Distributed optical flow navigation using Acromovi architecture

Patricio Nebot and Enric Cervera

Robotic Intelligence Lab
Department of Computer Science and Engineering
Univeritat Jaume I, Campus de Riu Sec
E-12071 Castellón de la Plana, Spain
Email: pnebot@icc.uji.es, ecervera@icc.uji.es

Abstract: Optical flow computation involves the extraction of a dense velocity field from an image sequence. The purpose of this work is to use the technique of optical flow so that a robot equipped with a color camera can navigate in a secure way through an indoor environment without collide with any obstacle. In order to implement such application, the Acromovi architecture has been used. Acromovi architecture is a distributed architecture that works as middleware layer between the robot architecture and the applications, which allows sharing the resources of each robot among all the team. This middleware is based on an agent-oriented approach.

Keywords: Mobile robots, Multiagent systems, Optical flow, Motion parameters, Obstacle avoidance.

1. Introduction

During years, the problem of processing image sequences for calculating the optical flow has been studied. The optical flow can be used in many applications, but recently it has been applied in mobile robot navigation [1][7].

The optical flow is the distribution of the apparent velocities of movement of the brightness pattern in an image, and arises from the relative movement of some objects and a viewer [3]. As the robot is moving around, although the environment is static, there is a relative movement between the objects and the camera onboard the robot.

The visual information coming from the camera of the robot is processed through the optical flow technique, which provides the information the robot needs to safely navigate in an unknown working-environment.

From the optical flow generated in this way the robot can get an inference on how far an object present in a scene is. This information is embedded in the time to crash (or time to collision) calculated from the optical flow field [8].

The purpose of this work is to use the optical flow technique to allow a mobile robot equipped with a color camera navigates in a secure way through an indoor environment without collide with any obstacle. It reacts to the environment by itself. The objective is that the robot be able to avoid obstacles in real-time.

So, the main objectives of this work include the use of vision libraries for capturing and processing image frames, performing all the calculations as fast as possible, to implement simpler algorithms to perform the optical flow and time-to-contact calculations, and to include all this aspects inside the Acromovi architecture. The first objective implies the use of libraries such as JMF for capturing images. The second objective is the most important to guarantee to the robot the capability of real-time response. The third objective is common in software development, and insures the accomplishment of the second objective. Finally, the fourth objective involves the use of the Acromovi architecture giving support for all the application.

Other important aspect is the reason for choosing to implement the behavior wander. This behaviour requires to calculate the optical-flow field over the entire image, and not exclusively over a part of it, being that much more time consuming. So, the behavior wander is suitable for evaluating the possibility of using optical flow to provide the sensorial information the robot needs in real-time.

The paper is organized as follows. Firstly, a brief introduction to the Acromovi architecture is done. This architecture gives all the support necessary for the implementation of the behaviour explained. Then, the most important aspects about the optical flow and time-to-contact theory are described. Following, it is shown how the application has been developed and how it works. Next, there is a brief description of the results obtained from the experiments. Finally, some conclusion and future lines of research are cited.

2. Description of the Acromovi architecture

As has been shown in previous works and papers [4][5], Acromovi architecture is a distributed architecture for programming and controlling a team of multiple heterogeneous mobile robots that are capable to cooperate between them and with people to achieve service tasks in daily environments.

This architecture has some peculiarities. It allows the reutilization of code, due to it use native components, by means of agent wrappers, providing the programmer with a set of tools for mobile robots. Also, it allows the sharing of the robots' resources among the team and an easy access to the robots' elements by the applications. Other two important characteristics of the Acromovi architecture are the scalability and the facility of use.

Acromovi architecture is a framework for application development by means of embedding agents and interfacing agent code with native low-level code. It addresses the implementation of resources sharing among all the group of robots. Cooperation among the robots is also made easier in order to achieve complex tasks in a coordinated way. Also, Acromovi architecture is a distributed architecture that works as a middleware of another global architecture for programming robots.

The embedded agents that constitute the Acromovi architecture work as interfaces between the applications and the physical elements of the robots. Some agents also make easier the handle of these elements and provide higher-level services.

As can be seen in Fig. 1, Acromovi architecture is a middleware layer between the robot architecture and the applications that allows the collaboration and cooperation of the robots in the team and allows sharing the resources of each robot among all the team. This middleware is based on an agent-oriented approach.

