Visual Odometry for Ground Vehicle Applications

David Nistér Oleg Naroditsky James Bergen

Sarnoff Corporation

CN5300, Princeton NJ 08530 USA

Abstract

We present a system that estimates the motion of a stereo head or a single moving camera based on video input. The system operates in real-time with low delay and the motion estimates are used for navigational purposes. The front end of the system is a feature tracker. Point features are matched between pairs of frames and linked into image trajectories at video rate. Robust estimates of the camera motion are then produced from the feature tracks using a geometric hypothesize-and-test architecture. This generates motion estimates from visual input alone. No prior knowledge of the scene nor the motion is necessary. The visual estimates can also be used in conjunction with information from other sources such as GPS, inertia sensors, wheel encoders, etc. The pose estimation method has been applied successfully to video from aerial, automotive and handheld platforms. We focus on results obtained with a stereo-head mounted on an autonomous ground vehicle. We give examples of camera trajectories estimated in real-time purely from images over previously unseen distances (600 meters) and periods of time .

1. Introduction

An important application of computer vision is to autonomous navigation of vehicles and robots. Effective use of video sensors for obstacle detection and navigation has been a goal in ground vehicle robotics for many years. Stereo vision for obstacle detection and ego-motion estimation for platform localization

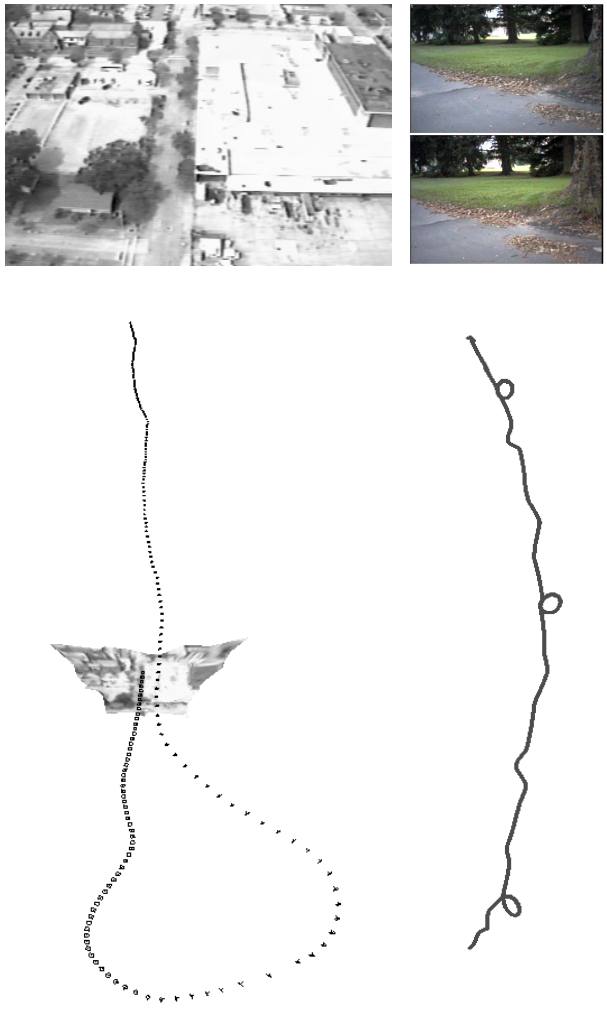

Figure 1: Left: Results with a single camera mounted obliquely on an aerial platform. The aeroplane flies at a low altitude and turns to make another sweep. The visual odometry shown was estimated in real-time with low delay. The result is based solely on visual input and no prior knowledge of the scene nor the motion is used. A textured triangulation in the frustum of the most recent camera position is also shown. Right: Results from a stereo pair mounted on a ground vehicle. The vehicle path is is over 600 meters long and includes three tight loops.

are some of the key aspects of this endeavor. Closely related is what is known in the robotics community as simultaneous localization and mapping (SLAM). SLAM has most often been performed using other sensors than regular cameras. However, relatively recent performance improvements in both sensors and computing hardware have made real-time vision processing much more practical and as computer vision algorithms mature, we expect to see more of visually based navigation systems. Such real-time applications as stereo analysis for detection of obstructions and curb or lane marker tracking for on-road control have become tractable to implement within standard PC-based systems. However, more general visual estimation functions remain difficult to achieve within the speed and latency constraints required for in-the-loop vehicle control functions.

When obstacle detection and mapping is performed using visual input such as for example stereo data, it becomes even more natural to use the visual input also to estimate the motion of the platform, so called visual odometry (Corke, Strelow and Singh 2004, Helmick et al. 2004, McCarthy and Barnes 2003, Srinivasan et al. 2000). In this paper we describe a real-time method for deriving vehicle motion from monocular and stereo video sequences. All of the processing runs at video rates on a 1GHz Pentium III class machine.

Work on estimating robot motion with stereo cameras goes back at least to Moravec's work at Stanford in the 1970s (Moravec 1980). Good results on short, indoor sequences were obtained at CMU and Stanford in the 1980s (Matthies and Shafer 1987, Kriegman, Triendl and Binford 1989). Similar work has also been going on in parallel in France (Zhang, Faugeras and Ayache 1988, Lacroix et al. 1999, Mallet, Lacroix and Gallo 2000, Jung and Lacroix 2005). Stereo visual odometry has also been used on Mars since early 2004 (Maimone et al. 2004).

Some of the components of our system have been given in detail in (Nistér 2003a,b) and we will not reproduce those here. Our system performs robust camera motion estimation based on feature tracks and is in that respect a development in the direction taken e.g. by authors (Nistér 2000, Fitzgibbon and Zisserman 1998, Pollefeys, Verbiest and Van Gool 2002) and the commercial software Boujou (Boujou 2005). However, it operates in real-time and in a calibrated framework and is in that sense more closely related to (Davison 2003, Chiuso et al. 2000, Jin, Favaro and Soatto 2000).

Our system is capable of estimating motion from both monocular and stereo video, see Figure 1. We

obtain our most reliable, accurate, and conveniently applicable results when using the system in the stereo setting. There are two major reasons for this. The first is that even with perfect estimates, the stereo setting essentially gives more information, because the overall scale of the motion is immediately and instantaneously known based on the baseline of the stereo-head. This makes it easier to integrate the visual odometry with information from other sources, and also to preserve the correct scale of motion past potential breakdowns in the motion estimation. The second reason is that the case of slow or no motion can be dealt with easily and without special handling, while for monocular estimation this is a major difficulty. We focus on ground truth evaluation of the stereo results against differential GPS and obtain good results in real-time for distances over 300 meters purely based on video.

A related system is discussed by (Olson et al. 2003). They perform extensive synthetic simulations and show that an absolute orientation sensor has to be incorporated to reduce the error growth in position to a linear function of the distance traveled. They report on results with real data from a rover traversing 20 meters. Their system is similar to ours in that it uses stereo and matches image features within stereo pairs and between successive stereo pairs. However, there are important differences. They use a Förstner interest operator in the left image and match from left to right using a pyramid, while we use Harris corner detection in all images and track feature-to-feature only. They also use approximate prior knowledge of the robot motion from robot odometry to guide the search for features over time, while we operate using imagery alone, essentially using the same matching procedure across time as within the stereo pair. They estimate 3D locations of landmark points from consecutive stereo pairs and estimate the motion of the stereo head by requiring alignment of the 3D points using a spatial Gaussian error model. They iteratively reject the landmark that appears to have moved the most. We use an image-based error model and perform full robust estimation in real-time.

The rest of the paper is organized as follows. Sections 2 and 3 present feature detection and matching, respectively. Section 4 discusses the robust estimation with one and two cameras. Section 5 gives results and Section 6 concludes.

2. Feature Detection

In each frame, we detect Harris corners (Harris and Stephens 1988). This type of point feature has been found to give detections that are relatively stable under small to moderate image distortions (Schmid, Mohr and Bauckhage 2000). Since we are using video input, we can rely on distortions between consecutive frames to be fairly small.

