Stereo and IMU Assisted Visual Odometry on an OMAP3530 for Small Robots

Steven. B. Goldberg
Indelible Systems Inc.
9411 Lurline Ave.
Chatsworth CA, 91311 USA
indeliblesteve@gmail.com

Dr. Larry Matthies
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109
Larry.Matthies@jpl.nasa.gov

Abstract

Small robots require very compact, low-power, yet high performance processors for vision-based navigation algorithms like stereo vision and visual odometry. Research on real-time implementations of these algorithms has focused on FPGAs, GPUs, ASICs, and general purpose processors, which are either too big, too hot, or too hard to program. System-on-a-chip (SoC) processors for smart phones have not been exploited yet for these functions. Here we present a real-time stereo vision system with IMU assisted visual odometry implemented on a single Texas Instruments 720Mhz/520Mhz OMAP3530 SoC. We achieve frame rates of 46 fps at QVGA or 8 fps at VGA resolutions while simultaneously tracking up to 200 features, taking full advantage of the OMAP3530's integer DSP and floating point ARM processors. This is a substantial advancement over previous work as the stereo implementation produces 146Mde/s in 2.5W, yielding a stereo energy efficiency of 58.8Mde/J, which is 3.75x better than prior DSP stereo while providing more functionality.

1. Introduction

This paper addresses the need for real-time, passive, 3-D perception and motion estimation on small, unmanned ground and air vehicles (SUGVs and SUAVs) for autonomous navigation, reconnaissance, and related applications. Examples of the class of SUGVs we target include wheeled, tracked, and legged vehicles from a few kilograms down to a few 10s of grams [1, 2]. We target SUAVs with vertical take-off and landing capability (VTOL) ranging in total weight from about 500 grams (g) down to under 50 g, including a variety of rotorcraft [3, 4]. Clearly, such systems have very constrained budgets for size, weight, and power (SWaP) of components, especially given the need to maximize endurance in real applications.

Vision capabilities for such robots can be provided by monocular, stereo, or other camera configurations. 3-D perception is central to autonomous navigation. Motion estimation requirements begin with estimating the egomotion of the robot as it moves through its environment, but ultimately must include perceiving dynamic elements of the scene. Even at these small sizes, robotic navigation systems almost always include inertial sensors (IMUs) for motion estimation and can be exploited to aid visual tracking.

Our goal is to develop vision systems for such vehicles that meet their SWaP constraints while simultaneously maximizing performance, programmability, and upgradability. System-on-a-chip (SoC) processors for the smart phone and related markets appear to be a very good fit to these goals. For example, the OMAP3530 from Texas Instruments includes a general purpose processor with floating point and a vector instruction set (ARM Cortex8a), an integer DSP (C64), a video input pipeline with additional low-level image processing capabilities, a graphics engine, a display driver, and abundant other I/O, including support for R/F communications. Indeed, this chip is the processing engine in the Gumstix Overo Fire board that is used in various robotics research projects, and this chip has been integrated into several SUGV and SUAV research vehicles [5, 6]. An aggressive upgrade roadmap exists for such processors, while maintaining very low power dissipation, because of the very large mobile computing and communications market they serve.

While some computer vision-oriented functions exist for these processors, such as the VLIB object code library from Texas Instruments, we are not aware of published descriptions of implementations of 3-D perception and motion estimation algorithms on these chips; therefore, despite significant promise, their potential performance in this application is unknown.

This paper fills this gap by developing and evaluating the performance of dense stereo vision and IMU-assisted visual odometry running simultaneously on an OMAP3530. Our goals here are to assess runtime potential, not to innovate the algorithms *per se*. We achieve dense SAD-based stereo throughput of 46 fps at QVGA or 8 fps at VGA resolutions while simultaneously tracking up to 200 FAST features. The stereo performance is equivalent to 146 million disparity estimates/second (Mde/s), or 58.8 million disparity estimates/Joule (Mde/J) for the Gumstix OMAP3530 board, which uses 2.5W. This energy efficiency metric has only recently begun to appear in the literature [7]; our value is 3.75x better than the C64

implementation of local optimization-based stereo reported in [7] and includes visual odometry at the same time. Especially given the performance increases that can be expected in this class of processors in the near future, this strongly supports their use for visual navigation functions in SUGVs and SUAVs.

