can process on small hardware with high frame rate, but with low image resolution and using low quality methods like correlation, and systems that process higher resolution images with advanced algorithms, but require much higher processing power which is not available on mobile systems.

In [12] we analyzed the influence of sensor time delay and framerate on the quality of ego-motion estimation. Considering the results, we propose a combination of specialized processing units, including a small but powerful CPU and an FPGA for processing images by SGM with up to 1024×508 pixels with 128 pixels disparity range and performing visual odometry at 14.6 Hz, which is fused with the data of an IMU on a resource efficient RISC processor (ARM).

[1]www.6d-vision.com

The main advantage of our system is the robustness and accuracy of ego-motion calculation in combination with sensor time delay compensation. The system produces stable estimation results even in the case of complete visual odometry drop offs for several seconds with interruptions in the connectivity of visual features which we demonstrate in several experiments. Using a key frame approach the system is locally drift free. The calculated high-resolution depth images can be used for collision avoidance and 3D mapping. All compution tasks are realized in real-time on a light-weight handheld device. The device can dirctly be used for control of highly dynamic mobile systems as flying robots.

III. The System

The proposed system consists of compact hardware and fine tuned software components that we will describe in the following sections.

A. Hardware

Figure 1 shows our mobile robot perception device. It is equipped with a pair of Point Grey FireFly cameras with a resolution of 752×480 pixels. The cameras are connected to an Intel Core2Duo L9400@1.86GHz processor board via USB. The CPU board uses Linux as operating system. A Spartan 6 LX75 FPGA Eval Board is connected via PCI express.

The system also includes an ADIS16407 Inertial Measurement Unit (IMU) integrating a triaxial digital accelerometer, gyroscope and magnetometer as well as a barometer. The IMU is connected via SPI to an OMAP3530 (ARM Cortex-A8, 720MHz) Gumstix Overo processor board running a Linux with real-time extension. The camera hardware sync signal is connected to the Gumstix to register the exact hardware time stamp of captured images.

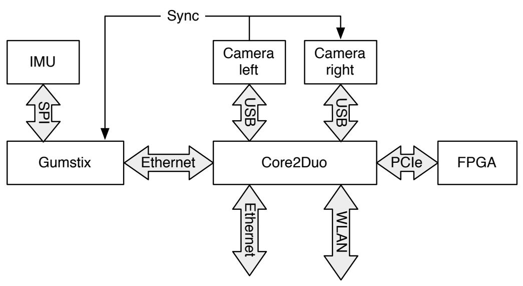


Fig. 2. Block diagram of the mobile robot perception device.

The ARM and Core2Duo processor boards communicate via Ethernet. For time stamp exchange, the systems are synchronized via the Precision Time Protocol V2. The X86 board is configured as Ethernet bridge and equipped with a second Ethernet port and a n-Wireless Lan adapter for external communication. Figure 2 depicts the hardware block diagram of the system.

Excluding camera mount and carry handle, the mobile robot perception device has a size of 12 cm x 11 cm x 11 cm (length x width x height). The weights of the individual hardware components are listed in Table I. The integrated system in its current configuration has a total weight of 830g which can be carried by a wide range of small-size mobile ground and flying robots.

TABLE I
WEIGHT OF SYSTEM COMPONENTS

Component	Weight
FPGA board	95 g
Core2Duo processor board	85 g
Core2Duo base board	89 g
Core2Duo cooler	178 g
Gumstix processor board	5 g
Gumstix base board	29 g
IMU	16 g
Cameras	2 x 36 g
Power supply	98 g
Mount and cables	163 g
Total	830 g