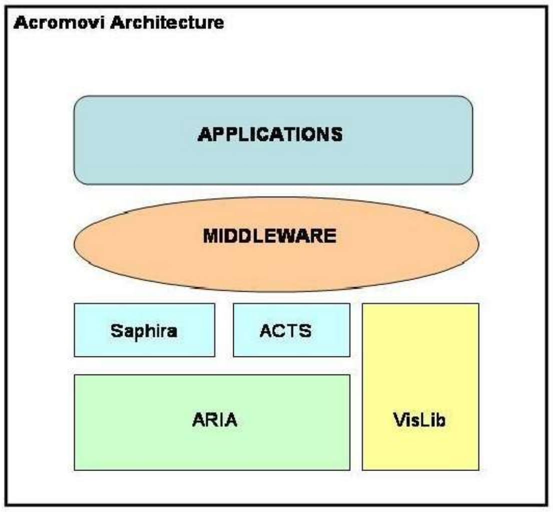


Fig. 1. General architecture diagram

Over the middleware layer, the applications layer is. These applications, to access to the components of the robots, must communicate with the agents of the middleware, which in turn access to the bottom layer that controls the robot.

This middleware level has been divided into two layers. In the lower layer, that can be seen in the Fig. 2, there are a set of components that access to the physical parts of the robots and other special components that offer services to the upper layer.

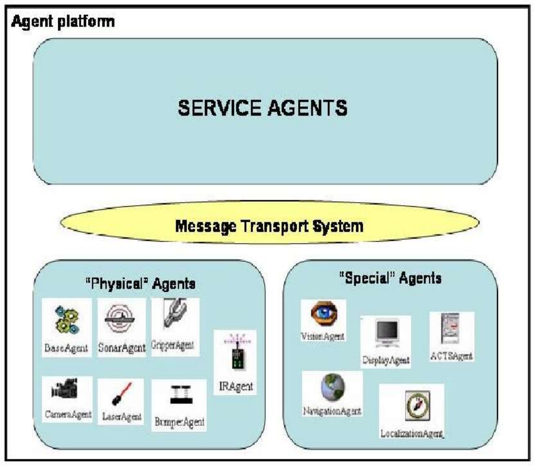


Fig. 2. The lower layer of the middleware

The upper layer, as it can be seen in Fig. 3, comprises a variety of embedded agents addressed to supervise and control the access to the agents of the lower layer.

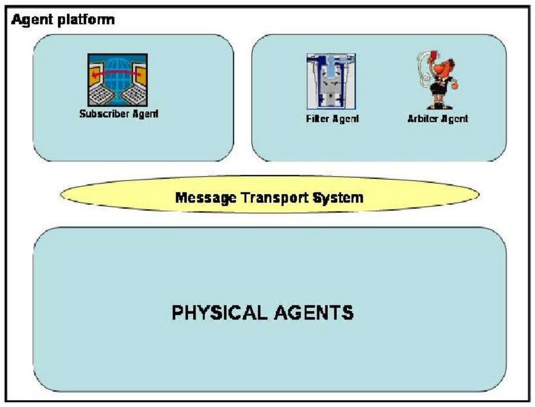


Fig. 3. The upper layer of the middleware

Other important characteristic of this middleware is due to the heterogeneous character of the team of robots. Because of this characteristic, there are different middleware layers for each robot, depending on the configuration of each robot of the team. These middleware layers are generated in execution time according to the elements that each robot has active at the moment of its activation.

Finally, another interesting characteristic of the overall architecture is the scalability. That is, once an application has been tested, and if it is useful for the entire architecture, it can be easily converted in a new agent of the upper layer of the middleware, increasing that our system to make applications more difficult and more interesting, following a bottom-up design.

3. Background Theory

(1) Optical Flow

Optical flow computation consists in extracting a dense velocity field from an image sequence assuming that intensity or color is conserved during the displacement. This result may be used by other applications like 3D reconstruction, time interpolation of image sequences using motion information, segmentation and tracking [6].

