

# Fusion of Range and Vision for Real-Time Motion Estimation

Julian Kolodko, Ljubo Vlacic

**Abstract**—In this paper we introduce a motion estimation algorithm that fuses visual and range data to give an unambiguous estimate of the velocity of objects visible to a camera and range sensor. Dynamic scale space is used to avoid temporal aliasing and a novel robust estimator based on Least Trimmed Squares is used to smooth results between boundaries established using range data. Simulation results (from a specially developed simulation environment) and experimental results (from an FPGA based implementation of our algorithm) show our approach gives accurate motion estimates.

## I. INTRODUCTION

FOR people, navigating in a complex dynamic environment is second nature, however engineers are yet to develop a system that can reliably perform the same task. Our work aims to bring such a system closer to reality by providing a means of measuring relative motion in the environment.

This paper considered the problem of developing a simple motion estimation algorithm that fuses both range and visual data to calculate accurate motion information [1]. Use of range data allows disambiguation of velocity information since, to a camera, a slow moving nearby object may have an visual motion equal to a distant, fast moving object. Velocity is calculated using a simple iterative averaging estimator rather than more complex robust estimators often used in such applications [2]. To avoid problems of temporal aliasing and consequently to allow real-time performance over a wide range of conditions, we developed a dynamic scale space approach. The algorithm has been implemented in FPGA hardware, creating a compact motion sensing system. We present both simulation and experimental results here. In the

experimental system, a FUGA 15D camera is used. Because a suitable range sensor was not available at the time of writing, simulated range information is used.

## II. SYSTEM ASSUMPTIONS

Our target application – autonomous vehicles – allows certain assumptions to be made regarding the environment and places certain restrictions on available information. From the point of view of an autonomous vehicle, occlusion and limited field of view make it impossible to have global knowledge of motion in the surrounding environment. In response to this we adopt a rule of thumb adapted from personal experience traveling in large cities. This rule is to simply avoid the closest object in front. Later we show that this assumption directly leads us to a dynamic scale space approach that allows temporal aliasing to be avoided efficiently by choosing a single scale.

For practical reasons, in this work we assume that the autonomous vehicles (equipped our sensor) either move on a *smooth* ground plane, or actively stabilize the input data (range & vision)<sup>1</sup>. In any case, it is assumed that the vehicle moves on a ground plane (i.e. in two dimensions), which allows us to make the following assumptions:

- We need only estimate horizontal motion. Assuming all objects remain in contact with the ground plane, vertical visual motion can only occur when objects approach or recede from the camera. However, the information regarding the rate of approach is equally encoded in the horizontal component of motion, so why calculate it twice? In this work no attempt is made to determine rate of approach using visual data. Rate of approach can be more accurately determined using range data.
- Since only horizontal motion is of interest, only narrow images are needed. Smaller images reduce the computational load and allow us to make further

Manuscript received December 10, 2003. This work was made possible by the generous donation of hardware by Fraunhofer.AiS.

Julian Kolodko is with the Intelligent Control Systems Laboratory, Griffith University, Brisbane, QLD, 4111, AUSTRALIA. (e-mail: j.kolodko@griffith.edu.au)

Ljubo Vlacic is the director of the Intelligent Control Systems Laboratory, Griffith University, Brisbane, QLD, 4111, AUSTRALIA. (Phone: +61 7 3875 5024, fax: +61 7 3875 5198, e-mail: l.vlacic@griffith.edu.au).

<sup>1</sup> Primitive experiments have shown our sensor can tolerate a degree of vertical (pitch) motion noise however this is yet to be quantified. For this reason we currently assume a smooth ground plane.

assumptions regarding the validity of range data. Since only a 1D range scan is available, we must assume range is valid over an entire image column. This is more likely to be true for a narrow image.

Furthermore, we assume all “objects” in the environment move rigidly and move with a known maximum velocity (so that the sensor can be tuned to avoid temporal aliasing). Our sensor will not provide any output if an object approaches too closely for temporal aliasing to be avoided. Since the sensor is configured based on assumed environmental dynamics, this situation is considered an emergency in which special purpose avoidance and harm minimization systems would take over.