B. Image Processing

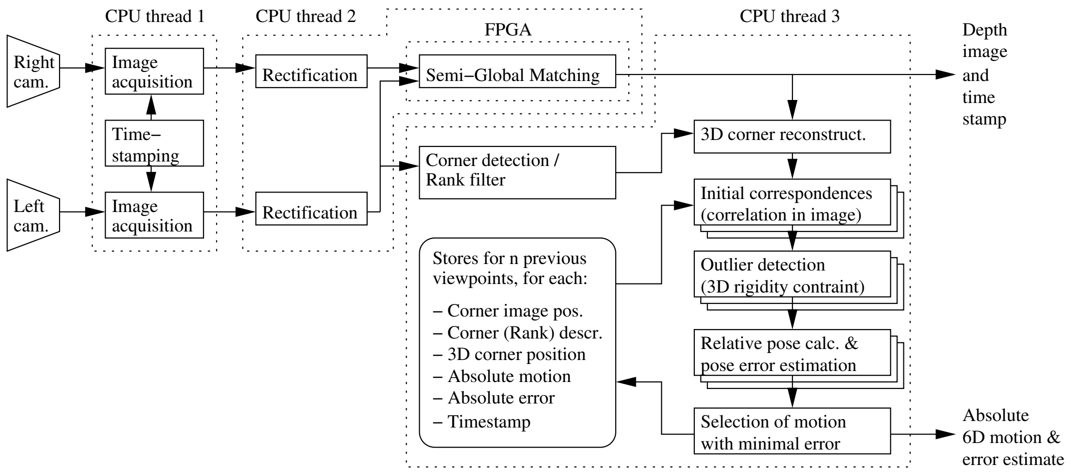
The image processing software is divided into three separated tasks as shown in Figure 3. The first thread is responsible for acquiring synchronized images from the cameras. Synchronization is ensured by connecting the hardware trigger input of the right camera to an output of the left camera that signals the frame integration time. For getting a reliable image timestamp from USB cameras, we are sending a software trigger to the left camera and storing the system time for the next image pair to arrive. Unfortunately, the used cameras cannot use auto-exposure when triggered externally. Therefore, the thread also controls the exposure time and gain of both cameras,

The second thread takes the most recently captured image pair from the first thread, performs planar rectification and transfers the images via PCI express to the Spartan 6 FPGA for Semi-Global Matching (SGM). The SGM algorithm performs pixelwise matching that is supported by a global cost function that prefers piecewise smooth surfaces in the disparity image [1]. The algorithm combines the accuracy of pixelwise matching with the efficiency of local stereo methods. Census [13] is used as matching cost, due to its high radiometric robustness that is required for real world applications [14]. The Spartan 6 implementation that we use, is an optimized version of Gehrig et al. [10], done by Supercomputing Systems (SCS) on behalf of our cooperation partner Daimler AG who use SGM for driver assistance systems. The implementation processes rectified stereo images with 12 bits per pixel, 1024×508 pixel resolution and a disparity range of 128 pixels in 68 ms. Some post filtering functions like for removing small, independent disparity patches are implemented on the CPU. Due to double buffering on the FPGA, rectification, image transfer and post filtering are effectively done in parallel to stereo matching of the (previous) image pair. Since the resolution of our cameras is only 752×480 pixels, the missing image rows and columns are filled by black pixels. Thus, currently the FPGA processes 44 % more data than it has to. In future, we will work on better matching the camera resolution to the FPGA implementation.

The third thread takes the most recently computed depth images from the second thread and computes the visual odometry. The algorithm follows mainly earlier publications [2] [15]. Firstly, features are detected in the left image by a Harris corner detector [16]. A Rank filter [13] with a window size of 15×15 pixels is applied before taking 19×19 pixels sized regions around the corners as feature descriptors. The Rank filter tolerates high radiometric differences as well as some geometric distortions. In contrast to many other publications, we do not use SIFT or SURF that require high computational demands, because scale and rotation invariance is not needed in an application with rather high frame rates, where the image content does not change much between subsequent images. Due to the available depth image, corners are reconstructed in 3D in the local camera coordinate system. Theoretically, three corresponding 3D points are sufficient for determining all six degrees of freedom of the motion between two successive camera images.

Correspondences to the previous left camera image are initially determined by matching all feature descriptors of the current image with all descriptors of the previous image. A consistency check is performed by matching additionally in the other way around for keeping only correspondences that are found in both directions. This strategy permits arbitrarily large motions as long as the images sufficiently overlap (e.g. approximately more than 50%) which is crucial for measuring fast rotations of the cameras. The outlier detection uses the fact that in static scenes the relative distance between two 3D corner positions must be the same in the camera coordinate system of the previous and current viewpoint. More details are given in previous publications [2], [15]. After removing outliers, the relative pose between both 3D point clouds is initially computed by Singular Value Decomposition [17] and optimized by minimizing an ellipsoid error [18]. The method has been extended recently [9] by estimating the relative motion error as well. This is done by assuming sub-pixel errors in the image locations of all corners and propagating them into the three translation and three rotation components of the motion.