The remainder of this section reviews prior work. Section 2 describes the hardware architecture of our system, section 3 describes the software architecture, and section 4 gives runtime results. Since the focus and contribution of this paper is to determine the performance possible with the OMAP3530 processor using existing algorithms, we do not evaluate error rates of these algorithms here; abundant prior literature does that.

1.1. Related work

There has been a substantial amount of prior work on real-time implementations of vision algorithms, including stereo vision and feature tracking, on commercially available FPGAs, DSPs, GPUs, and GPPs, as well on proprietary ASICs for vision that are becoming available as part of products, for example for smart cars. There is also ongoing basic and applied research on custom analog and mixed-signal chips for low-level vision functions, such as optical flow. Very little of this work has been targeted to run onboard very small robots; therefore, we briefly review this whole space, but mainly focus on work that is relevant to the class of applications addressed here.

The first real-time implementations of stereo vision were local optimization (e.g. SAD) algorithms implemented in FPGA, custom image processing boards, Intel GPPs with SIMD units, and/or DSPs [7, 8, 9]. Custom ASIC implementations have also appeared, in particular using the Census algorithm [10]. Local optimization algorithms have been used very successfully for obstacle avoidance in autonomous navigation of ground robots, though mostly off-road [11].

While research continues on improving real-time local optimization algorithms and exploiting new embedded processors for them [e.g. 12], recent work has produced real-time implementations of semi-global optimization algorithms, especially based on dynamic programming, partly to provide more range data more reliably in low-texture scenes, such as indoors. This includes FPGA, GPP, GPU-based implementations [13, 14, 15, 16].

A comprehensive survey of this work is included in [17].

Representative, recent Mde/s values include 2406 Mde/s [18] for SAD-based stereo in FPGA and 1067 Mde/s for dynamic programming-based semi-global optimization in a GPU [17]. However, this metric does not capture SWaP, which is critical to SUGV and SUAV applications. Humenberger [17] introduces million disparity evaluations/Joule (Mde/J) as a metric for energy efficiency of stereo algorithms and gives 2.29 Mde/J for GPP, 5.21 Mde/J for GPU, and 15.68 Mde/J for C64 for a Census-based local optimization algorithm. The DSP is far superior to GPP and GPU in energy efficiency and in the ballpark to be fast enough for small robots. It is also much more easily programmable than FPGAs; SoCs like the OMAP3530 are also physically much smaller than high-capacity FPGAs.

For robot navigation, especially in terrain where wheel slip is significant and GPS reception is poor or unavailable, visual egomotion estimation (i.e. visual odometry, or "VO") is important and is becoming a standard part of vision systems. Typical approaches involve detecting and tracking point features, using stereo to provide range to the features, and estimating the 6-DOF motion of the robot from the temporal displacement of the features. Many different approaches have been taken; [19] gives a recent survey. Recent research has tended to focus on increasing the accuracy of these methods for long distance navigation by exploiting variants of bundle adjustment and related methods to smooth results over time [20]. Combining VO with dense stereo vision enables efficiencies on the visual odometry side, since a separate stereo for feature correspondences is not necessary [21], and position errors of < 1% of distance traveled are achievable even without multi-frame estimation algorithms [21]. With the emphasis here on low-SWaP solutions, we adopt the two-frame approach of [21]. Floating point operations are necessary in the motion estimation stage of these algorithms, which requires a system with more than just an integer DSP.

Fusing the visual information with an IMU is important for several reasons, including capturing much faster motion dynamics (e.g. 100 Hz or faster update rates), providing a long-term sense of which way is up, and bridging over occasional failures in the VO system. It also provides an opportunity to improve the reliability and reduce the runtime of feature tracking by using the IMU to predict where to look for features in subsequent frames [22].

2. Hardware architecture

Our goals are both to evaluate smart cell phone SoCs as processing engines for vision-based navigation functions and, more practically, to identify useful components and development systems that will lead to complete visual/inertial navigation systems for SUGV and SUAV research testbeds. In this section, we discuss our choices of processor, processor boards, cameras, and inertial sensors.

2.1. Processor

Our review of prior work indicated that smart cell phone SoCs were a sweet spot in the processor space for SUGV and SUAVs. At the time we started development, it appeared that the best processor available in this trade space was the Texas Instruments OMAP3530. This is a heterogeneous dual core SoC with a 720Mhz ARM Cortex A8 and 520Mhz C64x+ DSP. It is available in several board variants and these boards are supported by large online user communities, which helped us through integration issues.