The exact implementation is described below, but first we give some introduction. Harris corner points essentially represent textureness maxima, where textureness is measured using the shape of the autocorrelation function around a point. The autocorrelation function is defined by comparing the pixel values in a slightly shifted window to the pixel values of the original centered window. Let $I(x, y)$ be the image function evaluated at the image point (x, y). Then the autocorrelation with shift (u, v) around point (x, y) is

$$\alpha(u, v, x, y) = \sum_a \sum_b w(a, b)(I(x + u + a, y + v + b) - I(x + a, y + b))^2, \tag{1}$$

where $w(a, b)$ denotes a weighting function that provides a window (typically Gaussian) around the origin. The Harris corner strength $s(x, y)$ is defined based on the Hessian $G_\alpha(x, y)$ of the autocorrelation function α with respect to u, v, at $u = v = 0$, or more precisely

$$G_\alpha(x, y) = \begin{bmatrix} \frac{\partial^2 \alpha}{\partial u \partial u} & \frac{\partial^2 \alpha}{\partial u \partial v} \\ \frac{\partial^2 \alpha}{\partial u \partial v} & \frac{\partial^2 \alpha}{\partial v \partial v} \end{bmatrix} (u = 0, v = 0). \tag{2}$$

The shape of the autocorrelation function can be analyzed based on the image derivatives I_x and I_y, since careful expansion yields

$$G_\alpha(x, y) = 2 \sum_a \sum_b w(a, b) \begin{bmatrix} I_x^2(x + a, y + b) & I_x I_y(x + a, y + b) \\ I_x I_y(x + a, y + b) & I_y^2(x + a, y + b) \end{bmatrix}. \tag{3}$$

The corner strength is defined as

$$s(x, y) = d(G_\alpha(x, y)) - kt(G_\alpha(x, y))^2 \tag{4}$$

where d stands for determinant, t stands for trace, and k is a constant (set below). Corners are declared at local maxima of $s(x, y)$ with respect to x and y.

There are many details such as the exact choice of filter taps, thresholding or order of computation and storage that will affect the result quality, the cache performance and the processing time. We will therefore describe our implementation in detail. Cache performance is often hard to predict and we tried many different implementations before settling on this one.

The incoming image is represented with 8 bits per pixel. We first compute the strength s of the corner response. For every output line of corner response, temporary filter outputs are needed for a certain number of lines above and below the current output line. All filter outputs are computed only once, and stored in wrap-around buffers for optimal cache performance. The wrap-around buffers represent the temporary filter outputs in a rolling window. The rolling window contains the minimal number of lines necessary in order to avoid recomputing any filter outputs. Let I_x and I_y denote the horizontal and vertical derivatives of the image. The wrap-around buffers and the resulting corner response are updated line by line, using four sweeps per line. The first sweep updates wrap-around buffers for I_xI_x, I_xI_y, I_yI_y. These buffers are 5 lines long and the typical sweep updates one line, positioned two lines ahead of the current output line of corner response. The derivatives I_x and I_y are computed by horizontal and vertical filters of the type $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ and shifted down one bit before the multiplications to keep the input down to 8 bits and output down to 16 bits. The second sweep convolves all the five lines in the wrap-around buffers vertically with the binomial filter $\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}$ to produce the three single lines g_{xx}, g_{xy}, g_{yy} of 32 bit filter output. This is accomplished by shifts and additions to avoid expensive multiplications. The third sweep convolves horizontally with the same binomial filter to produce the 32 bit single lines G_{xx}, G_{xy}, G_{yy}, stored back in the same place, but shifted two pixels. The fourth sweep computes the determinant $d = G_{xx}G_{yy} - G_{xy}G_{xy}$, trace $t = G_{xx} + G_{yy}$ and the strength $s = d - kt^2$ of the corner response, where $k = 0.06$, all in floating point.

The filter sweeps are implemented in MMX in chunks of 128 pixels and interleaved manually to avoid stalls and make optimal use of both pipelines. For details on the rules of thumb for MMX coding, see (Intel Corp 1997).

After the corner response is computed, non-max suppression is used to define the actual feature points. A feature point is declared at each pixel where the response is stronger than at all other pixels in a 5×5 neighborhood. No subpixel precision detection is used. For this computation step, it turns out that the

lazy evaluation of AND statements in C makes plain C code faster than any attempts to use MMX for the non-max suppression. The reason is that on average, a larger suppressing value is found long before the whole 5×5 neighborhood is exhausted.

In contrast to popular practice, we do not use any absolute or global thresholds on the strength of the corner response. We only use a local saturation that limits the number of detected features in a local region of the image. This saturation only sets in when the feature density becomes extremely excessive and threatens to hurt the processing time significantly. We typically allow up to 5000 feature points distributed in 10 by 10 buckets of the image, i.e. 100 features in each bucket. The survivor features in each bucket are found with the quickselect algorithm (Press et al. 1988) based on the strength of the corner response. Note however that the amount of allowed features is *very* generous. In fact, for low resolution images the limit can not even be reached, since a lower density is already enforced by the non-max suppression. The processing sweeps are detailed as pseudo-code in Appendix A. Example results are shown in 2.

3. Feature Matching

The features points are matched between pairs of frames. In contrast to the KLT tracker (Shi and Tomasi 1994), we detect features in all frames and only allow matches between features. No subpixel refinement of the matches is used. A feature in one image is matched to every feature within a fixed distance from it in the next image. That is, we match all features that are within a certain disparity limit from each other. We typically use a disparity limit that is 10% of the image size, but depending on the speed requirements and the smoothness of the input, we sometimes use as low as 3% and as high as 30% of the image size. Normalized correlation over an 11×11 window is used to evaluate the potential matches. For speed considerations, uniform weighting is used across the whole window. The key to achieving fast matching is to minimize the amount of computation spent on each potential matching feature pair. In comparison, computation spent on features separately in each image is negligible, since such computation is done only once per feature point, while every feature point is involved in a large number of potential matches. Since we only allow feature-to-feature matches we can preprocess the image patches separately. Each

11×11 patch centred on a detected feature is copied from the image and laid out consecutively in memory as an $n = 121$ byte vector (in fact, we pad to 128 bytes for convenience). At the same time, the values

$$A = \sum I \tag{5}$$

$$B = \sum I^2 \tag{6}$$

$$C = \frac{1}{\sqrt{nB - A^2}} \tag{7}$$

are precomputed for each patch. For each potential match, all we have to do is compute the scalar product

$$D = \sum I_1 I_2 \tag{8}$$

between the two patches. The normalized correlation is then

$$(nD - A_1 A_2) C_1 C_2. \tag{9}$$

The scalar product between two 128 byte vectors is computed very efficiently with MMX instructions. In fact, the multiplications can be carried out just as fast as the values can be fetched from and stored back into memory. This brings us to an interesting point. It is a common belief that the sum of absolute differences (SAD) is more efficient than normalized correlation, which is of course true in some circumstances or when considering the amount of chip surface spent in dedicated hardware, but this is not true in our setting. Changing the multiplications to subtractions would achieve nothing, since memory speed is the bottleneck.

To decide which matches to accept, we use an old but powerful trick, namely mutual consistency check. Every feature is involved in a number of normalized correlations with features from the other image, decided by the maximum disparity limit. The feature from the other image that produces the highest normalized correlation is the preferred mate. The feature from the other image also in its turn has a preferred mate. Only pairs of features that 'want to get married', i.e. mutually has each other as the preferred mate, are accepted as a valid match. Note that with good bookkeeping, the mutual consistency check can be accomplished without computing the correlations more than once.

We use the accepted matches both in stereo and in video to produce tracks. For tracking, we simply link the accepted matches between pairs of frames into tracks over time. Thus, we do not really distinguish between matching and tracking.

As with any method that links matches or tracks incrementally with some fixed time steps, the number of features tracked from the first frame to frame n falls off and the tracks have to be replenished by others. The average length of the feature tracks is data-dependent. Data that affords long tracks is of course beneficial for the visual odometry estimates.

Figure 2: Input frame (left) and feature tracker output (right). Circles represent current feature locations and curves are feature tracks through the image.

4. Robust Estimation

The feature tracking operates without any geometric constraints. The resulting tracks are then fed forward to geometry estimation. For the geometry estimation part of the system we have several incarnations. One is using monocular video as input and another one uses stereo. We also have versions that perform mosaicing with much the same methodology.

4.1 The Monocular Scheme

The monocular version operates as follows.

1. Track features over a certain number of frames. Estimate the relative poses between three of the frames using the 5-point algorithm (Nistér 2003a) and preemptive RANSAC (Nistér 2003b) followed by iterative refinement.

2. Triangulate the observed feature tracks into 3D points using the first and last observation on each track and optimal triangulation according to directional error. This can be achieved in closed form (Oliensis and Genc 1999). If this is not the first time through the loop, estimate the scale factor between the present reconstruction and the previous camera trajectory with another preemptive RANSAC procedure. Put the present reconstruction in the coordinate system of the previous one.

3. Track for a certain additional number of frames. Compute the pose of the camera with respect to the known 3D points using the 3-point algorithm (Haralick et al. 2004) and preemptive RANSAC followed by iterative refinement.

