Distributed Robotic Vision as a Service

William Chamberlain[*], **Tom Drummond**[†], **Peter Corke**[*]
ARC Centre of Excellence for Robotic Vision
[*]Queensland University of Technology (QUT), Brisbane, Australia
[†]Monash University, Melbourne, Australia

Abstract

The vision sense of standalone robots is limited by line of sight and onboard camera capabilities, but processing video from remote cameras puts a high computational burden on robots. This paper describes the Distributed Robotic Vision Service, DRVS, which implements an on-demand distributed visual object detection service. Robots specify visual information requirements in terms of regions of interest and object detection algorithms. DRVS dynamically distributes the object detection computation to remote vision systems with processing capabilities, and the robots receive high-level object detection information. DRVS relieves robots of managing sensor discovery and reduces data transmission compared to image sharing models of distributed vision. Navigating a sensorless robot from remote vision systems is demonstrated in simulation as a proof of concept.

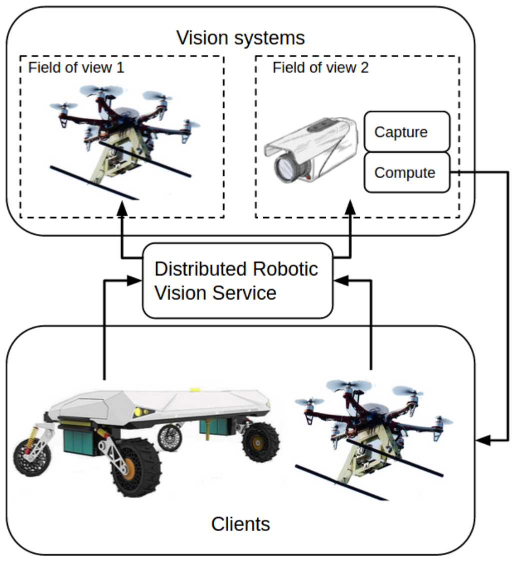

Figure 1: System overview: DRVS distributes requests from robots to vision systems with the required field of view, and robots receive processed visual information.

1 Introduction

Vision is a key sense for robotic autonomy and capability, currently limited by line of sight, onboard cameras, and onboard computing power. Robots can use visual information from remote cameras and from other robots to overcome the line-of-sight and camera capability limitations. Existing robot middleware and cloud solutions focus on passing raw image data and as a result place a high computational burden on the robot.

Robotic vision is driven by the information requirements of a robot's tasks. Middleware systems like the Robotic Operating System (ROS) [Quigley *et al.*, 2009] facilitate networking robotic system components to share data between robots and sensors. A disadvantage of this model for distributed vision is that the receiving robot must accept the data that is made available to it and process the data for its own purposes, which leaves the robot with less computing power for other functions.

The contribution of this paper is the Distributed Robotic Vision Service software framework, DRVS, which implements distributed visual object detection as a network service. Robots send object detection requests to DRVS specifying the type of object detection required, and DRVS distributes the requests to remote vision systems which perform the visual processing and send the processed information to the robot. The computational load is distributed to the vision systems, making distributed vision practical for robots with limited computation. The service is described in section 3.

[*]ARC Centre of Excellence for Robotic Vision, Queensland University of Technology, Brisbane, Australia. http://www.roboticvision.org {william.chamberlain, peter.corke}@qut.edu.au

[†]ARC Centre of Excellence for Robotic Vision, Monash University, Melbourne, Australia. http://www.roboticvision.org tom.drummond@monash.edu

This paper describes the architecture of DRVS and demonstrates navigating a robot using vision from off-board cameras in a simulated case study, showing that DRVS

1. allows a blind robot to navigate by making use of vision systems in its environment (section 4.1).
2. reduces the number of messages handled by robots by directing requests to appropriate vision systems (section 4.3).
3. manages sensor availability and coverage for robots (section 4.3).

DRVS focuses on object detection as a vision function which is widely applicable to many robot tasks, and is a step toward a full distributed robotic vision service.

2 Related Work

Networked robots have the potential to use millions of online vision systems which integrate camera sensors with processing power and network connectivity as sources of visual information.

Multi-view vision has been applied to create safe and effective robot behaviours in human environments. [Trautman and Krause, 2010] combine overhead vision and human behaviour prediction to plan efficient and safe paths through crowds of moving people, and minimal-disruption paths through office environments are planned in [Piyathilaka and Kodagoda, 2014]. These depend on distributed vision to gather positions and trajectories of people which are not visible to the robot's onboard cameras.

There is no commonly-used system for robotic distributed vision, and existing frameworks have disadvantages as services for distributed robotic vision. The Continuous Analysis of Many CAMeras system (CAM^2) [Kaseb *et al.*, 2014] demonstrates that low latency and high scalability are achievable for a distributed video processing architecture which processes video from distributed cameras on cloud infrastructure. CAM^2 is intended for human users and does not provide a programmatic interface for robotic distributed vision.