The optical-flow field used in this work is obtained by using formulas and constraints used by Horn and Schunck [3]. The equation that relates the change in image brightness at a point to the motion of the brightness pattern is:

$$E_x u + E_y v + E_t = 0 \qquad (1)$$

where,
E(x,y,t) represents the value of brightness at time t of a point with coordinates (x,y) in the image plane.
Ex, Ey, Et represent the partial derivatives of E with relation to x, y, t respectively.
u, v denote the components of optical flow along x and y respectively.

The values of Ex, Ey and Et can be calculated as follows:

$$\begin{aligned}
E_x &\approx 1/4\{E_{i,j+1,k}-E_{i,j,k}+E_{i+1,j+1,k}-E_{i+1,j,k}+E_{i,j+1,k+1}-E_{i,j,k+1}+E_{i+1,j+1,k+1}-E_{i+1,j,k+1}\}\\
E_y &\approx 1/4\{E_{i+1,j,k}-E_{i,j,k}+E_{i+1,j+1,k}-E_{i,j+1,k}+E_{i+1,j,k+1}-E_{i,j,k+1}+E_{i+1,j+1,k+1}-E_{i,j+1,k+1}\}\\
E_t &\approx 1/4\{E_{i,j,k+1}-E_{i,j,k}+E_{i+1,j,k+1}-E_{i+1,j,k}+E_{i,j+1,k+1}-E_{i,j+1,k}+E_{i+1,j+1,k+1}-E_{i+1,j+1,k}\}
\end{aligned} \qquad (2)$$

The iterative solution for the values of u and v is given by:

$$\begin{aligned}
u^{n+1} &= \bar{u}^n - E_x[E_x\bar{u}^n + E_y\bar{v}^n + E_t]/(\alpha^2 + E_x^2 + E_y^2)\\
v^{n+1} &= \bar{v}^n - E_y[E_x\bar{u}^n + E_y\bar{v}^n + E_t]/(\alpha^2 + E_x^2 + E_y^2)
\end{aligned} \qquad (3)$$

The values of the averages of u and v are approximated as follows:

$$\begin{aligned}
&\bar{u}_{i,j,k} = 1/6\{u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k}\}\\
&+1/12\{u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}\}\\
&\bar{v}_{i,j,k} = 1/6\{v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k}\}\\
&+1/12\{v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}\}
\end{aligned} \qquad (4)$$

(2) Time to Contact

A primary use of optical flow in robotics vision is collision detection, in particular time-to-contact computation. Using only optical measurements, and without knowing one's own velocity or distance from a surface, it is possible to determine when contact with a visible surface will be made [2].

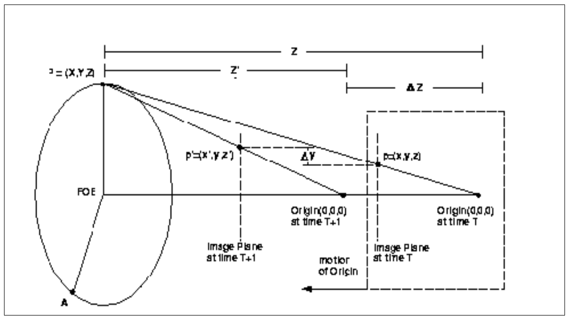


Fig.4. Optical geometry for time-to-contact.

A point of interest P at coordinates (X, Y, Z) is projected through the focus of projection centered at the origin of the coordinate system (0,0,0). P is fixed in physical space and does not move. The origin /focus of projection however moves forward with velocity dZ/dt. If the camera is facing the same direction as the direction of motion, then this direction is what is commonly known as the focus of expansion (FOE), since it is the point from which the optical flow diverges. The image plane is fixed at a distance z in front of origin.: for convenience, we set z=1 (The actual value of z depends on focal length of camera). P projects onto point p in this plane. As the image plane moves closer to P, the position of p in the image plane changes as well. Using equilateral triangle:

$$y/z = y/1 = Y/Z \qquad (5)$$

Differentiating with respect to time, we get:

$$y/\dot{y}=-(Z/\dot{Z})=\tau \qquad (6)$$

The quantity τ is known as the time-to-contact. Note that the left-hand side contains purely optical quantities, and that knowledge of τ does not provide any information about distance and velocity, but only of their ratio.