Since visual odometry is computed incrementally from one frame to the next and every step introduces small errors due to noise, a high framerate will introduce a higher motion drift due to noise. However, if the framerate is too low for the relative camera motion, correspondences are difficult to determine, which also degrades the accuracy. Therefore, we compute the motion to the current viewpoint not only from the previous viewpoint, but from n previous viewpoints that we call key frames, where n is a small, fixed number like 5. The overall motion error as combination of the motion error



CPU thread 1
CPU thread 2
FPGA
CPU thread 3
Right cam.
Left cam.
Image acquisition
Time–stamping
Image acquisition
Rectification
Rectification
Semi–Global Matching
Corner detection / Rank filter
Depth image and time stamp
3D corner reconstruct.
Initial correspondences (correlation in image)
Outlier detection (3D rigidity contraint)
Relative pose calc. & pose error estimation
Selection of motion with minimal error
Stores for n previous viewpoints, for each:
– Corner image pos.
– Corner (Rank) descr.
– 3D corner position
– Absolute motion
– Absolute error
– Timestamp
Absolute 6D motion & error estimate

Fig. 3. Overview of image processing functions.

from one of the previous viewpoints and the motion error from the beginning of the sequence to the previous viewpoint is used for choosing the pose with the minimal overall pose error. More details are given in a previous publication [9]. The new frame replaces the one with the highest overall motion error in the list of n previous viewpoints.

With this strategy, the frames between which motion is determined are chosen automatically by minimizing the estimated motion errors. In particular, the visual odometry is drift free, if the system is standing still or moving in the same place. Moreover, this strategy permits protecting a few viewpoints of the list from being overwritten. In this way, some important places can be *remembered* and the motion error corrected when the system revisits the place. This may be seen as light weight SLAM.

C. State Estimation for highly dynamic mobile robots

For control, highly dynamic robotic systems require a delay free, robust and accurate system state estimate. This is especially essential for inherently unstable systems with fast dynamics like flying robots. The states of interest include position, velocity and attitude. For considering these requirements, we use the fusion approach introduced in [12].

Visual odometry is generally very precise in measuring position and attitude due to the very high angular resolution of cameras. However, the quality heavily depends on the texturedness as well as distance of features in the scene (i.e. for stereo visual odometry). In the worst case, visual odometry suddenly fails due to overexposed, blurred images or textureless scene parts. Another issue of visual odometry measurements is the time delay resulting from communication and data processing delays. These can easily exceed 200 ms and can not be ignored if used for control of highly dynamic systems. To increase the robustness and compensating for measurement time delays, it is necessary to combine visual odometry with other sensors.

An inertial measurement unit (IMU), measuring angular rates and accelerations on three axis, is independent of the environment it is used in. MEMS (Micro Electro Mechanical Systems) IMUs are very suitable for mobile robots as they are small and light weight. The sensor measurements are available at a high sampling rate and have a minimal time delay compared to the system dynamic of mobile robots. Theoretically, the system attitude, position and velocity can be calculated by IMU measurements only. Nevertheless, due to sensor noise and biases of MEMS IMUs the system states will drift after a short time and have, therefore, to be corrected using other sensor information.

In the following, we give a short overview of our fusion algorithm combining IMU and time delayed odometry measurements which has been published [12]. We have shown that using these two complementary sensor concepts we get a delay free and accurate state estimate at a high rate, suitable for system control. Even with time delays of 1s highly dynamic maneuvers like flips on a quadrotor were stable.

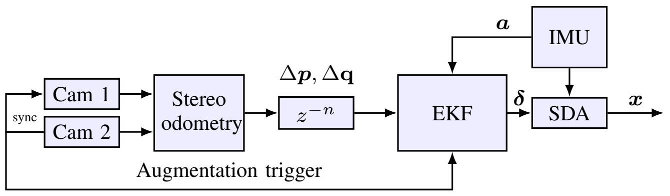


Fig. 4. State estimation system design.

1) State Estimation System Design: As shown in Figure 4, we designed our state estimator as error state space Extended Kalman Filter (EKF) in feedback configuration. Non time critical sensor processing as visual odometry runs on the Core2Duo processor board. Time critical tasks as IMU measurement processing, Strap Down Algorithm (SDA), EKF and an optional controller run on the Gumstix processor board. The camera sync signal is registered at the realtime system via a hardware trigger to compensate for measurement time delays (see III-C.2).

The realtime system calculates the direct system state including position, velocity attitude and IMU bias estimates for the IMU accelerometers and gyroscopes. The former three are calculated by the SDA at a rate of 200Hz.

As we estimate the system state errors instead of the direct system state the EKF can run at a lower frequency than the SDA, in our case at 50 Hz. The EKF propagates the state covariances in every time step linearizing at the current system state (including acceleration). The IMU noise parameters are used as error propagation system noise.