Robotic middleware systems such as ROS [Quigley *et al.*, 2009], YARP [Metta *et al.*, 2006], and Orca [Makarenko *et al.*, 2006] provide interoperability between robotic systems components on heterogeneous platforms. These middlewares do not have specific functions for distributed vision, but can provide the network communication mechanism for DRVS. Robots can offload high computation and memory requirements to remote processes through the robotics cloud platform provided by Rapyuta [Mohanarajah *et al.*, 2015], and can communicate with other robots through the cloud. Distributed vision is not a core function of Rapyuta and has to be implemented with custom solutions, whereas DRVS provides a simple interface for distributed vision.

Distributed sensor systems often leave responsibility for managing data about available sensors and data feeds to the robot, rather than providing an abstraction which let the robot controller focus on the robot's core tasks. The SensorCloud system provides standardised open specification for robots to discover sensors and stored data in the robot's region of interest [D'Este *et al.*, 2013]. SensorCloud provides current data feeds or historic records from the open Sense-T sensor data network, so robots cannot make request for sensors to gather data on demand for real-time tasks.

Wireless communication between mobile robots and remote cameras is limited by bandwidth and contention, and by time and computing costs [Kassir *et al.*, 2015]. Channel contention with even moderate concurrency in wireless communication has been demonstrated to have increased latency for robotic vision using web services [Blake *et al.*, 2011] and to prevent effective communication between mobile robots [Santos *et al.*, 2010], but there is less contention with smaller messages and lower total throughput. Rapyuta has high bandwidth in the cloud layer, but requires application-specific optimisations for some cooperative robotic tasks [Mohanarajah *et al.*, 2015]. Distributed vision systems become more effective as they incorporate more robots and cameras, and reducing message size by exchanging higher-level information will mitigate the channel contention.

Robots can use distributed vision to achieve safer and more effective behaviour, and for behaviour not possible using only onboard cameras. Wireless communication limits the bandwidth available for sending distributed vision data to mobile robots, and existing robotic software frameworks do not provide a distributed vision service which abstracts sensor configuration and delivers visual data on demand.

3 The Distributed Robotic Vision Service Framework

Robots use DRVS to obtain visual information from remote vision systems. The DRVS Server links robots to vision systems as shown in Figure 2. This section details the design and implementation of the DRVS architecture, API, messages, and distributed vision process.

Robots send object detection requests to the DRVS Server. An object detection request specifies the robot's region of interest, the algorithm and parameters that vision systems should use to process the visual information, and the network address on which the robot will listen for object detection response messages.

The DRVS Server then forwards object detection requests to vision systems. The DRVS Server maintains a registry of vision systems and their fields of view. When

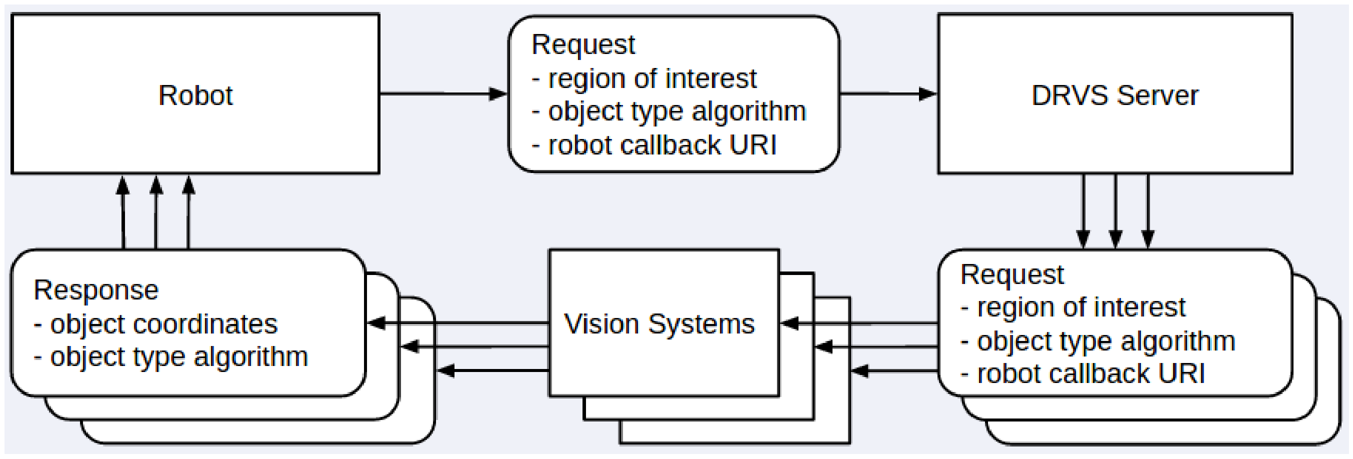

Figure 2: DRVS request and response processes.

it receives an object detection request from a robot the DRVS Server searches the registry for vision systems with fields of view that overlap the robot's region of interest, and forwards the object detection request onto each of these vision systems (Algorithm 1).