### III. REAL TIME AND DYNAMIC SCALE SPACE

Motion estimation algorithms based on image derivatives fail if visual motion of more than one pixel per frame is present in an image sequence. Greater motions generally result in temporal aliasing [1] and this in turn leads to errors in the derivatives of image intensity, which are used to estimate motion. In this work we define real-time as “a processing regime that avoids temporal aliasing”. Using a pinhole camera model and a rigid ground plane motion assumption (eq 3a), it is simple to show that the minimum frame rate required to avoid temporal aliasing is given by:

$$\text{Frame Rate} = \frac{X+f}{\zeta} \frac{1}{S} \text{ fps} \quad \text{Eq 1}$$

In this equation, the first term is related to the cameras intrinsic parameters and the second term is related to environmental dynamics. In the first term,  $f$  is the camera focal length,  $\zeta$  is the cameras’ pixel pitch and  $X$  is half the image sensor width. The parameter  $S$  in the second term is referred to as the safety margin (measured in seconds) and is defined as  $Z_{\min}/V_{\max}$ , - the ratio of minimum allowable object depth to maximum object velocity. The safety margin mimics the practice of drivers allowing a two second buffer to the vehicle in front. As such, using the safety margin gives the frame rate required to avoid temporal aliasing in the *worst case* - the required frame rate will fall as  $Z$  increases. Given the parameters of the camera used in this work and a safety margin of 2.5sec leads to a minimum required frame rate of 256fps to avoid temporal aliasing.

To reduce this frame rate requirement and to allow measurement of a wider range of velocities it is common to use a scale space approach. This technique builds an image pyramid with the original image at the bottom and successively subsampled images at each higher step (or scale). This subsampling allows higher velocities to be measured at higher scales since the pixel pitch  $\zeta$  is

effectively increased. A full implementation of scale space is not practical for a real time system since the overhead of generating the image pyramid and passing motion estimates between levels (usually via image warping [2]) is large.

To overcome this, our “large city” assumption is invoked. Remembering that the frame rate requirement increases with decreasing range. We only concern ourselves with the nearest obstacle since this is the object most likely to cause a collision in the short term - a scale is chosen so that the nearest object will not cause temporal aliasing. This will result in smaller visual motion for more distant objects, but ensures the algorithm can cope with the greater threat of the nearby object. If we allow up to 4 times subsampling, the effective pixel pitch becomes  $16\zeta$  and the frame requirement drops to 16fps, which is easily achievable.

### IV. MOTION ESTIMATION

Our motion estimation algorithm is based on a fusion of visual and range data. This is achieved at two levels. First, we directly fuse visual motion with range data to disambiguate object velocity. Secondly, range data is used to segment the visual scene so that our initial velocity estimate can be optimally smoothed for improved velocity accuracy. In gradient based algorithms, visual motion is estimated using Horn and Schunks’ optical flow constraint equation (eq 2) [3] or some variation

$$I_x u + I_y v + I_t = 0 \quad \text{Eq 2}$$

Here,  $I_x$ ,  $I_y$  and  $I_t$  are the spatial and temporal image derivatives and  $(u,v)$  is the visual velocity. In order to disambiguate this visual velocity we look to the equations of rigid body motion [4]. Eq 3 is a simplified form of these equations where we assume ground plane motion and no rotation.

$$\begin{aligned} u &= \psi \left( -\frac{1}{Z} (xT_x + fT_x) \right) \\ v &= \psi \left( -\frac{yT_z}{Z} \right) \end{aligned} \quad \text{Eq 3}$$

Here,  $f$  is the camera focal length (metres),  $(x,y)$  is the position on the image sensor (pixels),  $Z$  is depth (metres) and  $\psi$  is a constant that converts from m/sec to pixels/frame and equals  $1/(\text{frame rate} * \zeta)$ .  $T_x$  and  $T_z$  are the velocities along the  $X$  axis (perpendicular to the cameras’ optical axis) and the  $Z$  axis (along the cameras’ optical axis) respectively. Rather than directly substituting  $(u,v)$  from eq 3 into eq 2, we first make an assumption that, rather than having a single 2D image, we have multiple 1D images (one image for each row of the original image). This eliminates  $I_y$  from eq 2 and gives us the following constraint equation.

$$-\frac{\Psi}{Z} U_x I_x + I_T = 0 \quad \text{Eq 4}$$

We now use the symbol  $U_x$  because the value we are estimating is not purely the left/right motion, but encodes the projection of motion onto the X axis.

Our algorithm is embedded in a dynamic scale space and proceeds as follows:

1. Solve eq 4 for  $U_x$  at each pixel in our set of 1D images.
2. Robustly obtain a single estimate of  $U_x$  by combining corresponding values of  $U_x$  from each 1D image.
3. Segment range information. In this work we search for zero crossings in the second spatial derivative of range.
4. Smooth  $U_x$  by robustly finding the average value of  $U_x$  between the boundaries identified in the range segmentation.
5. Employ temporal integration by taking a weighted average of the current and previous estimate of  $U_x$ .