We use two measurements to estimate the current system state correction δ: Assuming small system accelerations, the IMU accelerometer measurement $\mathbf{a}$ is dominated by the gravity vector. We fuse this pseudo gravity measurement at the full filter rate to stabilize roll and pitch angle of the system. Furthermore, we fuse the time delayed delta position $\mathbf{\Delta p}$ and orientation $\mathbf{\Delta q}$ calculated by the visual odometry which will be discussed in the next section.

2) Delay Compensation for Key Frame Odometry Measurements: A general odometry system, including our visual odometry, calculates at the current time t_n the delta pose and its covariance from one point in time t_{p1} to another point in time t_{p2}. For the n-th measurement we have $t_{p1}^n < t_{p2}^n \leq t_n$. For our key frame visual odometry system t_{p1}^n corresponds to an arbitrary key frame and t_{p2}^n to the time stamp of the most recently processed image.

For enabling the EKF to process these time delayed, relative pose measurements we clone the error pose in the filter at the time of a camera measurement and save the current direct state [19]. Cloning is initiated by the hardware trigger of the camera. At the arrival of a delta pose measurement the error residual can be calculated using the buffered direct states and the delta measurement. The residual is processed directly in the filter by referencing the cloned states at t_{p1} and t_{p2}.

The visual odometry system has a limited number of key frames that can be hold. The key frame buffer of the visual odometry system and the corresponding augmented states within the EKF frame work have to be synchronized to keep the number of state augmentations at a minimum. Within the filter, the update has a complexity of $O(n^2)$ with n as the number of augmented states while the prediction has a complexity of $O(n)$ as the augmented states stay constant during prediction. For synchronization, the visual odometry system sends a list of time stamps of the currently buffered key frames with every delta pose measurement. The EKF frame work uses this information to remove all state clones and buffered direct states with time stamp $t < t_{p2}$ which are not in the key frame list. In this way we limit the number of clones in the filter to a minimum which is the number of used key frames plus the number of triggers between t_{p2} and t_n.

Using the described augmentation technique enables the filter to process further measurements from other sensors directly just at the time of arrival. These measurements can be of any type like further time delayed delta or (time delayed) absolute measurements. Compared to previous work, like [20], this approach does not need to buffer sensor data or wait for the oldest measurement and, thus, provide estimates which suffer from the longest sensor delay. Therefore, it is especially feasible for systems where measurements have different time delays.

IV. Experiments

We tested the system in several experiments. In IV-A we analyze the runtime performance for stereo image processing and ego-motion calculation. In the following section (IV-B) we present the results of accuracy tests for an indoor/outdoor round trip way of about 140 m. Finally, in Section IV-C we show the results for visual odometry and IMU fusion for different indoor/outdoor runs.

A. Runtime Performance

The system has been optimized for processing high framerates. The first thread (see Figure 3) is not computationally intensive, as it merely waits for the next camera images to arrive. The exposure and transfer time from the cameras resulted in a latency of about 65 ms from capturing the images until they are available by the first thread.

The runtime of the functions of the second thread are given in Table II. The time for rectification relates to both images. As SGM is computed on the FPGA in parallel to the other functions, the thread can compute a new image every 68 ms (i.e. with 14.6 Hz) with a latency of 104 ms per image. One CPU core is 49% busy with thread 2.

TABLE II
RUNTIME OF FUNCTIONS OF THREAD 2

Function	Resolution	Runtime in ms
Rectification	752×480	4.3
Transfer to FPGA	1024×508	10
SGM (FPGA)	1024×508	68
Post filtering	752×480	19

Table III shows the runtime of functions of the third thread. Corner detection aims at finding about 500 corners in the image. However, many of them are removed if there is no corresponding depth value or if the corner is too close to an object border as given in the depth image. All other functions are performed for each previous viewpoint that is stored. We set the list size to 5 in our experiments. This results in a latency of 54.5 ms for computing the motion to the current viewpoint. Since the third thread requires depth images from the second thread which is slower, the CPU load for the third thread is about 80% of the second CPU core.

TABLE III
RUNTIME OF FUNCTIONS OF THREAD 3

Function	Average corners	Runtime in ms
Corner det. & Rank filtering	500	22.8
Initial correspondences	5×164	10.2
Outlier detection	5×113	16.1
Motion and error calc.	5×111	5.4

According to these measurements, a framerate of 14.6 Hz can be reached with a minimum latency of 223.5 ms. The latency can increase depending on how long ago the last image pair has been captured by thread 1, which can be up to 65 ms. This increases the latency in the worst case to 288.5 ms. The total CPU load is about 129 %. The framerate, latency and CPU load have been confirmed in our experiments.