Vision systems in DRVS combine a camera sensor with processing and networking capabilities. The vision systems receive object detection requests from the DRVS Server, and capture and process visual information to detect objects using the algorithm specified in the request. DRVS is flexible for robots to specify their object detection requirements, so DRVS does not prescribe predefined object definitions. Vision systems parse the *algorithm* parameter of object detection requests to determine which object detection algorithm and parameters to apply. If the vision system detects one or more objects it sends the object locations to the robot at the network address specified in the object detection request. Finally, the robot receives object detection responses from vision systems and parses the *algorithm* parameter in each response to determine which object type the vision system has detected.

DRVS uses a global coordinate system. Vision systems are registered in the global coordinate system and report object detections as object vertices or bounding boxes in global coordinates. Robots specify their regions of interest to the DRVS Server in the same coordinate system.

Algorithm 1 DRVS Server - matching regions of interest to vision systems

```
1: for vision system in registry list do
2:     if field of view overlaps request location then
3:         send DRVS request to vision system
4:     end if
5: end for
```

Robots do not need to look up or discover vision systems to handle their object detection requests through DRVS. Vision systems are registered with the DRVS Server, and the DRVS Server determines which vision systems can handle each request. The DRVS Server is the only network address that robots use to make an object detection request, as the robot does not initiate communication with the vision systems. Changes to the available set of vision systems are therefore transparent to the robot.

3.1 Communication

DRVS performs distributed vision by communicating object detection requirements and object detections between the system components in a process that requires one-to-many messaging from the DRVS Server to vision systems.

Publish-subscribe messaging systems such as ROS topics allow a single component to publish data to many subscribing components. Client-server messaging systems such as ROS services have a one-to-one communication between the client and server in which the client makes a request and the server responds. DRVS is a framework for distributed vision which allows a single client to send its requirements to an unknown number of vision systems, with variable network and processing latencies for each response, and requires a different approach to either publish-subscribe or client-server.

DRVS uses ROS services for communication. DRVS does not depend on a specific aspect of ROS services. Other networked point-to-point communications would work as well, but ROS was selected for the communication layer for ease of use and integration, with implementations in C++, Python, Java, and LUA, and integration in MATLAB and VREP. DRVS can leverage the popularity of ROS for rapid integration with ROS-compatible robots and systems through its ROS API.

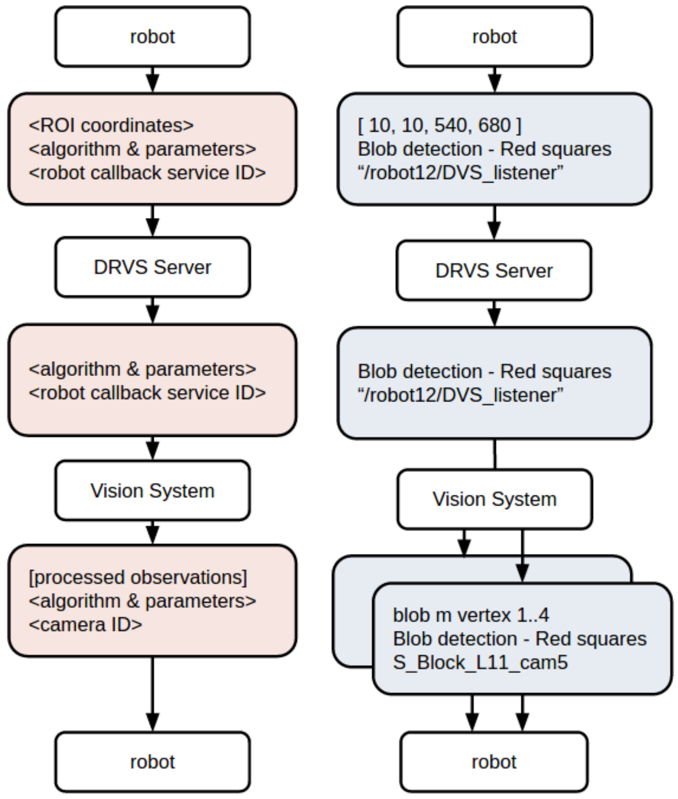

Figure 3: Message format definitions (left), and examples of specific message instances (right).

The DRVS request and response process consists of a sequence of ROS service calls over TCP/IP. The request comprises a service call from the robot to the DRVS Server, and a service call from the DRVS Server to each of several vision systems. The DRVS response process comprises one or more service calls from vision systems to the robot's callback service. Each ROS service call has its own request and response, but in DRVS only the request part carries information and the response is a simple acknowledgement which terminates the call. Each service call functions as a unidirectional message and DRVS uses sequences of these to create one-to-many and many-to-one messages.