Traditionally motion estimation algorithms attempt to directly segment  $U_x$  however this is exceedingly difficult because  $U_x$  is very noisy (see middle of fig 1 for an example of  $U_x$  calculated using simulated visual and range data). By performing our second stage of data fusion and basing our smoothing of  $U_x$  on the segmentation of range data we obtain results far superior to those that can be achieved by smoothing  $U_x$  directly. Further more, temporal integration by averaging ensures consistency of results over time. To avoid the “momentum effect” of simple averaging where it takes some time for  $U_x$  to settle at its true value near object boundaries, we do not average at object boundaries. Thus the variance of the object velocity reduced without any oversmoothing.

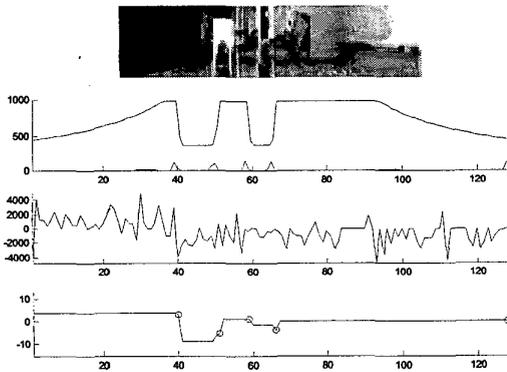


Figure 1. Example simulation result. Forward moving camera. Top shows the input visual information. Second shows the input range information and its 2<sup>nd</sup> derivative. 3<sup>rd</sup> shows  $U_x$  and bottom shows smoothed  $U_x$ .

Steps 2 and 4 of our algorithm require a robust average. What does this mean? When taking an average we are looking for the “usual value” for some parameter, but what if we have a data set as in figure 2. We immediately see that the short “peak” is probably caused by a process different to that which created the remaining data and should be excluded from our average calculation. But how do we reject this data automatically? This is the domain of robust estimators that attempt to give the best estimate (of the average in this case) by excluding “outlier” data points.

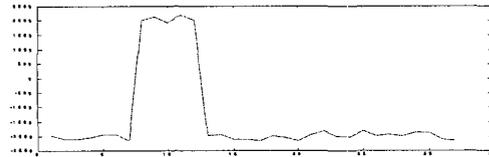


Fig 2. How would you find the average of this data?

An obvious solution in this case is to use a median instead of mean since the median is determined using a single value from the (ordered) data set. However what if the data could degenerate to something like figure 3? Here the mean would be the best estimate of the underlying process (assuming Gaussian noise) – the median would essentially be a random data point near the mean.

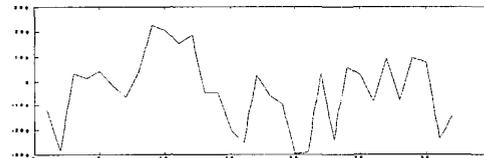


Fig 3. How would you find the average of this data?

Such situations can occur in motion estimation. Imagine we have an object close to the camera. This object will fill the entire vertical extent of the image so our motion estimation may look like figure 3. However, if an object is distant, it may only fill part of an image column. Thus, some of the values for  $U_x$  will correspond to the object and some to the background (perhaps like figure 2). In this case there are two processes – we wish to reject the “background” motion and only calculate the mean of the objects motion. If both cases can be present in a data set then neither a simple mean or median is a suitable estimator for the “average” of the data set.

A vast array of estimators exist that may solve this problem ranging from the M-Estimator to Least Median of Squares, RANSAC and others [5]. What all these estimators share is complexity. We have developed a simple iterative averaging estimator known as the LTSV (Least Trimmed Squared Variant) that is related to the Least Trimmed

Squares estimator [5] but requires no ordering or stochastic sampling (which is awkward in digital implementation)<sup>2</sup>. As all estimators, we begin by assuming that at least half the data is “good” (i.e. less than half the data elements are outliers) and proceed as follows:

```

thresh = abs(max possible input value)
current_mean = 0

for i=1 to number_iterations

    numValid=0;
    sum=0;

    for j=1 to n
        if current_mean-data(j) < thresh
            sum=sum+data(j);
            numValid=numValid+1;
        end if
    end for

    if numValid>=n/2
        threshold = thresh/alpha;
    else
        threshold = thresh*beta;
    end if

    if numValid!=0
        current_mean = sum/numValid;
    end if

end for