The results of using such approach are a meaningful reduction in the computation time and a less precise inference of the time to collision values. Experiments run, however, have shown that such values are precise enough for the sake of wandering.

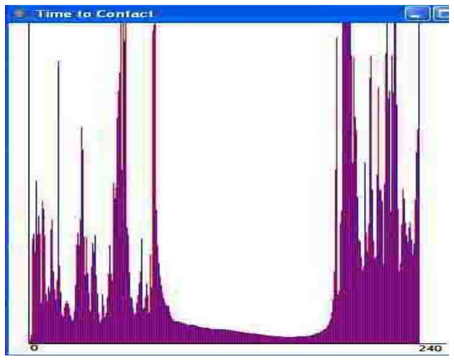


Fig.5. Example of time-to-contact graphic.

Based on the optical-flow field obtained as described above, the sensing subsystem delivers to the control subsystem information on the time to collision corresponding to distinct stripes in the visual field of the robot, that can be seen in Fig. 5. In addition to the vector of times to collision itself, the average value and the standard deviation corresponding to its elements are also calculated.

(3) The Control System
Regarding the time-to-contact results shown in Fig. 5 and the additional information associated to them, three different situations are considered here when implementing a system to control the heading angle of the robot. They are labeled “Imminent Collision”, “Side Obstacle” and “Normal” situations. Imminent Collision is characterized either when the average value and the standard deviation associated to the values of time to collision are both very small (meaning that a wide object is very close to the robot), or when an object for which the number of time-to-contact values inferior to the threshold adopted is greater than another threshold is detected in the middle of an image frame. In this case, the action the robot takes is to go back, to rotate in one direction and to move ahead again. However, if an object for which the number of time-to-contact values that are less than the threshold adopted is greater than another threshold is detected in the right side or in the left side of the visual field of the robot, the situation named Side Obstacle is characterized. In this case, the action the robot takes is to rotate in the opposite direction and to resume moving ahead. The Normal situation is characterized when no objects with high average optical flow magnitude are detected in the image frame. This means that all objects in the visual field of the robot are not close enough to deserve an abrupt manoeuvre. When this is the case, the robot moves straight ahead.

4. System Development

Three new agents have been designed for the development of this application, the Vision Agent, the OpticalFlow Agent, and the Control Agent. Each of these agents implements each of the modules described in previous sections. A diagram of the relation among these agents can be seen in Fig. 6.

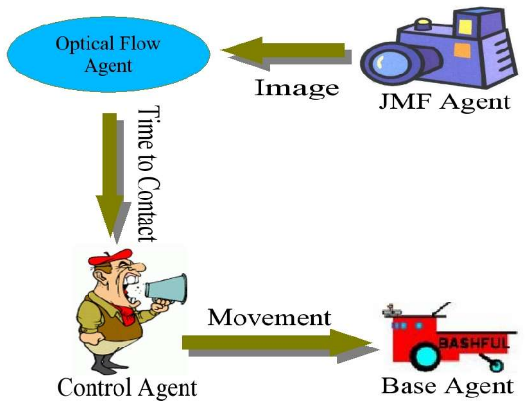


Fig.6. Interaction among the agents.

The first agent involved in the process is the JMF Agent, which is in charge of capturing the input image frames. The video from the camera on board the robot is captured using the Java Media Framework (JMF). The agent produce an image frame of size 240x180 pixels and provides grey scale data of each pixel at the rate of 5 frames per second. In Fig. 7 there is an example of frame captured by the JMF Agent.

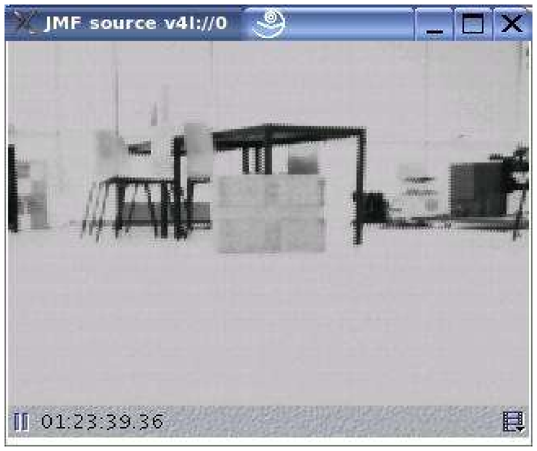
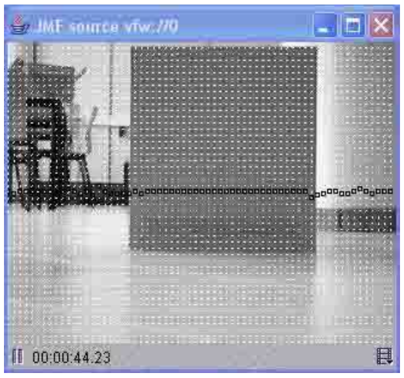


Fig.7. Captured image by the JMF Agent.