The robot's perception component is a client of the DRVS Server, making requests for information about object types in regions of interest. Vision systems receive DRVS requests through a daemon process which waits for communication from the DRVS Server (Figure 4). The vision system parses the object type and robot callback address from the DRVS request, starts the visual information capture and processing, and finally sends object detection data to the robot callback address. The robot's perception component parses incoming DRVS responses from vision systems and updates program state to notify the robot controller of the objects detected.

3.2 API

The DRVS API is focused on object boundary detection in global coordinates, with a flexible object description. The robot uses DRVS through the *DetectObjectBoundary* and *CallbackCoordinates* services.
DetectObjectBoundary defines the DRVS request interface and has as parameters the
- object detection algorithm and parameters as a string
- region of interest bounding box coordinates
- robot *CallbackCoordinates* ROS service name

CallbackCoordinates defines the DRVS response interface and has as parameters
- the object detection algorithm and parameters
- an array of object detection coordinates

RegisterVisionService defines vision system registration interface and has as parameters the
- vision system ROS name for *DetectObjectBoundary*

ROS service names are unique identifiers for each ROS service instance, analogous to web service URIs.

Vision systems integrate into DRVS to provide distributed vision services by implementing the *DetectObjectBoundary* interface to accept DRVS requests, and the *CallbackCoordinates* interface to call back to the DRVS robot client with object detection data.

Robots can use DRVS services by implementing the *DetectObjectBoundary* interface to make requests to the DRVS Server, and implementing the *CallbackCoordinates* interface to receive information from DRVS vision systems.

3.3 The DRVS Server

The DRVS Server is a central standalone process with two roles

1. mapping the regions of interest of robot requests to vision system observable areas and forwarding DRVS requests to the vision systems
2. managing the vision system registry and registering vision systems

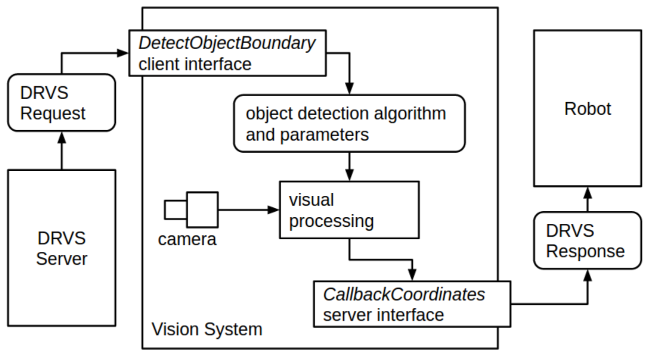

Figure 4: Vision System components and control flow during visual processing.

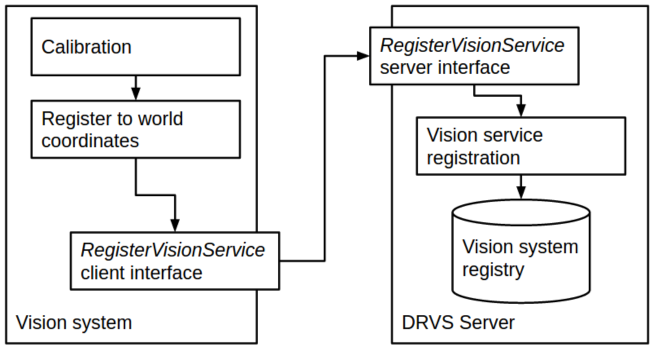
Calibration
Register to world coordinates
RegisterVisionService client interface
Vision system
RegisterVisionService server interface
Vision service registration
Vision system registry
DRVS Server

Figure 5: Vision system registration with the DRVS Server.

The DRVS Server maintains the registry of vision systems used in the DRSV request process. A registry entry contains the coordinates for the bounding box of the vision system's field of view and the unique ROS service name of the vision system.

Vision systems that implement *RegisterVisionService* (Figure 5) can register with the DRVS Server.

If a vision system is repositioned or moves such that its observable area changes, the new observable area is registered with the DRVS Server. The DRVS Server will forward requests for the new observable area to the vision system without interruption.

3.4 The DRVS Distributed Vision Process

The vision process in DRVS is initiated by a robot making a request to the DRVS Server, which forwards the request to appropriate vision systems. The vision systems perform the visual processing specified in the request, and send the processed visual information back to the robot.

A robot requests object detection in a region of interest from the DRVS Server as shown in Figure 2 and Figure 3. The *algorithm* parameter of the request specifies the object detection algorithm and parameters to the vision system. The DRVS Server matches the region of interest specified in the request to the vision systems in the registry, and forwards the requests following the DRVS process. The vision systems parse the *algorithm* parameter into the algorithm and parameters parts, interpret the parameters based on the algorithm, and perform the corresponding visual processing.