B. *Visual Odometry without Fusion*

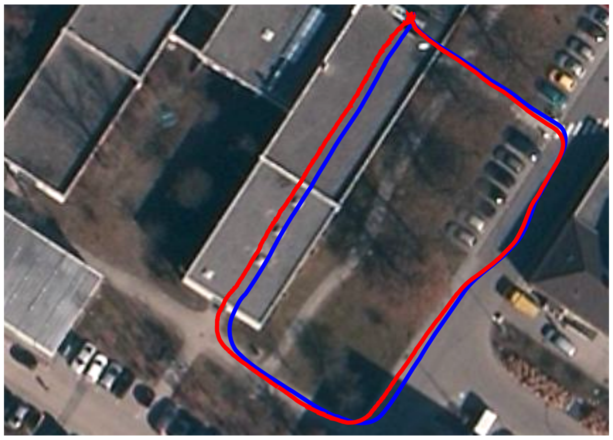


Fig. 5. Round trip of 140m with indoor and outdoor sections. Estimate of visual odometry in blue, fusion of visual odometry and IMU in red.

We have chosen a challenging round trip way through a part of our institute and a pathway outside for testing the robustness of our system. Changing between an inside and a sunny outside environment is very challenging for vision systems. The hardware has been carried by hand. We found during several runs, that motion estimation sometimes failed completely due to fast movements indoors that created blurred images, overexposed or underexposed images when going from inside out or vice versa, etc. Table IV shows the accuracy of four successful runs using visual odometry only. We measured the accuracy by manually closing the loop from the last to the first frame.

TABLE IV
ACCURACY OF VISUAL ODOMETRY ALONE ON A 140 M INDOOR/OUTDOOR ROUND TRIP WAY.

Run	Incremental, error in m	5 previous frames, error in m
1	18.4	3.8
2	8.2	3.7
3	5.1	2.4
4	7.4	1.2

It can be seen that computing pure incremental motion is inferior to using the proposed list of previous views. In the latter case, the accumulated error was about 2.7% in the worst run.

C. *Fusion of Visual Odometry and IMU*

We evaluated the accuracy of our ego-motion estimation in two different experimental setups. In the first setup we chose the mentioned indoor/outdoor run and measured the loop closure error. In the second setup we chose an outdoor trajectory to make the measurement of a position ground truth possible. Therefore, we used a Leica TPS1200 tachymeter tracking a 360° prism mounted on the perception device. The ground truth accuracy is about ±5mm.

Considering the results of our performance tests, we chose for the first experimental setup 5 key frames and 300 feature points for real-time onboard vision processing and IMU fusion. We selected a test run with medium quality results from visual odometry with a final loop closure error of about 2.9 m. Figure 6 shows a 3D plot of the round trip trajectory with pure visual odometry in blue and fusion with IMU in red.

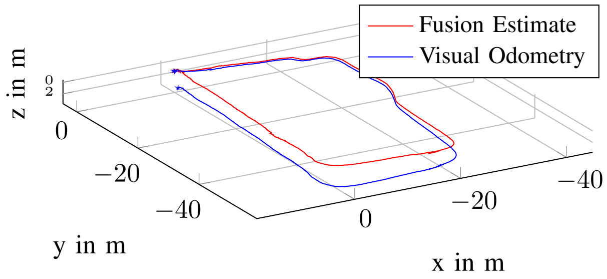


Fig. 6. Indoor/Outdoor round trip with onboard real-time high quality depth image and ego-motion calculation.

By fusing vision with IMU data, the total loop closure position error is reduced to 0.61 m. For the whole trajectory we improved the relative error from 2.1% to 0.44%.

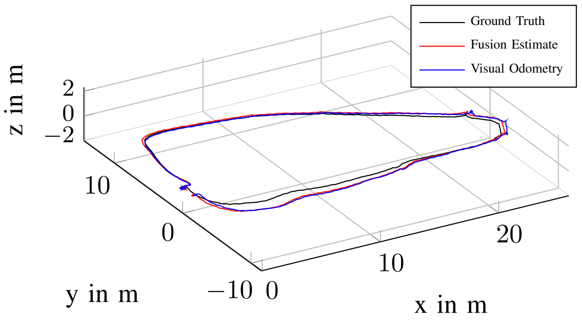


Fig. 7. Outdoor trajectory measured by tachymeter (ground truth), fusion estimation and visual odometry.