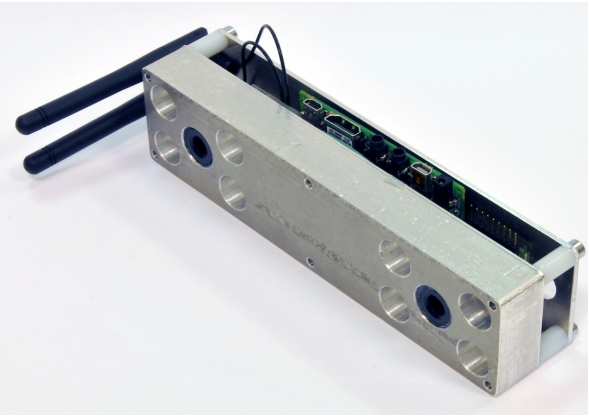

Figure 1: 12cm baseline stereo cameras, OMAP3530, IMU, and WiFi

The DSP side of the OMAP3530 has 80 KB of L1SRAM and 96 KB of L2SRAM accessible to the programmer. We chose to allocate 4 KB of the L1SRAM and 32 KB of the L2SRAM to cache. We then utilized the remaining 76 KB of L1SRAM and 64 KB of L2SRAM as scratch memory. When computing 512x384 stereo with 51 disparities, all our temporary memory products fit within the L1SRAM. Only when computing 640x480 stereo with 60 disparities, we completely fill this 76 KB with temporary memory products and begin utilizing some of the L2SRAM for additional scratch space.

Being a C64x+ class DSP means it lacks a floating point unit, but does contain powerful VLIW SIMD instructions and eight functional units. Our DSP side software has been carefully optimized to make use of these functional units. Almost all of our DSP side software has been written using compiler intrinsics. Special care was taken with each step of the algorithm to ensure it made the best use of each of the functional units.

The ARM side of the OMAP3530 allows us to run a normal Linux operating system and leverage all the software we have already written for Linux. The ARM side has 32 KB of L1 and 256 KB of L2 cache. This is not directly available to user, and we make little attempt to optimize its distribution. However, the ARM does offer floating point and a powerful SIMD (Simultaneous Instruction on Multiple Data) instruction set called NEON. We make use of these instructions in our ARM side feature matching.

2.2. Board selection

With many manufacturers offering products with OMAP3530's, we chose to start with the widely used BeagleBoard as a development system. It has one of the most active user communities and allowed us to begin developing software with a minimum investment of time and money. During development, a newer version of the BeagleBoard, the BeagleBoard xM, became available with a faster DM3730 processor. The DM3730 is a direct descendent of the OMAP3 processor family and is on the product roadmap of several OMAP3 board vendors. To evaluate how our software will scale to newer OMAP3-style processors, we purchased a BeagleBoard xM and were able to test the processing power and electrical power consumption.

BeagleBoards are not optimized for size and weight, hence are not useful for integration onboard SUGVs/SUAVs. Good, small, candidate boards for onboard use are available from LogicPD (15 x 27 x 3.8 mm and 2g) and Gumstix (17 x 58 x 4.2 mm and 5.3g). We are currently using the Gumstix, because they offer a wide variety of compact peripheral boards, whereas to use LogicPD will require developing peripherals.

2.3. Camera selection

Desirable characteristics in cameras for mobile robots include simultaneous global electronic shutter, low power dissipation, good low-light sensitivity, and good near-infrared response (for use with optional onboard light sources). Aptina makes several CMOS imagers with these characteristics for automotive and surveillance markets. Camera products with these imagers are available from Point Grey, Gumstix, and other sources. A key issue is interfacing with the processor. One drawback of the OMAP3530 is there can be only one active camera on the chip's parallel camera bus. It may or may not be possible to multiplex two image sensors directly to this bus; for the time being, an easier solution has been to use two Point Grey FireFly MV USB cameras based on the Aptina MT9V022 global shutter image sensor. These cameras utilize the IEEE IIDC standard over USB, which is well supported by the linux operating system through the libdc1394 [23] library. These cameras each consume about 1 W when running at full speed (compared to the < 350 mW specified by Aptina for the MT9V022 sensor alone).

