ATLANTIS PRESS

Journal of Robotics, Networking and Artificial Life
Vol. 5(3); December (2018), pp. 184–189
DOI: 10.2991/jrnal.2018.5.3.9; ISSN 2405-9021 print; 2352-6386 online
https://www.atlantis-press.com/journals/jrnal

Research Article

ROS-based Multiple Cameras Video Streaming for a Teleoperation Interface of a Crawler Robot

Ramil Safin[1], Roman Lavrenov[1], Edgar A. Martínez-García[2], Evgeni Magid[1,*]

[1]*Intelligent Robotic Systems Laboratory, Intelligent Robotics Department, Higher Institute for Information Technology and Intelligent Systems, Kazan Federal University, Kazan 420008, Russian Federation*
[2]*Robotics Laboratory, Department of Industrial Engineering and Manufacture, Institute of Engineering and Technology, Universidad Autónoma de Ciudad Juárez, Chihuahua 32310, Mexico*

ARTICLE INFO

Article History

Received 1 October 2018
Accepted 22 October 2018

Keywords

Video streaming
mobile robot
ROS package
real-time
camera calibration

ABSTRACT

Digital cameras are being widely used in robotics in 3D scene reconstruction, simultaneous localization and mapping (SLAM), and other applications, which often require real-time video stream from multiple cameras. In a case of a mobile robotic system with limited hardware capabilities performance issues may arise. In this work we present a real-time video streaming robot operating system (ROS) package for our crawler-type mobile robot Servosila Engineer, which has four on-board cameras. Compared to OpenCV based solution our package demonstrates less resource consumption.

1. INTRODUCTION

A variety of applications in robotics employ digital cameras for the purposes of 3D scene reconstruction, obstacle avoidance [1], object detection and tracking [2], visual simultaneous localization and mapping (SLAM) [3], path planning [4], human–robot interaction, teleoperation and in urban search and rescue operations [5], etc. Most of the up-to-date solutions require to use more than just a single on-board camera, for example, stereo SLAM with two monocular cameras [6] and multi-camera rig setup to achieve higher accuracy results.

Capturing video frames from a camera typically needs to be accomplished in real-time, or with an acceptably low latency. However, in case of mobile robotic systems performance issues may arise. Considering that a robot is equipped with multiple cameras, system resources consumption increases, which leads to the decreased robot operation time.

Moreover, cameras tend to distort captured images. Distortion influences the accuracy of measurements [7], e.g., straight lines become curved. Therefore, camera calibration is a necessary step in robot vision.

In our work we used a Russian crawler-type mobile robot Servosila Engineer [8]. It is equipped with four on-board cameras that are located on its head and has an application programming interface (API) for video streaming purposes. However, the manufacturer's API does not enable to capture and receive video frames from all cameras at once – only one camera is available at a time within the original graphical user interface. Furthermore, Servosila Engineer's cameras are featured with essential radial distortions, which could be removed only through a calibration procedure. Therefore, we developed a special video capturing robot operating system (ROS) package for the robot. It enables to stream video from all cameras of the robot simultaneously in real-time. Camera calibration was also accomplished in order to increase the accuracy of visual sensor-based algorithms, which will be implemented in the future.

2. SERVOSILA ENGINEER MOBILE ROBOT

In this work we used the Russian crawler-type mobile robot Servosila Engineer (Figure 1). It is designed to be use in urban search and rescue operations, demining, and as a research and education platform.

Vision system of the robot consists of four cameras that are located on the robotic head (Figure 2): a stereo pair and an optical zoom camera are on the front side of the robotic head and one monocular rear view camera (Table 1). Additionally, there is a torch on the front side of the head.

In its original configuration the robot operated under Ubuntu 14 operating system (OS), but it was upgraded to Ubuntu 16 OS in order to use an up-to-date ROS distribution, ROS Kinetic. Servosila Engineer's on-board computer has no dedicated graphics card and thus only an onboard Intel HD Graphics 4000 is available. The CPU is an Intel Core i7-3517UE @ 1.7–2.4 GHz × 2/4 (two cores, four threads). Robot has 4 GB DDR3 random access memory stick functioning at 1333 MHz.