```

In the first iteration, LTSV includes all the input data in the sum, so the `current_mean` will be the mean of the entire data set. Note that this mean will naturally lie closer to the “majority good” population (inliers) than the “minority bad” population (outliers). The threshold for including data is reduced (by a factor of  $\alpha$ ) and then we recalculate the mean only including data that is closer to `current_mean` than the threshold. Eventually the threshold will be reduced sufficiently so that only the majority good data will be included in the mean. If the threshold is reduced too far (so that more than half the data is rejected), the threshold is increased slightly (by a factor of  $\beta$ ) and the mean calculated again. Eventually the LTSV estimate will settle to some optimal state where as close as possible to half the data is rejected. In the case where all the data is “good”, we will have a result that is close to the mean. It will never be exactly equal to the mean since the final result will essentially be the mean of a random subset of the data. This gives our estimator a relatively low efficiency (high variance), though this can be counteracted using a weighted least squares post processing step as suggested by Rousseeuw & Leroy [5].

<sup>2</sup> To the best of our knowledge this estimator has not appeared in the motion estimation literature. A search of the estimation literature also failed to find evidence of this estimator.

While further modifications to this algorithm are possible<sup>3</sup> this algorithm works quite well as is for our application. In fact our experiments show that it performs better than other estimators commonly used for motion estimation (M-Estimators such as the Lorentzian [2] and Truncated Quadratic [6]) both in computational complexity and bias though it has lower efficiency (higher variance). Figure 4 shows results of applying these estimators to 100 realizations of data similar to that of figure 2. When both the Lorentzian and our iterative estimator are on equal footing (same number of iterations etc) our estimator is far superior. Given 100 realisations of data illustrated in figure 3, we see that the variance of the LTSV estimator is relatively large (see figure 4), however, on average, it still outperforms the Lorentzian.

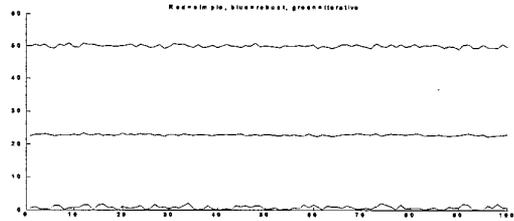


Figure 4. Comparison of estimators for figure 2 type data. Top is the error from the “true mean” for a simple average. Middle is the error for a robust estimator (Lorentzian M Estimator), bottom is the LTSV estimate. The LTSV and Lorentzian use same number of iterations.

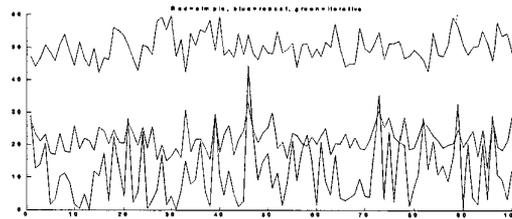


Figure 5. Comparison of estimators for figure 3 type data. Top is the error from the “true mean” for a simple average. Middle is the error for a robust estimator (Lorentzian M Estimator), and bottom is the LTSV estimate. The LTSV and Lorentzian use same number of iterations. Notice the relatively high variance of the LTSV estimate.

## VI. SIMULATION RESULTS

In order to ensure the correct operation of our algorithm before implementing it in hardware, it was tested using a simulation environment developed using MATLAB and a freeware raytracing program. This simulation environment generated realistic visual and range data suitable for evaluation of motion estimation algorithms. Since ground truth motion and optical flow is also provided by this

<sup>3</sup> For example,  $\alpha$  and  $\beta$  could be reduced each time more than half the data is rejected so that the result is converged upon more quickly.

simulation package, quantitative analysis of our motion estimation algorithm is possible. An example of the data generated by the simulation system and the operation of our algorithm is illustrated in fig 1.

Here we consider two sets of simulation experiments: one with a stationary camera, and the other with the camera moving. In both cases a single object moves through the environment parallel to the X axis and the velocity of this object is estimated. The experiments are repeated at a range of velocities. The results for the case of a stationary camera are shown in fig 6.