Stereo imaging requires the two cameras be synchronized. To synchronize our FireFly MV cameras, we choose to connect both camera trigger inputs together to one GPIO connector on the left camera. We could have used a GPIO connector from the Gumstix board or a GPIO

line from the IMU to trigger the cameras, but our configuration made it simpler to connect the cameras to other computers without the need to change the software.

2.4. Inertial sensors

Inertial sensors are becoming universal in robotic vehicles, for good reason. For our system, we chose an attitude and heading reference system (AHARS) from CH-Robotics (CHR-6dm) that provides 3 axis rate gyros, 3 axis accelerometers, 3 axis magnetometers, 6 GPIO lines and an additional floating point ARM processor. The CHR-6dm also has open source firmware [24] that we modified to include a clock and GPIO interrupt line, which is connected to the camera strobe output. When the cameras are triggered, the interrupt service routine registers the time of the cameras' exposure. A second interrupt service routine registers when the onboard analog to digital converters have finished reading the rate gyros. These two times allow us to exactly synchronize the camera exposure and the acquisition of inertial measurements. The accumulated rate gyro information and the camera trigger times are reported to the OMAP over an RS232 line.

3. Software architecture

Here we focus on stereo vision and IMU-aided visual odometry for SUGV applications; we expect that elements of this implementation will carry over to SUAVs in future work. There are many stereo vision and visual odometry algorithms in the literature. Since our objectives here are to advance the state of the art in low-SWaP implementation for small robots, not necessarily to innovate the algorithms *per se*, we choose highly efficient algorithms that have been well-proven in field testing. Howard's system [21] combines SAD-based, local optimization stereo with two-frame VO using FAST features. By exploiting the dense depth map to provide stereo correspondence for the FAST features, it is possibly the fastest VO algorithm in existence, while achieving very respectable position errors of 0.25% of distance traveled on datasets covering 400m of travel. This system and others using very similar stereo algorithms were heavily tested in DARPA's LAGR program, which gives a solid basis for their use in a first evaluation of the OMAP3530 for such algorithms [25]. In the future, we may explore cost/performance trade-offs of algorithm variants.

The novelty of our system is the parallel computation of stereo vision and visual odometry on both cores of the OMAP SoC. All stereo related computation is handled on the C64x+ side of the OMAP, while feature detection, matching/tracking, and egomotion estimation is handled on the ARM side. This is a convenient division of processing, as stereo computation is entirely an integer

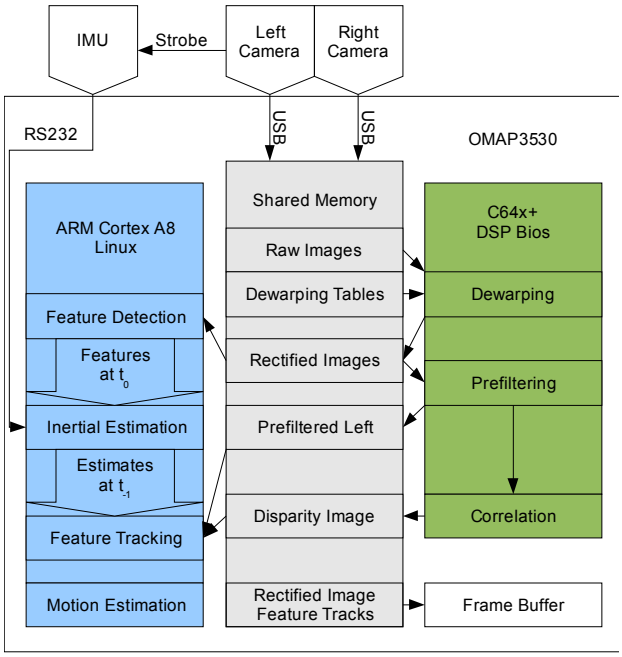

Figure 2: Software architecture diagram.

process, well suited to the integer only C64x+, while several parts of visual odometry involve floating point operations. We used the TI codec engine's IUniversal wrapper to integrate the ARM and DSP processes. The C6Accel library, which provides a similar wrapper, was released after we completed the development of this portion of the software.