In the second experimental setup we chose an outdoor trajectory of about 70m. We conducted three sets of experiments, each consisting of four runs. For each set we varied the experimental conditions, as follows: For the first set we chose a visual odometry setup using 5 key frames and 300 feature points. For the next set we deactivated the key frame feature of the visual odometry system. In the final, set we use the same visual odometry setup as in the first and verify the robustness of the fusion algorithm by two visual odometry interuptions per run. During the first interruption, lasting about 5s, we conducted mainly a translational motion while holding a hand in front of the camera. During the second, short interruption we turned the device quickly by about 130° covering again one of the cameras. The ground truth, visual odometry and fusion estimate of the 3D trajectory of run 2 is shown in Figure 7.

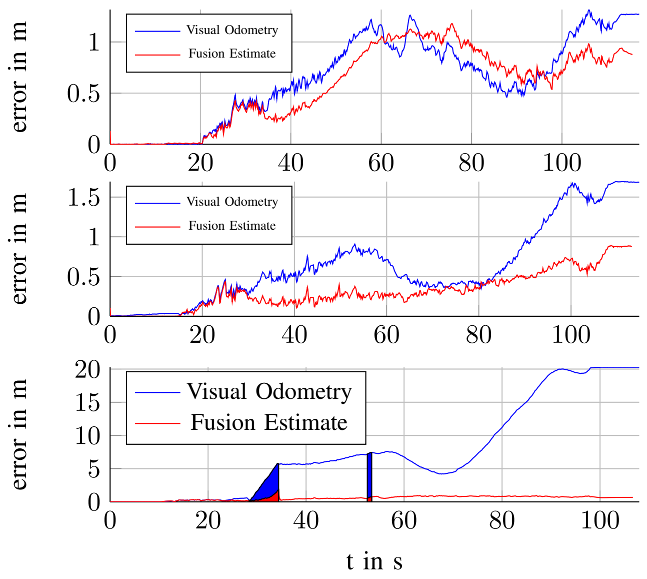


Fig. 8. Error of visual odometry and fusion estimate. From top to bottom: Run 2 (visual odometry with 5 key frames and 300 features), Run 5 (visual odometry without key frames and 300 features), Run 10 (visual odometry with 5 key frames and 300 features, forced interruptions marked as solid area)

Figure 8 depicts the absolute position errors of visual odometry and fusion estimate compared to the ground truth. Using 5 key frames the data fusion algorithm can slightly improve the position error. Deactivating key frames, the visual odometry becomes more inaccurate which can easily be seen in the first 15 seconds of run 5: The calculated position from visual odometry drifts even though the device is standing still. Forcing interruptions of the visual odometry the fusion algorithm increases the quality of the ego-motion estimate significantly compared to visual odometry only.

The results of all conducted experiments of setup two are summerized in Table V. During run 4, the tachymeter lost track shortly after the trajectory start. Therefore, no ground truth is available. During run 6, the ground truth tracking got lost at the beginning but could be recovered. Therefore, the usable track length is shorter than in the other runs.

To measure the use of key frames we calculated the ratio of used key frames and the total number of images for each run. Using 5 key frames the ratio is between 0.42 and 0.72. During the outdoor runs, the cameras were turned to mainly face the ground, therefore the number of needed key frames is relatively high. The position estimate of the fusion algorithm is better or in the range of visual odometry only. The relative position error at the end of the runs ranges between 0.93% and 1.62%. Espacially in the case of visual odometry drop outs as in run 9-12 the quality and robustness of the ego-motion estimation can be increased significantly.

It should be mentioned that even with a good quality

TABLE V
OUTDOOR EXPERIMENTS WITH GROUND TRUTH. FROM LEFT TO RIGHT: NUMBER OF RUN, DISTANCE OF RUN IN [M], NUMBER OF USED IMAGES DIVIDED BY TOTAL NUMBER OF IMAGES, MEAN OF VISUAL ODOMETRY POSITION ERROR IN [M], MEAN OF ESTIMATION POSITION ERROR IN [M], MAXIMUM OF VISUAL ODOMTRY ERROR IN [M], MAXIMUM OF ESTIMATION ERROR IN [M], FINAL VISUAL ODOMETRY POSITION ERROR IN [M], FINAL ESTIMATION POSITION ERROR IN [M], RELATIVE FINAL VISUAL ODOMETRY ERROR IN [%], RELATIVE FINAL ESTIMATION ERROR IN [%]