4. Re-triangulate the 3D points using the first and last observations on their image track. Repeat from Step 3 a certain number of times.

5. Repeat from Step 1 a certain number of times.

6. Insert a firewall and repeat from Step 1.

The meaning of "firewall" in this context is the following. From the system point of view, error accumulation and propagation is a serious concern. For example, if the pose is incorrectly estimated, this will lead to incorrectly positioned 3D points, which will in turn hurt subsequent pose estimates and the system will never recover. However, the above scheme opens up the possibility of building a firewall against error propagation by simply prescribing that triangulation of 3D points is never performed using observations beyond the most recent firewall. That is, for purposes of triangulation, the frame after the most recent firewall is considered the first frame. Our system then gets the desirable property that after a firewall, the relative poses will be estimated exactly as if the system was started afresh. The state of the system before the firewall can only affect the choice of coordinate system for subsequent poses, nothing else. The firewall helps protecting against propagation of gross errors. Perhaps more surprisingly, we have also found that it protects against slow error buildup that is not fully suppressed by the iterative refinements. Clearly, slow error buildup can not be completely removed without reconnecting to landmarks or performing other absolute measurements. However, we have empirically found that slow error

buildup can snowball into fast error buildup, eventually causing sudden gross errors. The firewall keeps this behavior in check, leaving only a low steady frame-to-frame error accumulation.

Note that as discussed by (Olson et al. 2003), error buildup in absolute position is necessarily super-linear unless an absolute orientation sensor is incorporated, due to rotation errors eventually turning into translation errors. However, the effects discussed above are present even in the relative motion estimates.

Our current real-time scheme is simplistic in the way it chooses which frames to use for relative orientation. We also have a version that uses model selection to choose the frames in the same spirit as (Nistér 2000). This has given promising results in offline simulations. However, it has not yet proved fast enough to improve upon the real-time results. One of the advantages of using a stereo head is that these choices are largely avoided.

4.2 The Stereo Scheme

When a stereo rig is available, we can avoid the difficult relative orientation step and instead perform triangulation followed by pose repeatedly. Moreover, the relative poses can be estimated in a known scale, since we know the size of the stereo baseline. This makes the estimates more useful and manageable. The stereo version of the system operates as follows.

1. Match feature points between the left and right images of the stereo pair. Triangulate the observed matches into 3D points.

2. Track features for a certain number of frames. Compute the pose of the stereo rig with preemptive RANSAC followed by iterative refinement. The 3-point algorithm (considering the left image) is used as the hypothesis generator. The scoring and iterative refinement are based on reprojection errors in both frames of the stereo pair.

3. Repeat from Step 2 a certain number of times.

4. Triangulate all new feature matches using the observations in the left and right images. Repeat from Step 2 a certain number of times.

5. Re-triangulate all 3D points to set up a firewall. Repeat from Step 2.

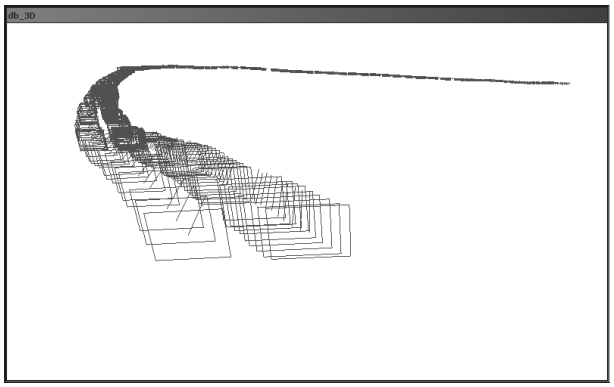

Figure 3: Camera path output from the stereo scheme in 3D.

The reader might wonder why an absolute orientation between 3D points is not used in Step 2, since 3D points can be obtained with the stereo rig at each time instance. We have also tried such a scheme and found it to be greatly inferior. The reason is the way ambiguity impacts the estimates. The triangulations are much more uncertain in the depth direction. When the same points are triangulated in two different stereo frames, there are therefore large discrepancies along the depth direction. For alignment between small sets of 3D points, this has a devastating effect on the results. For purposes of obtaining a cost function that indicates the correct motion, this problem can be avoided with larger sets of points. In fact, it was shown in the 1980's that incorporating a 3D Gaussian error model in the objective function resolves the problem (Matthies and Shafer 1987, Kriegman, Triendl and Binford 1989). However, it is not clear how to handle this efficiently in the hypothesis generation step of RANSAC and thereby achieve the robustness against outliers that RANSAC provides.

Instead, we use the 3-point algorithm for single camera pose. It is less affected by uncertainty in depth of the 3D points. The reason is that it is based on image quantities and that the new camera pose is not very far from the pose from where the 3D point was originally triangulated. The uncertainty in depth is caused essentially by the fact that moving the 3D point in the depth direction does not cause much change in the reprojected image position. Turning this around, it means that changes in depth do not change the pose estimates much. In a nutshell, if we stick to using image based quantities, the uncertainty effects 'cancel' to a large extent.

Figure 4: A picture of the autonomous ground vehicle. Note the two stereo heads mounted in the front. For this platform, a specifically tailored version of the camera motion estimation system is used. The available stereo input is leveraged to obtain even more stable visual odometry. The feature tracking front end is essentially the same, but the the known stereo geometry is used in the robust motion estimation. The somewhat more difficult relative orientation step used in the single camera case can thus be avoided. The system then essentially performs triangulation of scene points followed by estimation of pose from the structure that was previously determined. A significant advantage is that in this mode the system can perform well despite little or even no camera motion.

However, for the benefit of using image based quantities, we pay the penalty of using only one of the images in the hypothesis generation. To mitigate the effect of this, it is important to score and optimize using both images. We have tested the scheme without this improvement. The triangulation is then carried out using both views, balancing the reprojection errors between the two, but the pose is only based on one view. Any imperfections in the geometry, such as small calibration errors, will bias the triangulated positions of the 3D points. A pose estimate that only considers one view will then accomodate for some of the bias by 'forgetting' about the requirements from the other view. When the 3D points are then re-triangulated, they are again biased. This leads to a constant drift or torsion that increases with the frequency of re-triangulation. This problem is very much alleviated by scoring and optimizing based on both views simultaneously. At least, the pose estimate will not drift when the rig is motionless. Since the balanced reprojection error is essentially the same score that was used in triangulation to place the 3D points at the best possible location relative to the stereo rig, the position estimates for the stereo rig will have no reason to move based on that score.

Also, to make the hypothesis generation more symmetric between the images, we have developed a generalized version of the 3-point pose algorithm that can use non-concurrent rays (Nistér 2004b). With this generalization, it will be possible to pick the minimal sets randomly with points from both images.

The choices between triangulation and re-triangulation in the above scheme were made to accommodate certain trade-offs. We wish to re-triangulate all features quite frequently to put up firewalls against error propagation. We also wish to triangulate new features as they begin to track to avoid running out of 3D points. On the other hand, in order to suppress drift to the largest extent possible, we wish to use 3D points that were triangulated as far back in time as possible.

A significant advantage of the stereo scheme is that it can operate correctly even without any camera motion. This is also an indication of its greater stability, since many of the difficulties in monocular ego-motion estimation are caused by small motions.

4.3 Preemptive RANSAC and Scoring

For all the camera motion estimation, we use preeemptive RANSAC as presented in (Nistér 2003b). Multiple (typically 500) minimal random samples of feature correspondences are taken. Each sample contains the smallest number of feature correspondences necessary to obtain a unique solution for the camera motion. All the motion hypotheses thus generated then compete in a preemptive scoring scheme that is designed to quickly find a motion hypothesis that enjoys a large support among all the feature correspondences. All estimations are finished off with an iterative refinement with the support as objective function. Support is measured by a robust reprojection error. We assume Cauchy distribution for the reprojection errors and ignore constant factors of the likelihood whenever possible. This means that if the scaled squared magnitude of a reprojection error is u, it contributes a term $-\ln(1+u)$ to the log-likelihood. Most robustification kernels require expensive transcendental functions like in this case, the logarithm. Cauchy distribution has the advantage that we can use the following trick to prevent the logarithm from becoming a major bottleneck. The reprojection errors are taken in groups of ten and the robust log-likelihood for each group is computed as

$$-\ln\prod_{i=1}^{10}(1+u_i). \tag{10}$$

Larger groups should be avoided to ensure that the calculation stays within the floating point range.