3.1. Image acquisition

Given the Linux environment on the ARM side, we were able to leverage an existing IEEE IIDC library, libdc1394 [23], to capture images from the FireFly MV USB cameras. We modified this library to allocate the capture ring buffer from the memory shared between the ARM and the DSP. This allows us to achieve a zero-copy architecture.

3.2. Stereo pipeline

As shown in figure 2, our stereo pipeline is implemented on the DSP side of the OMAP. In general, it follows the same steps as defined in [21].

3.2.1 Rectification

Our raw cameras are calibrated and through an epipolar alignment process that produces linearized models. Then we compute 32-bit rectification tables that contain 16 bits of address and two 8-bit sub-pixel values. To allow the rectification process take full advantage of the available scratch memory, we rearranged the these tables from scanline order into blocks. The blocks are of such a size

that two blocks of rectification tables, two blocks of raw image data, and two blocks of rectified images all fit within the L1SRAM. This allows us to DMA new data into one buffer while the rectification function is processing another buffer. The blocks also allow us to use a 16-bit address in our rectification table, since the maximum possible size of a block, defined by 76 KB of L1SRAM, is about 5329 pixels. In the usual scan line arrangement of a rectification table, the address part needs to be large enough to encode the source address through the whole image. For example, a 640x480 image would require 19 bits of address, allowing for only 13 bits for sub pixel precision or 12 bits if the x and y parts are uniform.

3.2.2 Prefiltering

In order to remove intensity differences, we filter the rectified images using a difference of boxes. Since the right side prefiltered image is not used outside the stereo correlator, we implemented the prefilter step inline with the correlator and only store the complete left filtered image for use by the ARM side feature matching process. We do store 8 lines of filtered left and right images in the L1SRAM for use by the correlation stage.

3.2.3 Correlation

To compute stereo correspondence, we feed the 8 lines of filtered left and right images into a 1D correlator. The correlator computes a score for each potential match using the sum of absolute differences (SAD) between a 7x7 window in the left and right images. To optimize these operations for the DSP, we maintain the scores for each scan line at each disparity and update the scores for each new row processed. This dramatically reduces the amount of computation required at each new scan line. To find the optimal match, we keep the index and score of the smallest score in one 16-bit value. The top 6 bits of this value represent the index and the lower 10 bits represent the smallest score. We could have stored the score and index separately, but this arrangement reduces the total number of instructions and memory accessed in the inner loop. This method also introduces a new filter on our stereo computation, in which the best match must have a score below 1024 as it would otherwise not fit in the 10 available bits. Given the low dynamic range of the prefiltered images, this filter not effect the results.

In the correlation step, we compute the minimum score and index from both the left and right perspectives.

3.2.4 Sub-pixel disparity estimation

After the index of the minimum score is found, we compute a sub-pixel approximation by fitting a curve to the minimum score and two neighboring scores. We reject pixels with insufficient curvature in this fit. The sub-pixel estimation routine involves an integer division, which we have specially coded using the compiler intrinsics. This makes a significant difference in our overall runtime.

Also, during the sub-pixel computation, we must apply a minimum/maximum disparity test, since the sub-pixel estimation process requires one score to the left and right of the minimum. Therefore, matches at the minimum or maximum disparity are rejected. We also reject disparities where the minimum index from the left perspective is not within one value of the minimum from the right perspective, which is a variant of the standard left-right check.

3.3. Feature matching and tracking

We currently have two implementations of our feature tracking pipeline. The first uses the inertial sensors to estimate the location of features prior to matching and the other uses the algorithm exactly as defined in [21], which did not use inertial sensors. These approaches will be unified in the near future.

Both approaches start with the same FAST-based corner feature detection. We compute a batch of around 200 corner features using the FAST method described by [26] and we filter these features to ensure we have valid stereo disparity at each feature point. The FAST algorithm is applied to 16 different image regions over a 4x4 grid. This allows us to choose the best threshold for each region and ensures we get a good distribution of features. If too many features are computed in a single region, the threshold is adjusted and the FAST algorithm is run over the same region until the number of features computed is less than 1/16 the total number of features requested. If there are too few features computed, the threshold is adjusted but the algorithm is not rerun. The 16 thresholds are remembered from frame to frame.