Run	Dist	KF/TF	$\bar{E}^{cam}_{abs}$	$\bar{E}^{est}_{abs}$	E^{cam}_{max}	E^{est}_{max}	E^{cam}_{end}	E^{est}_{end}	E^{cam}_{rel}	E^{est}_{rel}
1	68	0.42	0.32	0.36	0.81	0.70	0.59	0.63	0.88	0.93
2	69	0.53	0.64	0.58	1.32	1.18	1.27	0.88	1.84	1.27
3	67	0.51	0.48	0.40	1.01	0.95	0.90	0.77	1.35	1.16
4	–	–	–	–		–	–	–	–	–
5	68	1.00	0.64	0.32	1.69	0.89	1.69	0.88	2.46	1.29
6	63	1.00	0.92	0.57	1.66	1.13	1.66	1.10	2.64	1.75
7	66	1.00	0.61	0.39	1.44	0.90	1.32	0.86	2.00	1.30
8	68	1.00	0.71	0.51	1.78	1.49	1.70	1.45	2.51	2.13
9	70	0.71	8.52	0.57	26.47	1.14	26.47	0.84	38.06	1.21
10	69	0.69	7.66	0.55	20.28	1.86	20.28	0.67	29.22	0.97
11	70	0.71	7.65	0.63	23.25	1.75	23.25	0.97	33.26	1.39
12	70	0.72	8.13	0.38	25.86	1.28	25.85	1.13	37.16	1.62

ego-motion estimate of the visual odometry as in runs 1-8 the estimate could not be used to control highly dynamic systems as for example flying robots due to its time delay. In contrast, within the ego-motion filter framework the delay is compensated. Experimental results demonstrating the use of the fusion ego-motion estimate for control of a quadrotor using simultaniously time delayed visual and laser odometry have been published previously [12] [21]. A video of these experiments is available online [22].

V. DISCUSSION AND FUTURE WORK

The presented system is a major step on our roadmap towards a small, light-weight, real-time system for depth image and ego-motion computation for mobile robots.

In terms of weight, we see a potential in combining the CPU baseboard and the FPGA evaluation board into one smaller baseboard that contains the FPGA. Furthermore, the Core2Duo CPU cooler can be easily exchanged against a lighter version saving at least 100 g. Also power supply cables and mounts can be optimized for weight saving at least another 100 g of payload.

In terms of software, we are aiming at implementing more parts on the FPGA. Rectification would be a candidate, although it does not require much processing time on the CPU. It appears obvious to perform image acquisition directly on the FPGA as well. However, handling high level protocols like USB, Firewire or Gigabit Ethernet on an FPGA is difficult and we would lose a great deal of flexibility that is valuable for an experimental system. Instead, we are already working on moving some functions for computing the visual odometry onto the FPGA, like feature detection, description and initial correspondence determination. In this way, the

CPU load could be lowered by about 50%, which leaves more space for implementing higher level navigation on the CPU.

Concerning the IMU, only acceleration and gyroscope sensors are used. In a next step we will integrate the barometer and magnetometer measurements as absolute measurements for height and yaw angle respectively.

VI. Conclusion

We introduced a compact and highly efficient mobile robot perception device for real-time, high quality depth image and system state (pose, velocity and sensor biases) estimation. The system is designed as a sensor box usable for control of highly dynamic systems, obstacle avoidance and 3D-environment modeling.

The perception box has a size of 12 cm x 11 cm x 11 cm and a weight of 830 g. Considering the single hardware components the weight can be easily reduced by at least 200 g by optimizing CPU cooler, cables and hardware mounts.

High quality depth image calculation is realized by an SGM stereo algorithm FPGA implementation running at a rate of 14.6 Hz with a resolution of 0.5 MPixel and a latency of 250 ms. Visual odometry with key frames is calculated on a Core2Duo (@1.86 GHz) board. The CPU load is 129% using 5 key frames and 500 corners. On a indoor/outdoor round trip of 140 m visual odometry achieves a relative loop closure error of 2.7% corresponding to 3.8 m in the worst case of our test runs.