The reprojection errors for single poses and the stereo rig are straightforward to compute since we have the 3D points. For scoring the three-view relative pose estimates, we use a trifocal Sampson error, which is significantly faster than the previous state of the art (Torr and Zisserman 1997). The closed form trifocal Sampson error is described in (Nistér 2004a). The iterative refinement for three views is however carried out with a complete, hand-optimized bundle adjustment, since full bundle adjustment turns out to be faster than any attempts to eliminate the structure parameters.

5. Results

We have tested the visual odometry system on video from various platforms, see Figure 5 for some examples.

Feature Detection	Matching with Disparity Range			SaM
	3%	5%	10%	
30ms	34ms	45ms	160ms	50ms

Table 1: Approximate average timings per 720×240 frame of video for the monocular system components on a modest 550MHz machine. Disparity range for the matching is given in percent of the image dimensions. The average timings for the stereo version are very similar, the reason being that both systems are most of the SaM processing time performing RANSAC estimations of the pose with respect to known 3D points.

In particular, we have performed extensive quantitative testing against ground truth in an automonous ground vehicle setting. The visual odometry system was integrated into a mobile robotic platform equipped with a Differential Global Positioning System (DGPS) as well as a high precision Inertial Navigation System (INS). DGPS functioning in RT-2 mode in principle allows collection of position data with up to 2cm relative accuracy when stationary. Under motion, the performance can be quite a bit worse (e.g. 0.5m), but sufficient for our evaluation. We compare the visual odometry output to the integrated INS/DGPS navigation system, which we treat as ground truth. We show that the visual odometry pose estimates are accurate and reliable under a variety of maneuvers in realistic ground vehicle scenarios. We also demonstrate the usefulness of visual odometry for map building and obstacle avoidance.

No a priori knowledge of the motion was used to produce the visual odometry. A completely general 3D trajectory was estimated in all our experiments. In particular, we did not explicitly force the trajectory to stay upright or within a certain height of the ground plane.

5.1 System Configuration

The vehicle was equipped with a pair of synchronized analog cameras, see Figure 4. Each camera had a horizontal field of view of 50°, and image fields of 720×240 resolution were used for processing. The stereo pair was tilted toward the side of the vehicle by about 10° and had a baseline of 28cm (see

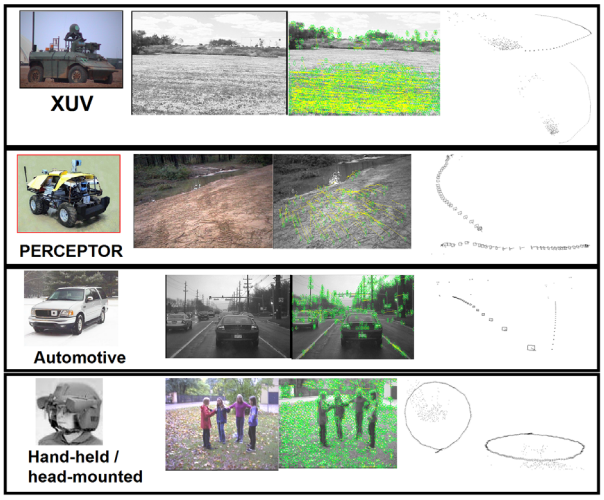

Figure 5: Various platforms used to test the visual odometry system. In particular, we have performed extensive quantitative testing against ground truth in an automonous ground vehicle setting (PerceptOR).

Figure 4). Due to a variety of other tasks running concurrently on the system, the visual odometry's frame processing rate was limited to around 13Hz.

During each run of the vehicle we collected time stamped pose data from visual odometry and the vehicle navigation system (VNS), which includes GPS and INS. To obtain quantitative comparisons, the coordinate systems of the visual odometry and the VNS were aligned by a least squares fit of the initial twenty poses. In the absence of VNS, visual odometry can be used directly for navigation relative to the vehicle's initial pose. Note that there is no need to estimate a scale factor between the coordinate systems since we can take advantage of the known stereo baseline to determine absolute scale.

The VNS was designed to select the highly accurate DGPS position estimate over the inertial estimate when both are available. The orientation of the vehicle, on the other hand, always came from the inertial sensors.

5.2 Visual Odometry vs. DGPS

Our experiments prove that visual odometry is highly effective for estimating position of the vehicle. Table 2 compares the estimates of total distance travelled by the vehicle in three outdoor runs conducted

Run	Frames	DGPS(m)	VisOdo(m)	% error
Loops	1602	185.88	183.90	1.07
Meadow	2263	266.16	269.77	1.36
Woods	2944	365.96	372.02	1.63

Table 2: Metric accuracy of visual odometry position estimates. The number of frames processed is given in column 2. Total vehicle path lengths estimated by DGPS and visual odometry are given in columns 3 and 4 with relative error in distance given in column 4.

on a trail in a wooded area. We are using distance traveled as a measure of performance because attitude and heading errors eventually dominate absolute position errors, hiding the frame-to-frame performance. The path lengths provide a measurement of how translation is estimated without distortion due to rotation errors. In official PerceptOR tests only the linear velocity of the vehicle was used when GPS drop-out was simulated, which led us to regard that error as the main performance measure.

Figure 6 shows side-by-side plots of vehicle trajectories as recorded by DGPS and visual odometry. Figure 7 shows the visual odometry overlaid on the vehicle's position in DGPS North-East-Down coordinates. In each case the visual odometry is stable and performs correctly over thousands of frames of dead reckoning and hundreds of meters of uneven terrain. This figure also illustrates the result of using visual odometry as sole means of navigation for the vehicle.

5.3 Visual Odometry vs. INS

We compare the vehicle yaw angle logged by the INS to the one from visual odometry. Unlike GPS, both the visual odometry and the INS direction sensor function incrementally by dead reckoning, and therefore accumulate error over time. Table 3 shows the mean frame to frame accumulation of discrepancy in yaw magnitude between the visual odometry and the INS for each of our three sequences.

Figure 8 further illustrates the correspondence between yaw angles of the vehicle recovered from visual odometry and INS. In most cases visual odometry exhibits subdegree accuracy in vehicle heading recovery.

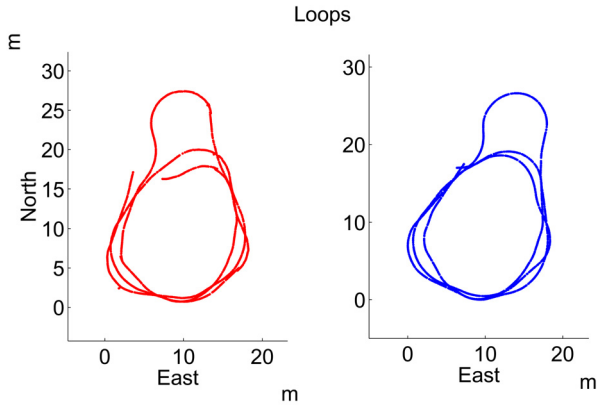

Figure 6: Vehicle positions estimated with visual odometry (left) and DGPS (right). These plots show that the vehicle path is accurately recovered by visual odometry during tight cornering as well as extended operation. In this example the vehicle completes three tight laps of diameter about 20 meters (travelling 184 meters total) and returns to the same location. The error in distance between the endpoints of the trip is only 4.1 meters.

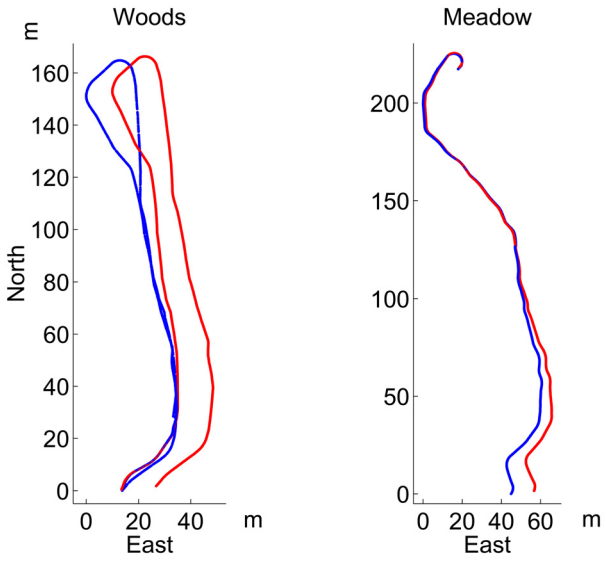

Figure 7: Visual odometry vehicle position (light red) superimposed on DGPS output (dark blue). No a priori knowledge of the motion was used to produce the visual odometry. A completely general 3D trajectory was estimated in all our experiments. In particular, we did not explicitly force the trajectory to stay upright or within a certain height of the ground plane.