3.3.1 Feature tracking using inertial estimation

Similar to [22], this implementation computes the image to image change in our inertial reference frame and uses this to estimate change in image feature positions between frames. The change in our inertial frame is a simple accumulation of rate gyro information from our IMU between image timestamps. Since we are only interested in the image to image motion over a short period of time, we do not compute absolute orientations but rather just the change in orientations. We use the inertial estimate to create a secondary camera model from the initial linearized stereo camera model. Then we project the features through our initial camera model into 3D space and back into our secondary camera model. This requires that each feature point have valid stereo disparity. The reprojection gives estimated locations of each feature in the previous image. [22] was only a monocular system, so differs in the details of the projection.

Given this estimate, we then perform a local template based search around the predicted feature location to find

a sub-pixel location of the feature in the previous frame. The size of the area searched is relative to the amount of inertial motion computed and hence related to the uncertainty of the inertial estimate. Unlike [22] (which used a GPU), we do not modify the templates, so radical changes in scale or pose change between frames is not pre-compensated.

3.3.2 Feature matching

Our second implementation is similar to [21]. After feature detection has been performed in both the previous and current left rectified images, we compute a score matrix where each feature in the current frame is compared to all the features in the previous frame using a SAD operation over a 7x7 window. The SAD operation has been implemented using the ARM side NEON SIMD intrinsics.

The score matrix is analyzed and the correspondences with the lowest SAD scores are selected. These correspondences are then grouped into sets that describe rigid transformations and the set with the most inliers is selected.

3.4. Egomotion estimation

For egomotion estimation, we again follow the procedure as described in [21]. Given a set of feature correspondences between successive stereo frames, we solve for the homogeneous transform using the standard Levenberg-Marquardt least-squares algorithm. Since the rigidity test already discards most outliers, we can then run a very efficient data editing loop that discards matches with residuals exceeding a threshold and re-run the optimization.

4. Results

Given the purpose of this paper, we focus performance evaluation on the overall runtime and system efficiency.

4.1. Stereo results

The results in Figures 3 and 4 show sample data from our DSP implementation of stereo. Qualitatively speaking, the depth map is quite good, and typical of what this class of algorithm produces in scenes like this. Comparing to ground truth is not our goal here, and in any case ground truth is very difficult to obtain in unstructured, outdoor scenes.

In Table 1, overhead includes ARM side cache write-back and invalidation. The optimization of the rectification tables allowed for a significant improvement in rectification speed and incurred only a small increase in pre-runtime setup. A left-right consistency check is computed in the correlation stage and used in the sub-pixel stage. The left-right consistency check accounts for about 5% of their combined runtime. We usually use 10% of the image width as the maximum disparity search range, but in the case of the 640x480 images, we had to use 60 disparities as we became resource limited. Our sub-pixel estimation uses optimized fixed point division, which accounts for its low runtime.

Figure 3: 512x384 rectified image

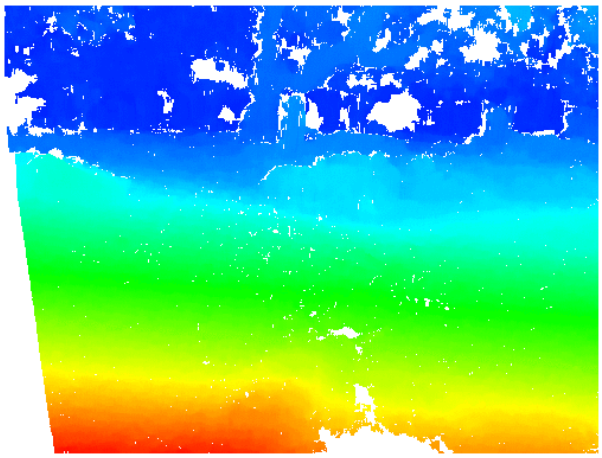

Figure 4: 512x384 false color disparity.

Table 2 gives a review of relevant DSP stereo runtimes and power consumptions. Khaleghi et al. in [27] demonstrates a single, small, low power stereo system based on a dual core Blackfin DSP. They implemented a 3x3 local Census algorithm and integrated a pair of image sensors. Their system is capable of 160x120 stereo over 30 disparities at 20fps on a 600Mhz CPU (19.2 Mde/Ghz) and they cite a total system power of 2.3W. It should be noted they do not call out the power usage of the image sensors. Ambrosch in [12] implemented a sparse Census algorithm using two 1 GHz TI C64+ DSPs. Their system is capable of 450x375 stereo over 60 disparities at 11.8 fps