4 Case Study

The following case study demonstrates the features of DRVS in three simulated scenarios, based on a mobile robot using DRVS as its sole source of information for localisation and navigation while attempting to reach its recharging station. These scenarios show that DRVS allows robots to access sensors that the robot does not carry, with on-demand processing in the remote vision system, and without the robot performing any visual processing on the returned data.

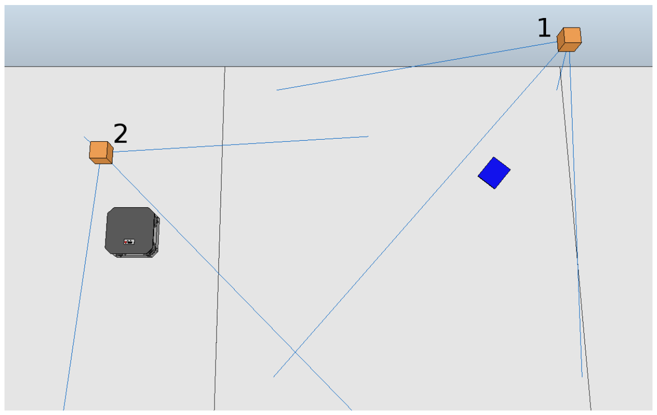

Figure 6: The blind robot case study. The robot is on the left in the field of view of vision system 2, and the charging station is on the right in the field of view of vision system 1.

For this case study the robot is assumed to be effectively blind (due to sensor failure, for example), and capable of using the global coordinate system used by DRVS and the vision systems. The case study is simulated in VREP, a commonly-used robotics and automation tool.

Figure 6 shows the environment for the location and planning scenario of this case study. The robot is on the left of the scene at (83,265) in the 2D world coordinate system, and the blue rectangle on the right at (590,355) represents the charging station. The vision systems are shown as yellow cubes, mounted high and viewing the scene from overhead, as is common for security camera views of rooms and corridors. Vision system 2 is centred above (205,205) with the robot's initial position in view, and vision system 1 is centred above (485,278) with the charging station in view; neither vision system can view both objects.

Object detection is simplified to polygon detection with colour segmentation for this case study. The robot specifies the class of object as a colour model which is parsed into the minimum and maximum RGB thresholds for colour segmentation. D* [Stentz, 1994] is used for path planning.

This case study does not address the issues caused by non-orthogonal objects occluding space, or unknown object size, which are limitations of the information obtainable from a single camera. The bounding boxes of all detected objects are reported as perceived by the cameras assuming projection on the ground plane.

Algorithm 2 Blind robot navigation

```
 1: procedure ROBOT PERCEPTION
 2:     call DRVS ($selfColourModel$)                ▷ robot
 3:     call DRVS ($stationColourModel$)             ▷ goal
 4:     repeat
 5:         if $goal$ and $start$ are known then
 6:             plan path
 7:         else
 8:             repeat requests
 9:         end if
10:     until timeout
11: end procedure
12: procedure ROBOT DRVS CALLBACK
13:     if $model = stationColourModel$ then
14:         $goal \leftarrow$ location
15:     else if $model = selfColourModel$ then
16:         $start \leftarrow$ location
17:     end if
18: end procedure
19: navigate from robot location to target location
```

4.1 Locatisation and Planning

Localisation is a key function for autonomy and this scenario demonstrates a sensorless robot using DRVS object detection to locate itself and its goal.

The robot perception component makes requests to the DRVS Server to locate the robot and charging station goal in the scene shown in Figure 6, as shown in Algorithm 2. The object description is a colour model.

The key points in this scenario are

1. all computation for visual processing is distributed to the remote vision systems. The robot cannot perform any visual processing in this case study because it does not receive any pixel data from the scene either directly or via DRVS.
2. the robot navigation system needs object boundary information to create a map for planning with D*, and DRVS provides distributed object detection services, so the robot uses the high-level information obtained through DRVS to carry out its tasks without onboard visual processing.
3. the robot's request determines the visual processing scheduled and performed by the vision systems.
4. the robot can determine both object location and object type from the vision system responses even though information may be received out-of-order by the robot-side *Callback Coordinates* server component.
5. the robot does not have to perform sensor discovery or be configured with information about available sensors.

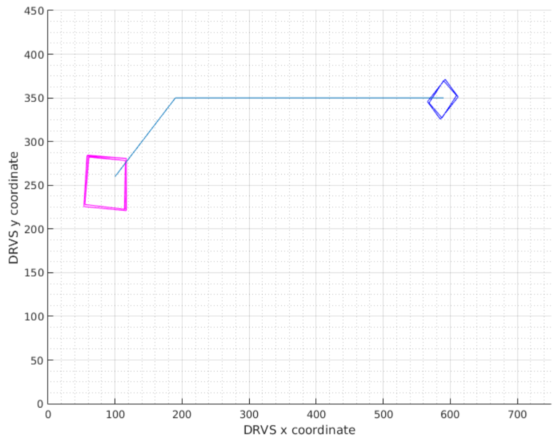

Figure 7: Objects detected with DRVS, and the planned path in the location and planning scenario 4.1. The robot is at (90,260) and the goal at (590,350).