Run	Std. Dev.(°)	Mean (°)
Loops	0.50	$1.47{\cdot}10^{-2}$
Meadow	0.59	$-1.02{\cdot}10^{-2}$
Woods	0.53	$2.39{\cdot}10^{-4}$

Table 3: Frame-to-frame error analysis of the vehicle heading estimates. Column 2 shows the standard deviation of the errors plotted in Figure 8 (d), and column 3 shows the mean of the distribution. We observe the approximately zero mean, which suggests that our estimates are not biased. Note that the magnitudes of these errors depend on the vehicle speed and cornering behavior during the run.

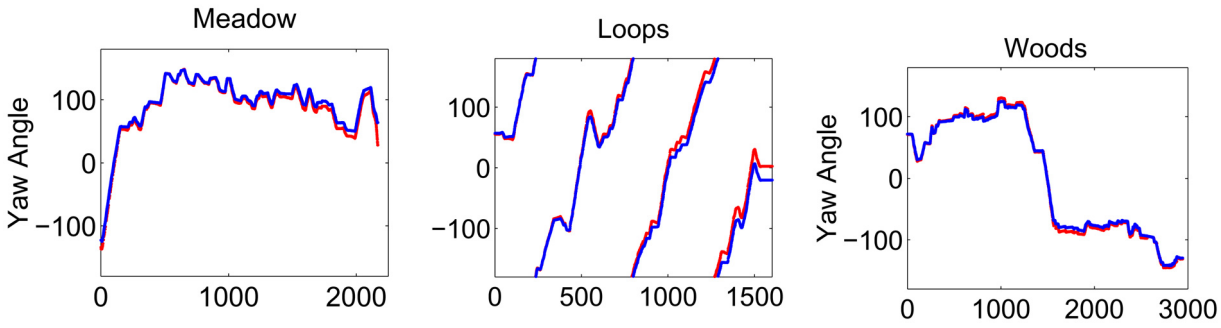

Figure 8: Yaw angle in degrees from INS and visual odometry. The correspondence is readily apparent. In most cases, visual odometry yields subdegree accuracy in vehicle heading recovery. The accumulated yaw angle is shown with respect to frame number.

5.4 Application to Mapping

Combining visual odometry and obstacle detection will allow unmanned ground vehicles to maintain a coherent map of the world over periods of extended autonomous operation. Visual odometry can also be used to supplement traditional navigation systems since it is not affected by GPS dropouts due to obstacles, wheel slip in uneven terrain or other adverse conditions. An example is shown in Figure 9.

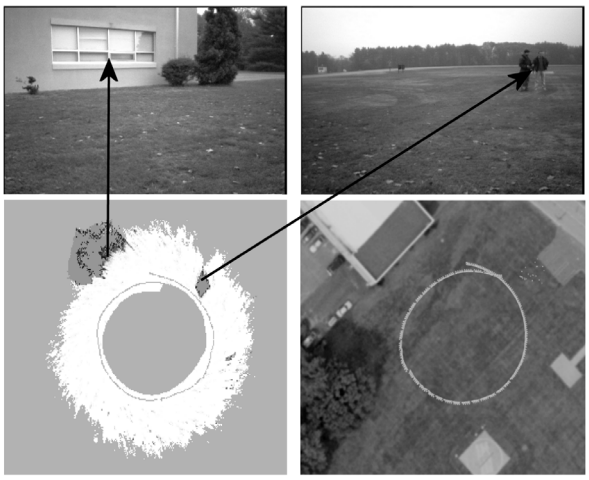

Figure 9: Visual odometry poses can be used to build accurate obstacle maps. Visual odometry poses are combined with a stereo obstacle avoidance algorithm, resulting in a map (bottom left). Note that the map is a successful merge of about 500 separate pose estimates. Obstacles such as people and buildings are retained by the vehicle despite the cameras having a small field of view ($50°$). The bottom right image shows the visual odometry data overlaid on an aerial photograph of the site.

5.5 Official Runs

We also report results from two official runs that demonstrated the visual odometry capability to DARPA. The tests where performed as part of the program PerceptOR. They were carried out in the Yuma desert in conditions with large and small boulders and desert vegetation, hills and valleys. The PerceptOR vehicle is a modified honda all-terrain vehicle (ATV), see Figure 4. The vehicle has an autonomous mode in which the pose is used to construct an obstacle map. In this mode visual odometry could only

Run	GPS	Gyro+Wheel	Gyro+Vis
Autonomous	96.09	95.80	95.32
Remote	163.14	169.92	159.64

Table 4: Total distance traversed (in meters) as estimated by DGPS, wheel encoders and visual odometry in two official test runs where the vehicle was operating autonomously and through remote control by an official who is not a member of the visual odometry team.

Run	Gyro+Wheel	Gyro+Vis
Autonomous	0.711	0.678
Remote	5.52	3.65

Table 5: Distance from true DGPS position at the end of each run (in meters).

operate at 5 frames per second. The vehicle has a mode where an operator drives the vehicle using a remote control in the form of a pendent. In this mode there are sufficient computational resources for 10 frames per second visual odometry. Only two (of four) cameras (right-looking stereo pair) were used in the trials.

One of the official runs was autonomous, and the other was controlled by a remote control. The remote control was operated by a DARPA official, not a member of the visual odometry team. In the autonomous run, the navigation system on board the vehicle was in complete control. The runs were unrehearsed. They were timed and recorded. The visual odometry was combined with gyro by using the gyro for the vehicle heading and the visual odometry for the distance traversed. The results are compared with GPS and with using wheel encoders to estimate distance traversed. The statistics are shown in Tables 4 and 5. The estimated paths are also shown in Figures 10 and 11. In Figures 12 and 13 the speed estimates from visual odometry and wheel encoders are plotted against time.

At the time of the test a mechanism for combining IMU and visual angle estimates was not ready. The combination of gyros with visual estimates was done due to the PerceptOR program requirement that vehicle pose be estimated accurately at the time of a GPS drop-out. It did not call for a purely visual solution.

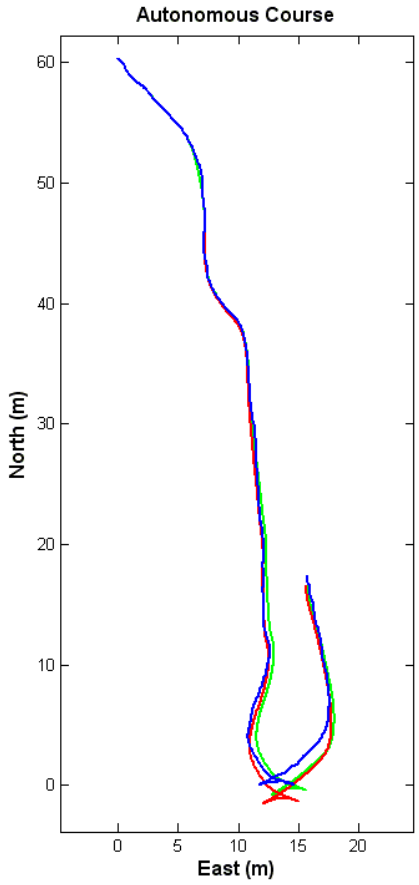

Figure 10: Estimated path of the autonomous run. Global pose (DPGS) - Dark blue. Gyro + visual odometry speed - Light Green. Gyro + wheel encoder speed - Medium Red. Note the three point turn.

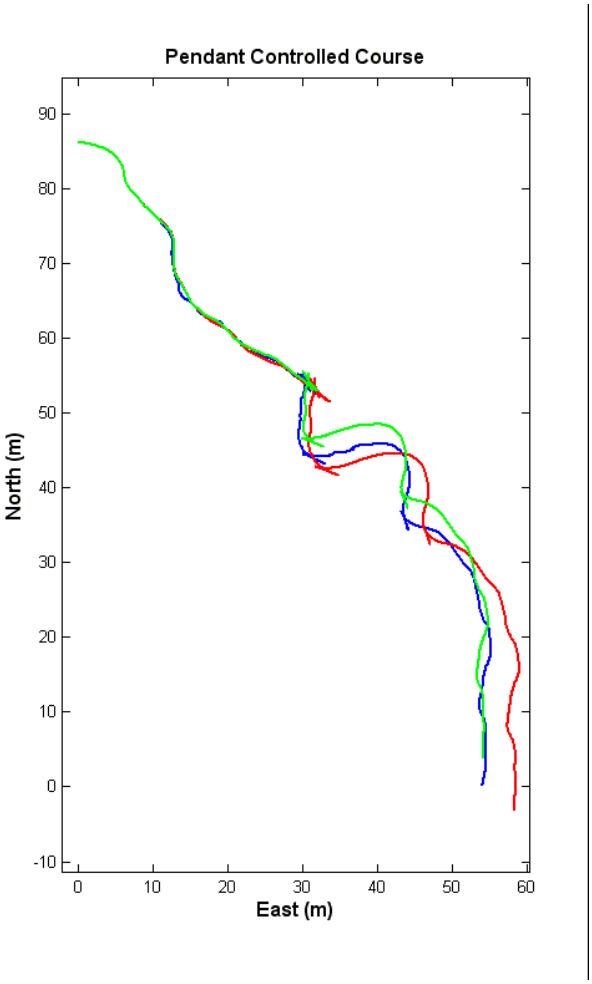

Figure 11: Estimated path of the remote operated run. Global pose (DPGS) - Dark blue. Gyro + visual odometry speed - Light Green. Gyro + wheel encoder speed - Medium Red. Note the multiple three point turns.