(120.23 Mde/Ghz). They do not discuss the total system power, but they do utilize two 1 GHz DSPs similar to those in Humenberger [17] ,which when combined can consume over 3.4 W [28] for the CPUs alone. Chang in [29] demonstrated a jigsaw SAD implementation capable of 384x288 over 16 disparities capable of 50 fps on a 1 GHz TI DSP (88.47 Mde/Ghz). The stereo implementation described by Humenberger in [17] claims 78.38 Mde/s on a 1 GHz DSP using 5 W, for a total efficiency of 15.68 Mde/J. Our implementation is capable of 146 Mde/s on a 520Mhz TI DSP (280.77 Mde/GHz) and only uses 2.5 W, for an energy efficiency of 58.8 Mde/J on the OMAP3530 and 84.8 Mde/J on the DM3730. We attribute the majority of this difference to the level of optimization achievable via our SAD approach and a significantly (7x) faster sub-pixel estimation implementation.

Stereo	320x240x32	512x384x51	640x480x60
Overhead	3ms	6ms	10ms
Rectification	0.5ms	2ms	3ms
Box-filter	1ms	3ms	5ms
Correlation	14ms	55ms	98ms
Sub-pixel	3ms	5ms	8ms
Frame Rate	46fps	14fps	8fps

Table 1: Stereo runtime performance on 520Mhz C64x+ side of OMAP3530

DSP Stereo	Method	Mde/Ghz	Mde/J
Khaleghi [27]	3x3 Census	19.2	5
Humenberger [17]	8x8 Census	78.38	15.68
Ambrosch [12]	Sparse Census	120.23	n/a
Chang [30]	4x5 Jigsaw SAD	88.47	n/a
OMAP3530	7x7 SAD	280.77	58.8
DM3730	7x7 SAD	264.96	84.8

Table 2: Comparison of DSP stereo runtimes and power utilization.

4.2. Visual odometry runtime results

This paper also goes further than any other DSP vision system by simultaneously computing dense stereo and visual motion estimation in a single low power chip.

Tables 3 and 4 show the runtime breakdowns of our visual odometry implementation. Notice the relatively low cost of the FAST feature detector. This is important because feature detection is the only process in the visual odometry algorithm that is resolution dependent. Therefore, the varying frame rates are more directly related to the number of inliers and not to resolution. Also notice the non-linear growth of matching with respect to initial features. That is due to the SAD-based comparison of each feature in the current frame to all the features in the previous frame. Similarly, inlier selection is also nonlinear with initial features, as it tries to find the largest clique that satisfies a rigid transformation for each correctly matched feature with corresponding disparity.

Initial Features	200	300	400
Detection	7ms	7ms	8ms
Matching	6ms	12ms	23ms
Inlier	13ms	24ms	50ms
Motion Estimation	16ms	20ms	21ms
Inlier Features	80	115	140
Frame Rate	14fps	10fps	7.5fps

Table 3: 512x384 visual odometry runtime breakdown on 720Mhz ARM side of OMAP3530

Initial Features	150	200	300
Detection	4ms	5ms	5ms
Matching	3ms	7ms	12ms
Inlier	2ms	3ms	5ms
Motion Estimation	12ms	14ms	13ms
Inlier Features	18	24	30
Frame Rate	46fps	34fps	28fps

Table 4: 320x240 visual odometry runtime breakdown on 720Mhz ARM side of OMAP3530

5. Conclusion

We presented an OMAP3530-based stereo vision and visual odometry system for use on very small, SWaP-constrained robot vehicles. Our DSP SAD stereo implementation achieves energy efficiency of 58.8 Mde/J, better than all previously published DSP stereo implementations while simultaneously computing visual odometry with 100-200 features. To our knowledge, this is the first published description of such algorithms on the OMAP3530 and the first to combine both algorithms in one SoC. These results demonstrate the feasibility of using this architecture for vision-based navigation in SUGVs and show great promise for implementing similar functionality with it for SUAVs.