We fused visual odometry with IMU measurements to compensate time delays of the vision pipeline and robustify the state estimation. On the test indoor/outdoor roundtrip we improved the loop closure error for a medium quality visual odometry run from 2.1% to 0.44% corresponding to a relative error of 2.9 m and 0.61 m, respectively. In several outdoor experiments we demonstrated the accuracy and robustness of the system. Runs of about 70 m with forced visual odometry drop offs resulted in total final errors in the range of 0.9 m for fused ego-motion and in the range of 24 m for visual odometry only corresponding to a relative error of 1.3% and 34% respectively. The measurement delay compensated system state estimation runs at a rate of 200 Hz and can even be used for control of agile mobile robots like multicopters. The unit is going to be the next generation autopilot platform for our multicopters [21].

We see great potential in miniaturizing the system by designing a CPU baseboard including the FPGA. Implementing parts of the visual odometry on the FPGA will further lower the CPU load.

Acknoledgements

We would like to thank Supercomputing Systems (SCS) in Switzerland as well as Uwe Franke and Stefan Gehrig (Daimler AG, Germany) for the FPGA implementation of SGM.

References

[1] H. Hirschmüller, "Stereo processing by semi-global matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, February 2008.

[2] H. Hirschmüller, P. R. Innocent, and J. M. Garibaldi, "Fast, unconstrained camera motion estimation from stereo without tracking and robust statistics," in *Proceedings of the 7th International Conference on Control, Automation, Robotics and Vision*, Singapore, 2-5 December 2002, pp. 1099–1104.

[3] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *ICCV*, 2003.

[4] R. A. Newcombe, S. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *ICCV*, 2011.

[5] J. Sturm, W. Burgard, and D. Cremers, "Evaluating egomotion and structure-from-motion approaches using the tum rgb-d benchmark," in *Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS IROS*, 2012.

[6] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *ISMAR*, 2011.

[7] D. Honegger, P. Greisen, L. Meier, P. Tanskanen, and M. Pollefeys, "Real-time velocity estimation based on optical flow and disparity matching," in *IROS*, 2012.

[8] S. B. Goldberg and L. Matthies, "Stereo and imu assisted visual odometry on an omap3530 for small robots," in *IEEE Computer Vision and Pattern Recognition Workshops*, June 2011, pp. 169–176.

[9] A. Stelzer, H. Hirschmüller, and M. Görner, "Stereo-vision-based navigation of a six-legged walking robot in unknown rough terrain," *International Journal of Robotics Research: Special Issue on Robot Vision*, vol. 31, no. 4, pp. 381–402, 2012.

[10] S. Gehrig, F. Eberli, and T. Meyer, "A real-time low-power stereo vision engine using semi-global matching," in *ICVS*, vol. LNCS 5815, Liege, Belgium, October 2009, pp. 134–143.

[11] I. Ernst and H. Hirschmüller, "Mutual information based semi-global stereo matching on the gpu," in *ISVC08*, vol. LNCS 5358, Part 1, Las Vegas, NV, USA, December 2008, pp. 228–239.

[12] K. Schmid, F. Ruess, M. Suppa, and D. Burschka, "State estimation for highly dynamic flying systems using key frame odometry with varying time delays," in *IROS*, oct. 2012, pp. 2997 –3004.

[13] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondance," in *Proceedings of the European Conference of Computer Vision*, Stockholm, Sweden, May 1994, pp. 151–158.

[14] H. Hirschmüller and D. Scharstein, "Evaluation of stereo matching costs on images with radiometric differences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 9, pp. 1582–1599, September 2009.

[15] H. Hirschmüller, "Stereo vision based mapping and immediate virtual walkthroughs," Ph.D. dissertation, School of Computing, De Montfort University, Leicester, UK, June 2003.

[16] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151.

[17] R. Haralick, H. Joo, C.-N. Lee, X. Zhuang, V. G. Vaidya, and M. B. Kim, "Pose estimation from corresponding point data," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 6, pp. 1426–1446, November-December 1989.

[18] L. Matthies and S. A. Shafer, "Error modeling in stereo navigation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 3, pp. 239–248, June 1987.

[19] S. Roumeliotis and J. Burdick, "Stochastic cloning: A generalized framework for processing relative state measurements," in *ICRA*, vol. 2. IEEE, 2002, pp. 1788–1795.

[20] S. Weiss, M. Achtelik, M. Chli, and R. Siegwart, "Versatile distributed pose estimation and sensor self-calibration for an autonomous mav," in *ICRA*, 2012.

[21] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *Robotics & Automation Magazine, IEEE*, vol. 19, no. 3, pp. 46–56, 2012.

[22] http://www.dlr.de/rm/desktopdefault.aspx/tabid-7903.