4.2 Static and Dynamic Obstacles

Scenario 4.1 is extended to the robot localising itself, its goal, and static and dynamic obstacles. The robot perception component uses DRVS to locate obstacles by making DRVS object detection requests with obstacle colour models as shown in Algorithm 3, and the robot plans and navigates a path to the goal (Figure 8).

The key point in this scenario is that the new parameters specified by the robot perception module change the visual processing carried out by the remote vision systems, rather than the vision system implementation and configuration being changed to accommodate the new object classes. This demonstrates that the visual processing is flexible and determined by the robot requirements. Algorithm 3 is an extension of Algorithm 2 with the additional object colour models and re-planning if the mapped obstacles change.

This scenario also demonstrates the robot controller interacting with DRVS to monitor the environment for changes during navigation, and to re-plan when the planned path to the charging station becomes obstructed.

For a dynamic environment the robot makes repeated object detection requests during navigation and can re-plan its path in response to changes in obstacle locations. In this scenario an obstacle moves to obstruct the planned path (Figure 9), and the robot re-plans to find a path around the obstruction (Figure 10).

The dynamic obstacle detection and path re-planning are shown in Figure 10. The initial path is in blue and passes between the static obstacles along y=260. The robot starts at (80,300), and is at (280,360) when it detects the dynamic obstacle and the path is re-planned.

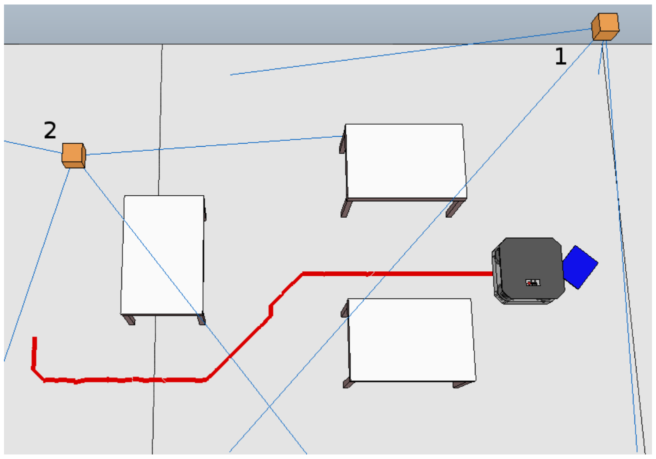

Figure 8: Navigation between static obstacles. The robot trajectory to the charging station is shown in red.

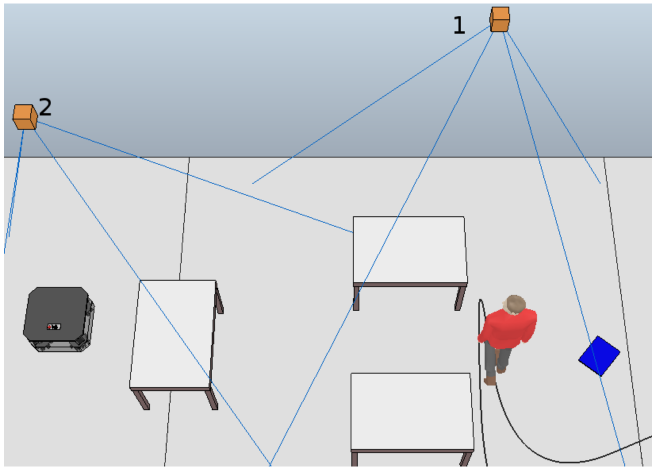

Figure 9: The dynamic environment scene with the red dynamic obstacle following its path across the robot's planned path.

The new path is shown in orange and passes around the obstacles along y=450.

4.3 Selectivity

This scenario demonstrates the DRVS Server mapping the DRVS requests to a subset of the available vision systems in response to the robot limiting the region of interest of its object detection requests.

The environment in this scenario is extended to a larger area with four vision systems (Figure 11). The robot addresses its initial requests to locate itself and the charging station in the entire floor area. It only receives responses from the vision systems that detect each object; vision system 2 for the robot and vision system 1 for the charging station. Vision systems 3 and 4 do not respond because they do not detect any objects of

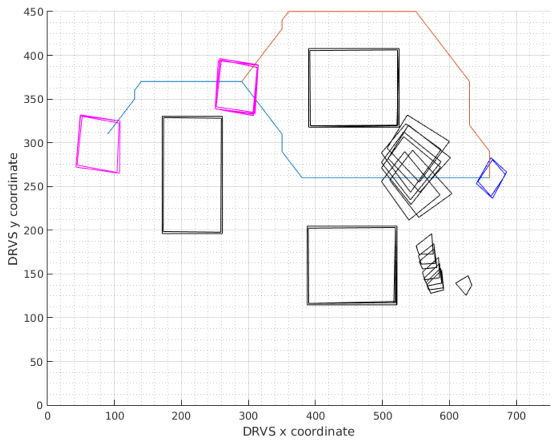

Figure 10: Path re-planning in response to a dynamic obstacle in scenario 4.2.