Then, this image is sent to the next agent in the process, the OpticalFlow Agent. This agent performs two different works. Firstly, it calculates the optical flow derived from the new image, using the formulas described above. Then, it calculates, from the optical flow, the time-to-contact vector, using also the formulas described above in the article. In Fig. 8, there is an example of the optical-flow field and the time-to-contact graphic.



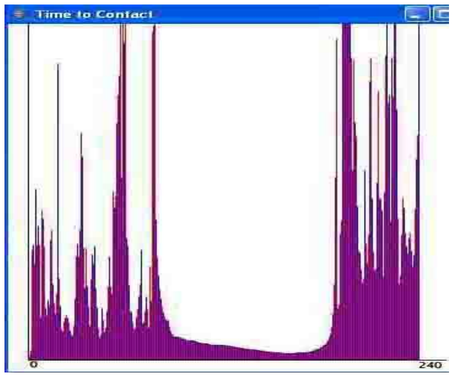


Fig.8. The optical flow field and the time to contact graphic.

Next, the OpticalFlow Agent sends the information corresponding to the time to contact to the Control Agent. This agent, depending on the graphic, can distinguish three distinct situations, as it has been said above. These situations are Imminent Collision, Side Obstacle, and Normal situations. Depending on the situation that the Control Agent recognizes, it sends different actions to the Base Agent, in order to the robot can avoid an obstacle or can follow its way.

The Imminent Collision situation is done when an object is in the middle of the image frame. That is, the elements 80 to 160 of the vector of times to collision have the values less than the others. In such cases, the actions the Control Agent sends to the Base Agent are to go back about 10 cm, to rotate 180 degrees and to move ahead again.

The Side Obstacle situation is done when an object is in the right or in the left side of the image frame. The obstacle is in the right side when the elements 160 to 240 have the smaller values, and it is in the left side when the elements 0 to 80 have the smaller values. In such cases, the manoeuvre adopted sent to the Base Agent is to rotate 15 degrees in the opposite direction and to resume moving ahead.

Here after turning 15 degrees to left or right and in case of imminent collision after the turning, the calculation of time-to-contact is elapsed during one image frame. This turning can produce an abrupt change in the image frame, and therefore it can cause getting an incorrect optical flow field or a very high magnitude which can result into low time-to-contact values and hence a wrong decision.

Finally, the Normal situation is executed when no object is in the image frame. In this case, all the elements of the vector of times to collision have similar values. So, the command sent to the Base Agent is to continue moving straight ahead.

However, in some situations an object can be either too close to the robot or moving towards it. In both cases the regions in the image frame considered that are detected as part of such object exhibit an optical flow vector of high magnitude. Thus, a simple comparison of this value with a threshold value allows detecting situations in which the robot is either about colliding to an object or close to an object moving towards it. The value of such threshold is experimentally adjusted, and limits how close to an object the robot can get before starting an evasive manoeuvre.

It is important to remark that the time spent in calculating the three parameters needed to the execution of the application (the average value and the standard deviation associated to the 240 values of time to collision used to build the graphic of Fig. 8 and the average value of the magnitude of the optical flow vectors associated to each object detected in the scene) is very small, not decreasing the capability of the robot to react in real-time.

Other important feature, due to the use of JADE and the mobility of agents that it allows, is the possibility to move one agent to another platform in order to improve the total time of computation. Thus, it is possible to move the OpticalFlow Agent to another platform faster that the mobile robot.

5. Experimental Results

In order to check the performance of the robot when wandering in its working-environment using the sensing and control subsystems discussed here the robot is programmed to wander around the lab, avoiding all the obstacles it detects.