References

[1] http://www.qinetiq-na.com/products-dragon-runner.htm

[2] A.M. Hoover, S. Burden, X.-Y. Fu, S. Sastry, and R.S. Fearing, “Bio-inspired design and dynamic maneuverability

of a minimally actuated six-legged robot," *IEEE Int'l Conf. on Biomedical Robotics and Biomechanics (BioRob)*, 2010.
[3] http://www.asctec.de/universities-research-2/
[4] S. Zarovy, M. Costello, A. Mehta, A. Flynn, G. Gremillion, D. Miller, B. Ranganathan, and J.S. Humbert, "Experimental study of gust effects on micro air vehicles," *AIAA Atmospheric Flight Mechanics Conference*, 2010
[5] http://www.irobot.com/gi/research/Semi-Autonomous_Operations/LANdroids_robot
[6] L. Meier, F. Fraundorfer, M. Pollefeys, "Object recognition on the PIXHAWK micro air vehicle," *4th Int'l Conf on Cognitive Systems*, 2010
[7] K. Nishihara, "Practical real-time stereo matcher," *Optical Engineering*, 23(5), 1984.
[8] L. Matthies, "Stereo vision for planetary rovers: stochastic modeling to near real-time implementation," *International Journal of Computer Vision*, 8(1), 1992
[9] K. Konolige, "Small vision systems: hardware and implementation," *8th International Symposium of Robotics Research*, 1998
[10] Tyzx, Inc. http://www.tyzx.com
[11] M. Bajracharya, A. Howard, L. Matthies, B. Tang, and M. Turmon, "Autonomous off-road navigation with end-to-end learning for the LAGR program," *Journal of Field Robotics*, 26(1), 2009
[12] K. Ambrosch, C. Zinner, and H. Leopold, "A miniature embedded stereo vision system for automotive applications," *26th IEEE Convention of Electrical and Electronics Engineers in Israel*, 2010
[13] M. Humenberger, T. Engelke, and W. Kubinger, "A Census-based stereo vision algorithm using modified semi-global matching and plane fitting to improve matching quality," *Embedded Computer Vision Workshop*, 2010
[14] R. Kalarot and J. Morris, "Comparison of FPGA and GPU implementations of real-time stereo vision," *Embedded Computer Vision Workshop*, 2010
[15] S. Mattoccia, "Fast locally consistent dense stereo on multicore," *Embedded Computer Vision Workshop*, 2010
[16] S.K. Gehrig and C. Rabe, "Real-time semi-global matching on the CPU," *Embedded Computer Vision Workshop*, 2010
[17] M. Humenberger, C. Zinner, M. Weber, W. Kubinger, and M. Vincze, "A fast stereo matching algorithm suitable for embedded real-time systems," *Computer Vision and Image Understanding*, v. 114, pp. 1180-1202, 2010
[18] C.Y. Villalpando, A. Morfopoulos, and L. Matthies, "FPGA implementation of stereo disparity with high throughput for mobility applications," *IEEE Aerospace Conference*, 2011
[19] N. Sunderhauf and P. Protzel, "Stereo odometry -- a review of approaches," Tech report 3/07, Institute of Automation, Chemnitz University of Technology, 2007
[20] G. Sibley, C. Mei, I. Reid, and P. Newman, "Vast scale outdoor navigation using adaptive relative bundle adjustment," *International Journal of Robotics Research*, 29(8), 2010
[21] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," *IEEE/RSJ Int'l Conf. on Intelligent Robotics and Systems*, 2008
[22] M Hwangbo, J-S. Kim, and T. Kanade, "Inertial-aided KLT feature tracking for a moving vehicle," *IEEE/RSJ Int'l Conf. on Intelligent Robotics and Systems*, 2009
[23] http://damien.douxchamps.net/ieee1394/libdc1394/
[24] http://www.chrobotics.com/downloads/CHR6dmRelease.zip
[25] See the entire issue of *Journal of Field Robotics*, *Special issue on Learning Applied to Ground Robots*, 26(1-2), 2009
[26] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *European Conference on Computer Vision*, 2006
[27] B. Khaleghi, S. Ahuja, and Q.M.J.Wu, "*An Improved Real-Time Miniaturized Embedded Stereo Vision System (MESVS-II) ," Embedded Computer Vision Workshop*, 2008
[28] TMS320C64x Power Consumption Calculator, http://www−s.ti.com/sc/techlit/spraa45.zip
[29] N. Chang, T.-M. Lin, T-H. Tsai, Y-C. Tseng, and T.-S. Chang, "Real-time DSP implementation on local stereo matching," *IEEE Int'l Conf. on Multimedia and Expo*, 2007