Algorithm 3 Blind robot navigation - with obstacles

```
 1: procedure ROBOT PERCEPTION
 2:     call DRVS ($selfColourModel$)                ▷ robot
 3:     call DRVS ($stationColourModel$)             ▷ goal
 4:     call DRVS ($obstacleColourModel$)            ▷ obstacles
 5:     call DRVS ($humanColourModel$)               ▷ obstacles
 6: end procedure
 7: procedure ROBOT DRVS CALLBACK
 8:     repeat
 9:         if $model = stationColourModel$ then
10:             $goal \leftarrow$ location
11:         else if $model = selfColourModel$ then
12:             $currentposition \leftarrow$ location
13:         else if $model = obstacleColourModel$ then
14:             $obstacle \leftarrow$ location
15:         else if $model = humanColourModel$ then
16:             $obstacle \leftarrow$ location
17:         end if
18:         if $obstacle$ map changed then
19:             re-plan path
20:         end if
21:     until timeout
22: end procedure
23: navigate from robot location to target location
```

the types requested. To reduce the size of the map used for planning, the robot only requests the locations of obstacles in an area bounded by the locations of the robot and the charging station. The DRVS Server matches this region of interest to vision systems 1 and 2, excluding vision systems 3 and 4 from the distributed vision request.

The key points in this scenario are that

1. the object detection request is only forwarded to

vision systems with fields of view overlapping the robot's region of interest

2. the robot only receives responses from vision systems which detect objects, reducing the number of messages it has to handle and the volume of network traffic.

3. the number, type, and observable areas of the vision systems are transparent to robots using DRVS. The DRVS Server is responsible for these considerations when it maps DRVS requests to vision systems, and the configuration information is maintained through the vision system registration process. The robot is relieved of responsibility for managing up-to-date vision system configuration information.

4.4 Computation and Data

The following comparison calculations are made between DRVS, a distributed vision service transmitting single raw images on demand from remote vision systems to robots, and a service transmitting compressed video. The calculations are for one round of vision with each robot in a scenario accessing each vision system for one image, or one second of video.

Each DRVS request and response consumes less than 0.2 kB. The following calculations assume distributing raw visual data from a VGA 800×600 pixel source with each image compressed to 70 kB using the ROS JPEG compressed image transport at 80% quality, and streaming video compressed to 112kBs^{-1} using ROS Theora. Optimal communication with no data loss is assumed.

Table 1 shows that DRVS has a significant advantage over distributed vision services which transmit image data to robots, due to the lower computational loads placed on robots and the total bandwidth required to transmit data. The robots receive significantly less data that they have to process, and the computation advantage will translate to significant power consumption advantages in mobile robots. The lower transmitted data makes DRVS less vulnerable to wireless contention, and further reduces power consumption.

Distribution type	Number of robots	Number of cameras	Total data transmitted (kB)	Data received per robot (kB)
DRVS	1	1	0.6	1
	1	20	8.2	4
	10	20	82	4
	10	50	202	10
compressed image	1	1	70.2	70
	1	20	1404	1400
	10	20	14040	1400
	10	50	35100	3500
compressed video (1s)	1	1	112	112
	1	20	2244	2240
	10	20	22440	2240
	10	50	56100	5600

Table 1: Comparison of transmitted data and data received per robot between DRVS and a system distributing image data for a single frame, or for one second.

5 Discussion

The case study scenarios presented in this paper use a simple object detection algorithm to focus on the design of DRVS. Specifying the object detection algorithm as a string parameter provides flexibility despite DRVS' simple service API, but the robots and vision systems using DRVS need a shared set of algorithm definitions to parse the parameter, and these algorithm definitions also form part of the API. Standardised representations such as those being developed by the IEEE-RAS Ontologies for Robotics and Automation working group [Schlenoff *et al.*, 2012], may be used to extend the object algorithms defined for DRVS.

Using ROS services limits DRVS to robots, vision systems, and DRVS Server connected to the same ROS network. DRVS could be extended to use non-ROS web services or a similar remote procedure call protocol, with each entity identified by its IP address-based service URIs, and the DRVS Server using a standardised public URL.

The current version of DRVS assumes a single coordinate system for all robots and vision systems. This is a constraint that will be relaxed in future work.