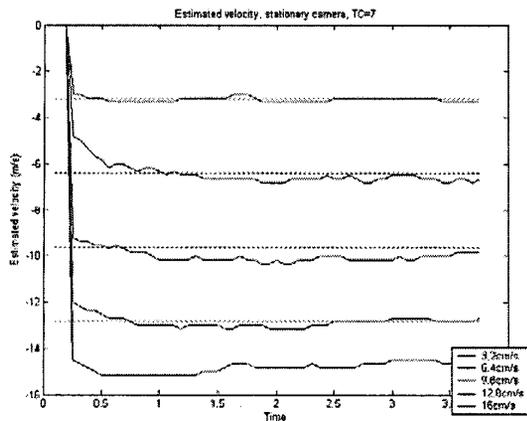


Figure 6. Estimated object velocity for stationary camera. Solid lines indicate estimated velocity while dotted lines indicate true velocity.

In fig. 6 solid lines indicate estimated velocity while dotted lines indicate expected velocity. One can clearly see that for a stationary camera, our algorithm gives excellent motion estimates for velocities up to about 12cm/s. The underestimation of higher motion estimates is caused by our exclusion of any motion estimate larger than a maximum determined using vehicle dynamics and aliasing limits [1].

Fig 7 illustrates results for a moving camera. In this case the camera moves forward along the Z axis with the same speed that the target object moves along the X axis. Since the projected motion will be larger towards the edge of the image (see eq 3) the expected velocity (dotted lines in fig. 7) is not a constant, but depends on the objects position in the image frame. Once again we see excellent motion estimates for low velocities (up to 9.6cm/s) above which exclusion of large results again causes underestimation. Excessively large estimates are excluded to add a degree of robustness to false motion caused by shadows [1] however this clearly causes problems. Current work focuses on resolving this issue, and initial experiments show that thresholding rather than excluding motion estimates significantly reduces the degree of underestimation.

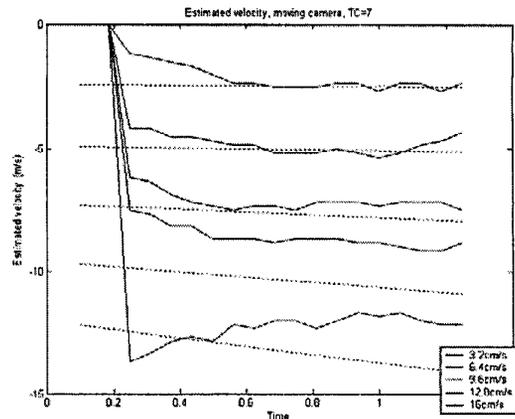


Figure 7. Estimated object velocity for stationary camera. Solid lines indicate estimated velocity while dotted lines indicate true velocity.

## VII. PRACTICAL IMPLEMENTATION

To illustrate the correct operation of our approach in the real world, we have implemented it using FPGA technology. This has allowed us to build a “system in a chip” where all interface logic, glue logic and processing logic is contained in a single chip. This effectively allows for a single chip sensing solution however our prototype is based on the LSP SignalMaster/GatesMaster platforms [7]. We use the FUGA 15D CMOS camera [8] since its simple RAM like interface removes the need for dedicate video capture systems. At the time of writing, a range sensor was not available to us hence range data was simulated using the Sharc DSP available on the SignalMaster board. Our experimental setup is illustrated in figure 8.

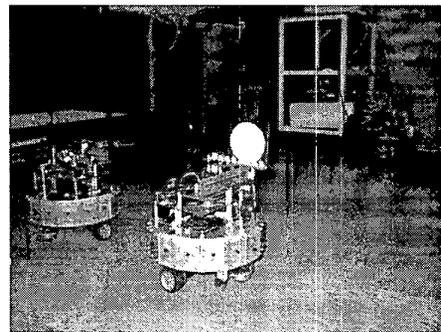


Figure 8. An experiment in progress. The camera is mounted on the ICSL CAMR [9] vehicle furthest from the viewer. The processing and visualization systems are on the trolley beside this vehicle.

Experimental results (see fig. 9) confirm that our algorithm gives good motion estimates in real environments even with a camera such as the FUGA 15D, which has relatively low contrast (due to its logarithmic intensity response) and relatively high noise. In fig. 9 the camera is stationary and

the tracked object moves to the left at 10cm/sec. Clearly the sensor gives excellent results under these conditions and this has been verified in a number of experiments [1].