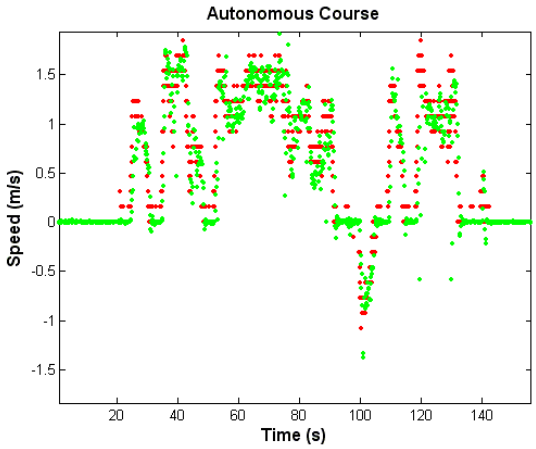

Figure 12: Speed estimates from visual odometry (Light Green) and wheel encoders (Dark Red) for the automonous run. Note that the wheel encoder data is notchy because the throttle input is notchy.

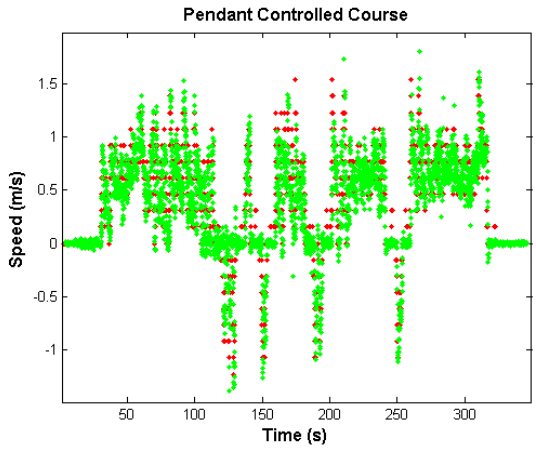

Figure 13: Speed estimates from visual odometry (Light Green) and wheel encoders (Dark Red) for the remote controlled run.

5.6 Further Results

One of the benefits of the visual odometry is that it can correct for wheel slip that affects the wheel encoders during a GPS fallout. Figure 14 shows the results of wheel slip on a muddy trail. Figure 15 shows the improved result of fusing the visual odometry with the other information. Another example that clearly demonstrates the visual odometry's advantage over wheel encoder is given in Figure 16. This run was performed under the most extreme conditions. High speed approach to a hill was followed by several seconds of almost stationary wheel spin. In the general area where the test was performed the DGPS signal was not always available, which caused jumps in DIPS+IMU position. The same phenomenon can be seen intermittently also in Figure 18. The heading offset at the northernmost point of Figure 16 may be due to the wrong absolute heading from GPS being combined with IMU between updates. In general we feel that in the absolute sense, GPS may have given a better position at several points along the way, but it is clear that the vision+IMU results are smoother and still close to the true position.

Figure 17 shows results from a remote controlled run in a parking lot. Figures 18 and 19 show more results from autonomous runs in the Yuma desert.

6. Summary and Conclusions

We presented a system for real-time ego-motion estimation of a single camera or stereo rig. We concentrated on results with stereo cameras mounted on an autonomous ground vehicle. Coherent and surprisingly accurate real-time results for hundreds of meters of driving were demonstrated, based only on visual input from relatively small field of view cameras. The results were evaluated quantitatively by comparing with a highly accurate integrated INS/DGPS navigation system, proving the speed, low latency, accuracy and robustness of our results.

One of the most important differences between our system and prior art is that we in real-time perform full RANSAC estimations for each motion estimate, which allows us to include more point tracks in the estimation, even tracks that may be outliers. Using more of the available information ultimately allows more reliable estimates.

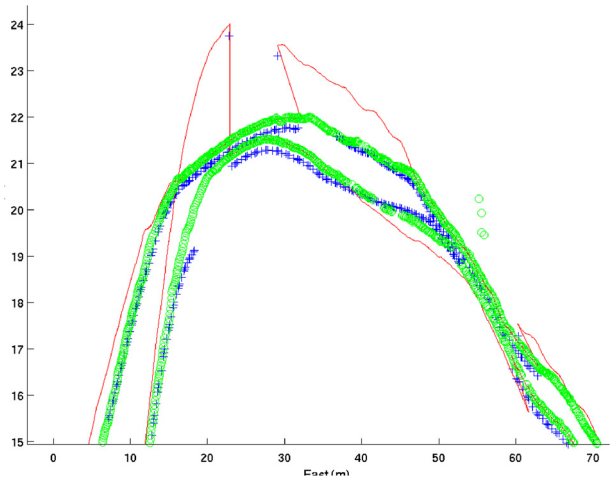

Figure 14: An example of the effect of wheel slip without visual odometry or GPS. DGPS - Dark Blue plus signs. Wheel encoders fused with IMU - Thin Red. Visual odometry - Thick Green. Note the incorrect overshoots from the wheel encoders. The motion of the vehicle was left to right in the bottom arc and right to left in the top arc.

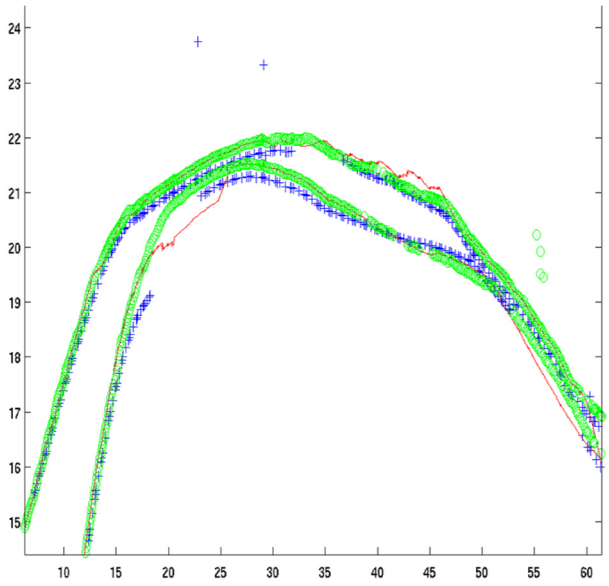

Figure 15: Results corrected by adding the visual odometry. DGPS - Dark Blue plus signs. Wheel encoders fused with visual odometry - Thin Red. Visual odometry - Thick Green.

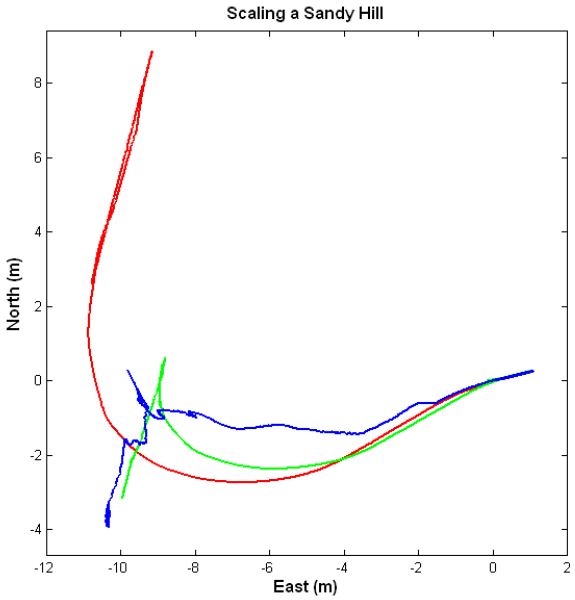

Figure 16: In this example, the vehicle tries to climb a 50 degree unstable sandy slope, gets stuck, but eventually makes it to the top of the hill. DGPS - Dark Blue. Wheel encoders fused with visual odometry - Medium Red. Visual odometry - Light Green. Note that the wheel encoder produces a completely erroneous position, while visual odometry is both smooth and close to the truth. The run was remote controlled.