6 Conclusion

DRVS is a step toward a full distributed vision service, and toward robots which can use information from multiple views to overcome the line-of-sight limits on robotic vision, without prohibitive computational burden on the

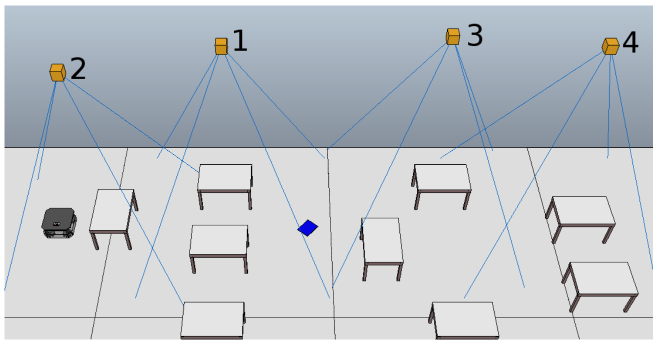

Figure 11: The selective vision system scenario. Vision systems 1 and 2 have the charging station and robot in view, while the fields of view of vision systems 3 and 4 are not on the robot's path.

robot, and without overwhelming wireless communications by streaming raw visual data.

DRVS implements a service for robots to specify their object detection requirements, and the DRVS Server communicates the requirements to remote vision systems which perform the object detection processing on demand. Computation is distributed to the remote vision systems and the robot receives only the processed high-level information, reducing the quantity of data transmitted and also reducing the processing load on the robot. Discovery of remote vision systems is handled dynamically by the service and is transparent to the robot.

The system is demonstrated in simulation, navigating a blind robot in scenarios with static and dynamic obstacles, and with selection of vision systems with appropriate fields of view.

Acknowledgments

This research was supported by the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016).

References

[Blake *et al.*, 2011] M. Brian Blake, Sekou L. Remy, Yi Wei, and Ayanna M. Howard. Robots on the web. *Robotics & Automation Magazine, IEEE*, 18(2):33–43, 2011.

[D'Este *et al.*, 2013] Claire D'Este, Chris Sharman, Ritaban Dutta, Ahsan Morshed, Jessica Lethbridge, Andrew Terhorst, Ben Howell. Sustainability, Scalability, and Sensor Discovery with Cloud Robotics. In *Proceedings of the Australian Conference on Robotics and Automation*, Sydney, Australia, December 2013.

[Kaseb *et al.*, 2014] Ahmed S Kaseb, Everett Berry, Youngsol Koh, Anup Mohan, Wenyi Chen, He Li, Yung-Hsiang Lu, and Edward J Delp. A system for large-scale analysis of distributed cameras. In *IEEE Global Conference on Signal and Information Processing*, 2014.

[Kassir *et al.*, 2015] Abdallah Kassir, Robert Fitch, and Salah Sukkarieh. Communication-aware information gathering with dynamic information flow. *The International Journal of Robotics Research*, 34(2):173–200, December 2015.

[Makarenko *et al.*, 2006] Alexei Makarenko, Alex Brooks, and Tobias Kaupp. Orca: Components for robotics. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 163–168, 2006.

[Metta *et al.*, 2006] Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. YARP: yet another robot platform. *International Journal on Advanced Robotics Systems*, 3(1):43–48, 2006.

[Mohanarajah *et al.*, 2015] Gajamohan Mohanarajah, Dominique Hunziker, Raffaello D'Andrea, and Markus Waibel. Rapyuta: A Cloud Robotics Platform. *Automation Science and Engineering, IEEE Transactions on*, 12(2):481–493, April 2015.

[Piyathilaka and Kodagoda, 2014] Lasitha Piyathilaka and Sarath Kodagoda. Active visual object search using affordance-map in real world: A human-centric approach. In *Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on*, pages 1427–1432. IEEE, 2014.

[Quigley *et al.*, 2009] Morgan Quigley, Ken Conley, Brian Gerkey, Josh FAust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Mg. ROS: an open-source Robot Operating System. In *Proceedings of the Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA), 2009*, 3:5, 2009.

[Santos *et al.*, 2010] Frederico Santos, Luís Almeida, Luis Seabra Lopes, José Luís Azevedo, and M Bernardo Cunha. Communicating among robots in the robocup middle-size league. In *RoboCup 2009: Robot Soccer World Cup XIII*, pages 320–331. Springer, 2010.

[Schlenoff *et al.*, 2012] Craig Schlenoff, Edson Prestes, Raj Madhavan, Paulo Goncalves, Howard Li, Stephen Balakirsky, Thomas Kramer, and Emilio Miguelanez. An IEEE standard ontology for robotics and automation. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1337–1342. IEEE, 2012.

[Stentz, 1994] Anthony Stentz. The D* Algorithm for Real-Time Planning of Optimal Traverses. Technical report, DTIC Document, 1994.

[Tenorth *et al.*, 2013] Moritz Tenorth, Alexander Perzylo, Reinhard Lafrenz, and Michael Beetz. The RoboEarth language: Representing and exchanging knowledge about actions, objects, and environments. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3091–3095, 2013.

[Trautman and Krause, 2010] Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *The IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems (IROS 2010)* , pages 797–803. IEEE, 2010.