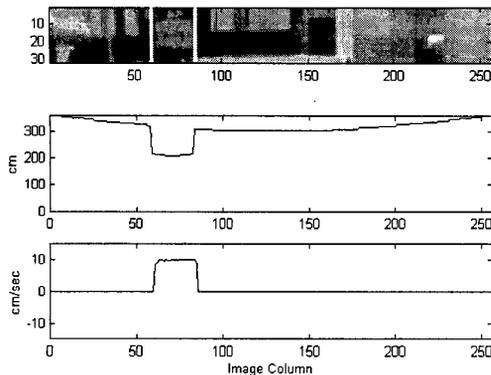


Figure 9. Experimental Results. Upper part of the diagram shows visual data (white lines indicate range discontinuities and contrast is increased for publication purposes), the central plot illustrates (simulated) range data and the lower plot is the estimated velocity

Figs. 10 & 11 show results for a moving camera at two different velocities. The high noise is a result of image noise and low detail in the logarithmic image however, in fig 10 (velocity = 3.85cm/sec) we see that over a range of experiments the motion estimate is close to the expected value. Fig 11 (velocity = 10cm/sec) we see that while motion estimate follows the expected trend, it is significantly underestimated. We believe that this is related to the exclusion/thresholding issue discussed earlier and is currently under investigation.

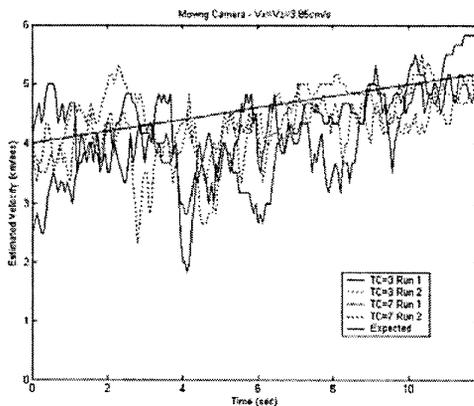


Figure 10. Velocity of tracked object when both object and camera move at 3.85cm/sec. The camera moves along the optical axis and the object moves perpendicular to the optical axis. Straight line indicates the expected motion estimate. TC is the temporal integration coefficient.

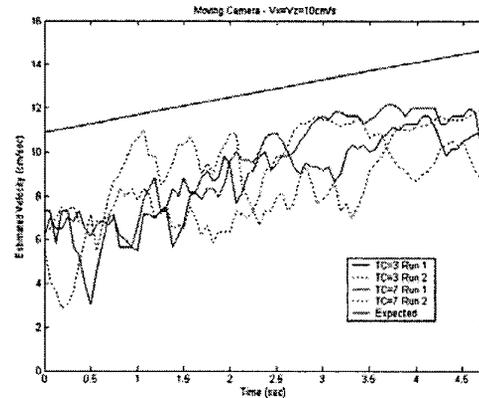


Figure 11. Velocity of tracked object when both object and camera move at 10cm/sec. The camera moves along the optical axis and the object moves perpendicular to the optical axis. Straight line indicates the expected motion estimate. TC is the temporal integration coefficient.

## VIII. CONCLUSION

In this paper, we have presented a new motion estimation algorithm that explicitly fuses both visual and range data in two ways to generate accurate, robust motion estimates. Robust average estimation is performed using a deterministic variant of the LTS algorithm that is simple, and does not require stochastic sampling or reordering of data. This paper also gives both simulation and experimental results showing that the algorithm provides accurate motion information under a range of conditions though issues remain to be resolved regarding noise reduction (preliminary results show this can be dealt with using an increased temporal integration coefficient) and underestimation (preliminary results show this is related to exclusion of excessively large motion estimates can that this can be resolved using thresholding rather than exclusion).

## REFERENCES

- [1] Kolodko J., Vlacic L., "Real Time Motion Estimation for Autonomous Navigation", PhD Thesis, Intelligent Control Systems Laboratory, Griffith University, 2004
- [2] Black M. J., "Robust Incremental Optical Flow", Ph.D. Thesis, Yale, 1992.
- [3] Horn B., Schunk B., "Determining Optical Flow", MIT Artificial Intelligence Lab, AI Memo 572, April 1980
- [4] Mitiche A., "Computational Analysis of Visual Motion", New York, Plenum Press, ISBN: 0-306-44786-X, 1994
- [5] Rousseeuw P. J., Leroy A. M., "Robust regression and outlier detection", John Wiley & Sons, ISBN 0-471-85233-3, 1987
- [6] Blake A., Zisserman A., "Visual Reconstruction", MIT Press, 1987. ISBN 0-262-02271-0
- [7] <http://www.lyrtech.com/>
- [8] <http://www.vector-international.be/>
- [9] Vlacic Lj., Engwirda A., Hitchings M., O'Sullivan Z., "Intelligent Autonomous Systems: Griffith University's Creation", Intelligent Autonomous Systems, IAS-5, Kakazu, Wada and Sato (Eds.), IOS Press, pp.53-60, 1998