The robot used for it is a Pioneer2 mobile robot with an onboard computer based on the Intel Celeron 650 MHz processor, having 508 Mbytes of RAM memory. A Logitech web camera is also available onboard the robot. The image capturing program in java grabs the images that are at most 320x240 pixels bitmaps at 5 fps in Linux platform.

An analysis of all the actions the robot has taken shows that it was effectively able to avoid the obstacles that appeared in its way, as expected, using the time-to-contact based sensorial information.

As mentioned above, the robot acquires image frames continuously at the interval of 200 ms, the calculation of the optical flow vectors plus the calculation of the new heading angle is compatible with the rate of acquisition of images, thus showing that the use of optical flow for this kind of sensing is suitable. In order to synchronize the calculation with the image acquisition time, an image of 240x180 pixels is used.

6. Conclusions and future work

As it has been said above, the purpose of this work was to use the technique of optical flow to make possible that a robot equipped with a color camera could navigate in a secure way through an indoor environment without colliding with any obstacle, using the Acromovi architecture.

New agents have been implemented based on the Acromovi architecture to make feasible that the robot can navigate in an environment using the optical flow technique. These agents have been implemented taking into account the limited computational setup available onboard the robot.

The experimental results have shown that the robot is effectively able to avoid any obstacle, in real-time, based only on the information from the optical-flow and the time-to-contact agents.

However, although the results are correct and sufficient for a real-time execution, it is possible to move the slower agents to another faster platform in order to improve the total time of computation.

Regarding the future work to do with the described system, first, sonars will be also used to distinguish those situations in which an object is too close to the robot and permits the robot to realize the evasive manoeuvre.

Another important improvement is to try to change from the wander behaviour to a most reliable navigation, following a specific path or trying to get to a specific point in the working-environment.

Also, it is important to try of reducing the time consumption by the calculation of the optical flow. For that, it can be used new methods faster than the used in this work. One possible option that implies a little variation over the original method is the described in [9]. So, with a few changes it is possible to get a system faster and capable to react in a more reliable way to changes in the environment.

Finally, this work can be extended to a team of robots that can cooperate so that a certain robot without the needed resources could navigate using the optical flow technique can do it. This can be possible thanks to one of the advantages of the Acromovi architecture, the possibility to share resources among the robots of the team. So, it is possible to move a resource from one robot to another to help the second robot to navigate in the environment, i.e. if one robot is not equipped with a camera, another robot equipped with one can help the first robot to go through a complicated environment where it is necessary more information that the one provided from the sonar.

Acknowledgments

This work has been partly funded by project GV05/137 from the Generalitat Valenciana.

References

[1] Carelli, R. et al. “Stable AGV Corridor Navigation with Fused Vision-Based Control Signals”, Proceedings of the 28th Annual Conference of IEEE Industry Electronics Society, 2002.
[2] Camus, T. “Calculating Time-to-Contact Using Real-Time Quantized Optical Flow”, National Institute of Standards and Technology NISTIR 5609, March 1995.
[3] Horn, B. K. P. and Schunk, B. G. “Determining Optical Flow.”, *Artificial Intelligence,* vol. 17, pp. 185-203, 1981.
[4] Nebot, P. and Cervera, E. “A Framework for the Development of Cooperative Robotic Applications”, Proceedings of 12th International Conference on Advanced Robotics (ICAR 2005).
[5] Nebot, P. and Cervera, E. “Agent-based Application Framework for Multiple Mobile Robots Cooperation”, Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2005).
[6] Quénot, G. “Computation of Optical Flow”. Available from: http://www-clips.imag.fr/mrim/User/georges.quenot/opflow/opflowE.html. Accessed: 2005-09-26.
[7] Sarcinelli-Filho, M. et al. “Using Optical Flow to Control Mobile Robot Navigation”, Proceedings of the 15th IFAC World Congress on Automatic Control, 2002.
[8] Sarcinelli-Filho, M. et al. “Optical Flow-Based Reactive Navigation of a Mobile Robot”, 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles, 2004.
[9] Tello, D.F. et al. “Optical Flow Calculation Using Data Fusion with Decentralized Information Filter”, Proceedings of IEEE International Conference on Robotics and Automation (ICRA 2005).