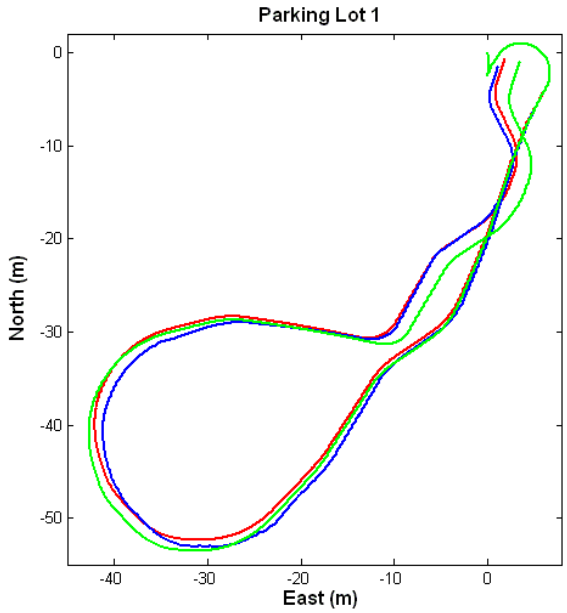

Figure 17: Remote controlled run in a parking lot. DGPS - Dark Blue. Wheel encoders fused with gyro - Medium Red. Visual odometry fused with gyro - Light Green.

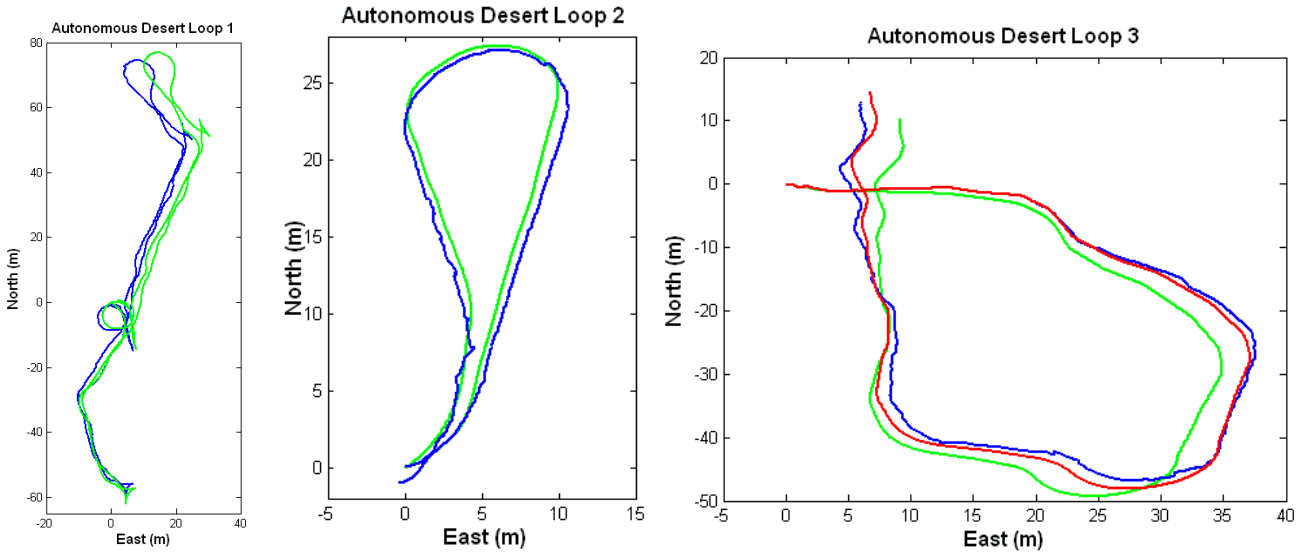

Figure 18: Autonomous runs in the desert. DGPS - Dark Blue. Visual odometry fused with gyro - Light Green. The last figure also shows wheel encoders fused with gyro in Medium Red.

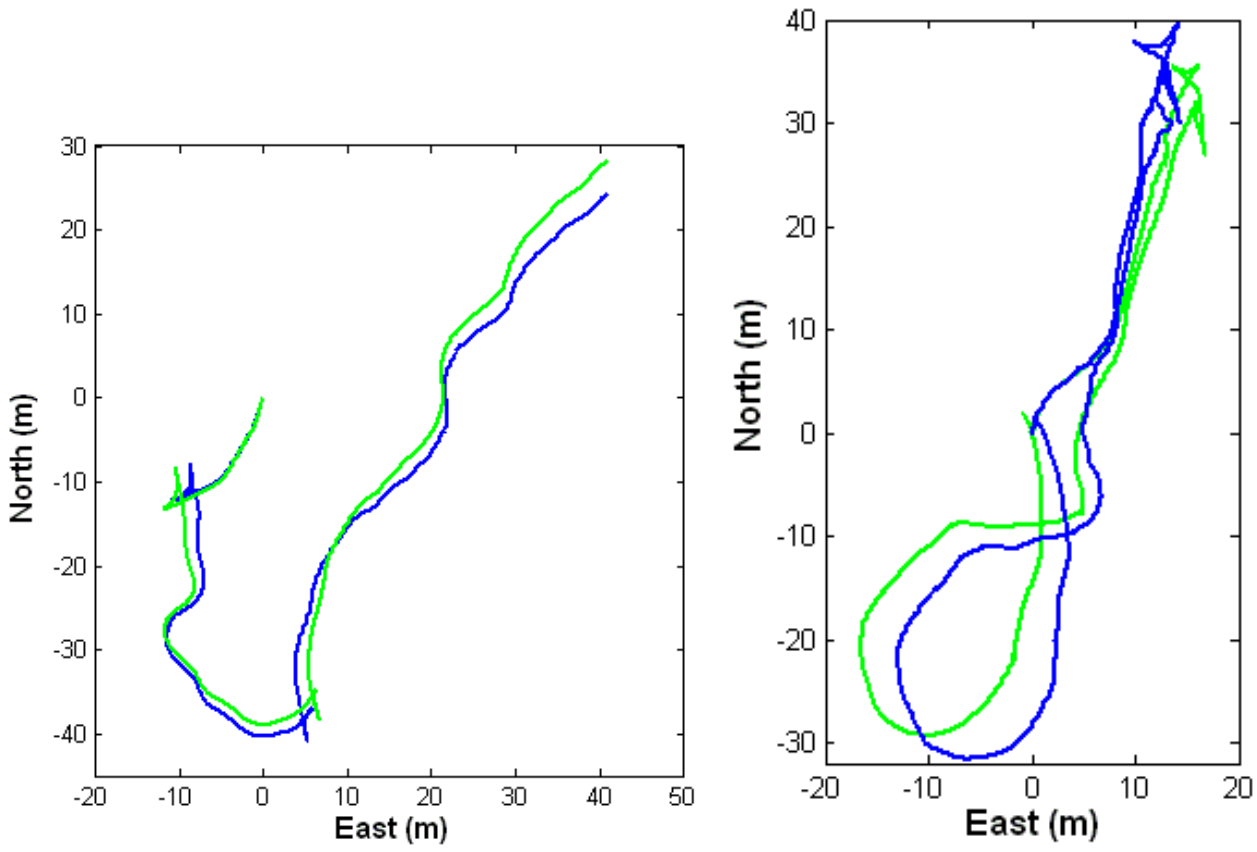

Figure 19: Autonomous runs in the desert. DGPS - Dark Blue. Visual odometry fused with gyro - Light Green.

In the future, we believe that uncertainty information regarding the visual estimates will become increasingly important. The ultimate motion estimation system should use information from all available sources, which can not efficiently be achieved without having a notion of which estimates are reliable. As the error distribution for visual estimates is highly non-Gaussian, covariance and Kalman filtering, although an important step, will not be sufficient. Sampling methods allowing multimodal distributions may therefore become the methods of choice. We regard RANSAC as highly targeted sampling, and our system as a good step towards robust and widely used visual odometry.

A. Pseudo-Code for Feature Detection

```
Sweep 1: Let i,j indicate beginning of line
for(c=0;c<128;c++){
      Ix=(img[i][j+c-1]-img[i][j+c+1])>>1;
      Iy=(img[i-1][j+c]-img[i+1][j+c])>>1;
      dxx[c]=Ix*Ix;
      dxx[c+128]=Ix*Iy;
      dxx[c+256]=Iy*Iy;}

Sweep 2: Let d0-d4 be pointers to the five buffer lines
for(c=0;c<128;c++){
   dd=d2[c];
   g[c]=d0[c]+(d1[c]<<2)+(dd<<2)+(dd<<1)+(d3[c]<<2)+d4[c];
   dd=d2[c+128];
   g[c+128]=d0[c+128]+(d1[c+128]<<2)+(dd<<2)+(dd<<1)+(d3[c+128]<<2)+d4[c+128];
   dd=d2[c+256];
   g[c+256]=d0[c+256]+(d1[c+256]<<2)+(dd<<2)+(dd<<1)+(d3[c+256]<<2)+d4[c+256];}

Sweep 3: for(c=0;c<124;c++)
```

This work was sponsored by DARPA, under contract 'Perception for Off-Road Mobility (PerceptOR)' (contract number MDA972-01-9-0016). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies or endorsements, expressed or implied, of the U.S. Government.

```
      g[c]=g[c]+(g[c+1]<<2)+(g[c+2]<<2)+(g[c+2]<<1)+(g[c+3]<<2)+g[c+4];

Sweep 4: for(c=0;c<124;c++){
     Gxx=gxx[c];
     Gxy=gxy[c];
     Gyy=gyy[c];
     d=Gxx*Gyy-Gxy*Gxy;
     t=Gxx+Gyy;
     s[c]=d-k*t*t;}

Non-max Suppression: for(i=top;i<=bottom;i++) for(j=left;j<=right;j++){
   v=s[i][j];
   if(
    v>s[i-2][j-2]&&v>s[i-2][j-1]&&v>s[i-2][j]&&v>s[i-2][j+1]&&v>s[i-2][j+2]&&
    v>s[i-1][j-2]&&v>s[i-1][j-1]&&v>s[i-1][j]&&v>s[i-1][j+1]&&v>s[i-1][j+2]&&
    v>s[  i][j-2]&&v>s[  i][j-1]&&             v>s[  i][j+1]&&v>s[  i][j+2]&&
    v>s[i+1][j-2]&&v>s[i+1][j-1]&&v>s[i+1][j]&&v>s[i+1][j+1]&&v>s[i+1][j+2]&&
    v>s[i+2][j-2]&&v>s[i+2][j-1]&&v>s[i+2][j]&&v>s[i+2][j+1]&&v>s[i+2][j+2])
      Declare Feature}
```

References

[1] 2d3 Ltd. Boujou, http://www.2d3.com.

[2] Chiuso, A., Favaro P., Jin, H. and Soatto, S., 2000, 3-D Motion and Structure Causally Integrated over Time: Implementation, *Proc. European Conference on Computer Vision, LNCS* 1842:735-750, Springer Verlag.

[3] Corke, P., Strelow, D. and Singh, S., September 2004, Omnidirectional visual odometry for a planetary rover, *IROS*.

[4] Davison, A., 2003, Real-Time Simultaneous Localization and Mapping with a Single Camera, *IEEE International Conference on Computer Vision*, pp. 1403-1410.

[5] Fischler, M. and Bolles, R., 1981, Random Sample Consensus: a Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography, *Commun. Assoc. Comp. Mach.*, 24:381-395.

[6] Fitzgibbon, A. and Zisserman, A., 1998, Automatic Camera Recovery for Closed or Open Image Sequences, *Proc. European Conference on Computer Vision*, pp. 311-326.

[7] Haralick, R., Lee, C., Ottenberg, K. and Nölle, M., 1994, Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Problem, *International Journal of Computer Vision*, 13(3):331-356.

[8] Harris, C. and Stephens, M., 1988, A Combined Corner and Edge Detector, *Proc. Fourth Alvey Vision Conference*, pp.147-151.

[9] Hartley, R. and Zisserman, A., 2000, *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN 0-521-62304-9.

[10] Helmick, D., Cheng, Y., Clouse, D. and Matthies, L., March 2004, Path following using visual odometry for a Mars rover in high-slip environments, *IEEE Aerospace Conference*.

[11] Intel Corp., DuLong, C., Gutman, M., Julier, M. and Keith, M., 1997, *The Complete Guide to MMX Technology*, McGraw-Hill, ISBN 0-070-06192-0.

[12] Jin, H., Favaro, P. and Soatto, S., 2000, Real-time 3-D motion and structure from point features: a front-end system for vision-based control and interaction, *Proc. IEEE Intl. Conf. on Computer Vision and Pattern Recognition* pp. 778-779.

[13] Jung, I. and Lacroix, S., 2005, Simultaneous localization and mapping with stereovision, *Robotics Research: the 11th International Symposium, P. Dario and R. Chatila (eds.)*, Springer.

[14] Kriegman, D., Triendl, E. and Binford, T., Dec. 1989, Stereo vision and navigation in buildings for mobile robots, *IEEE Trans. on Robotics and Automation*, 5(6).

[15] Lacroix, S., Mallet, A., Chatila, R. and Gallo, L., 1999, Rover self localization in planetary-like environments, *5th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (i-SAIRAS)*.

[16] Mallet, A., Lacroix S. and Gallo, L., 2000, Position estimation in outdoor environments using pixel tracking and stereovision, *ICRA*.

[17] Matthies, L. and Shafer, S., June 1987, Error modeling in stereo navigation, *IEEE Journal of Robotics and Automation*, RA-3(3).

[18] Moravec, H., 1980, Obstacle avoidance and navigation in the real world by a seeing robot rover, *PhD thesis*, *Stanford University*.

[19] McCarthy, C. and Barnes, N., 2003, Performance of temporal filters for optical flow estimation in mobile robot corridor centring and visual odometry, *Australasian Conference on Robotics and Automation*.

[20] Maimone, M., Johnson, A., Willson, R. and Matthies, L., June 2004, Autonomous navigation results from the Mars Exploration Rover (MER) mission, *ISER*.

[21] Nistér, D., 2000, Reconstruction From Uncalibrated Sequences with a Hierarchy of Trifocal Tensors, *Proc. European Conference on Computer Vision*, Volume 1, pp. 649-663.

[22] Nistér, D., March 2001, *Automatic dense reconstruction from uncalibrated video sequences*, PhD Thesis, Royal Institute of Technology KTH, ISBN 91-7283-053-0.

[23] Nistér, D., 2003, An Efficient Solution to the Five-Point Relative Pose Problem, *IEEE Conference on Computer Vision and Pattern Recognition*, Volume 2, pp. 195-202.

[24] Nistér, D., 2003, Preemptive RANSAC for Live Structure and Motion Estimation, *IEEE International Conference on Computer Vision*, pp. 199-206.

[25] Nistér, D., June 2004, An Efficient Solution to the Five-Point Relative Pose Problem, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756-770.

[26] Nistér, D., 2004, A Minimal Solution to the Generalised 3-Point Pose Problem, *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, Volume 1, pages 560-567.

[27] Oliensis, J. and Genc, Y., 1999, New Algorithms for Two-Frame Structure from Motion, *Proc. International Conference on Computer Vision*, pp. 737-744.

[28] Olson, C., Matthies, L., Schoppers, M. and Maimone, M., June 2003, Rover Navigation Using Stereo Ego-motion, *Robotics and Autonomous Systems*, 43(4):215-229.

[29] Pollefeys, M., Verbiest, F. and Van Gool, L., 2002, Surviving Dominant Planes in Uncalibrated Structure and Motion Recovery, *Proc. European Conference on Computer Vision*, Volume 2, pp. 837-851.

[30] Press, W., Teukolsky, S., Vetterling, W. and Flannery, B., 1988, *Numerical recipes in C*, Cambridge University Press, ISBN 0-521-43108-5.

[31] Schmid, C., Mohr, R. and Bauckhage, C., 2000, Evaluation of Interest Point Detectors, *International Journal of Computer Vision*, 37(2), 151-172.

[32] Shi, J. and Tomasi, C., 1994, Good Features to Track, *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593-600.

[33] Srinivasan, M., Zhang, S., Altwein, M. and Tautz, J., 2000, Honeybee navigation: nature and calibration of the 'odometer', *Science*, vol. 287, issue 5454, Feb 4.

[34] Torr, P. and Zisserman, A., 1997, Robust Parameterization and Computation of the Trifocal Tensor, *Image and Vision Computing*, 15:591-605.

[35] Zhang, Z., Faugeras, O. and Ayache, N., Dec. 1988, Analysis of a sequence of stereo scenes containing multiple moving objects using rigidity constraints, *Proc. 2nd ICCV*.