* *Corresponding author. Email: evgeni.magid@gmail.com*

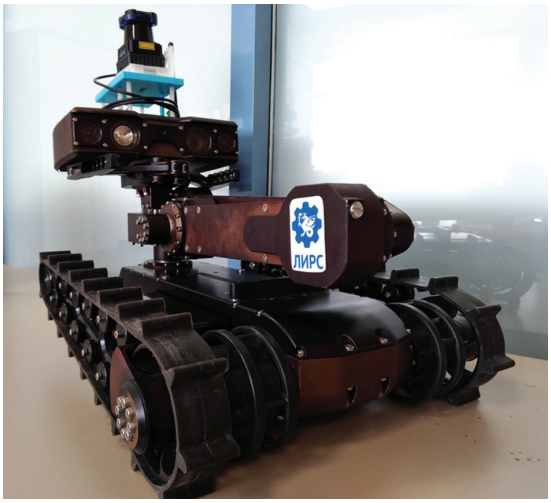

Figure 1 | Servosila Engineer crawler-type mobile robot (front view). Robotic head with four cameras and the LiDAR is attached to the 4DoF arm manipulator.

Figure 2 | Servosila Engineer's front and rear view cameras

Table 1 | Properties of the Servosila Engineer's cameras

Camera	Properties		
	Pixel format	Resolution	Frame rate
24× optical zoom	YUYV 4:2:2	1280 × 720	50
Rear view/stereo pair	8-bit Bayer	744 × 480 644 × 480	15/25/30/60

The manufacturer's original API enables to control the robot and receive video frames from one of its cameras at a time, i.e., only from a single active camera. Changing the active camera to the other camera could be done in runtime with a special command through the API.

3. RELATED WORK

In this section, we describe the development of a real-time video streaming ROS package using low-level Video4Linux2 API.

3.1. Video4Linux2 Video Capturing Layer

Video4Linux2 (V4L2) is an audio and video capturing API in Linux systems. It is especially effective in capturing video in real-time due to its close integration with the Linux kernel [9]. It was preinstalled within the OS by the robot's manufacturer and V4L2 was used for capturing video in the original API of the robot. Typical workflow with V4L2 is as follows (Figure 3):

- Open a device (a camera).
- Define device properties (variable values).
- Perform a video capturing loop (retrieve frames).
- Close the device (stop capturing).

Communication with a device (e.g., one of the robot's cameras) is executed by means of input/output (I/O) control requests – *ioctl*. It is a Linux system call, which allows to manage device parameters (frame rate, resolution, pixel format, etc.) and capture video frames.

Initially the device is needed to be opened for reading and writing operations. Further, device capabilities are queried in order to correctly setup the camera for streaming. Data from the device's internal buffer comes in the supported raw image format. Then device buffers are allocated in order to be able to exchange data between the device and an application (a program) using one of the streaming I/O methods. In this work we used memory mapping I/O approach, which enables to eliminate memory copies from the kernel space (device's driver) to the user space (our program). The program and the device's driver exchange pointers to captured video frames to make them directly available within the application. Querying buffers allow to retrieve all necessary information about requested buffers, e.g., size and location of the allocated memory, number of buffers. Capturing frames loop (streaming) is constituted of queue and dequeue operations (Figure 4).

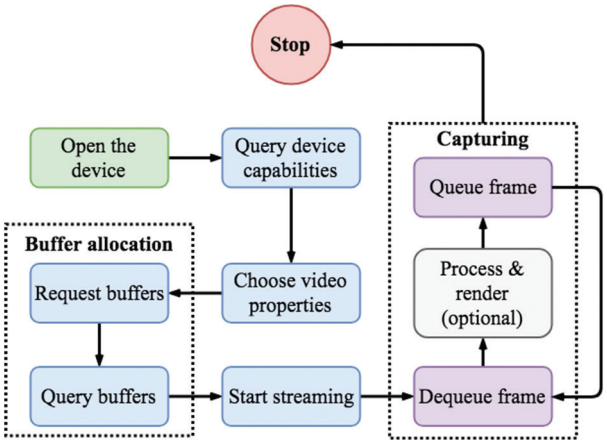

Figure 3 | Basic Video4Linux2 API video capturing workflow. Initial buffer allocation and capturing loop are highlighted with dashed lines

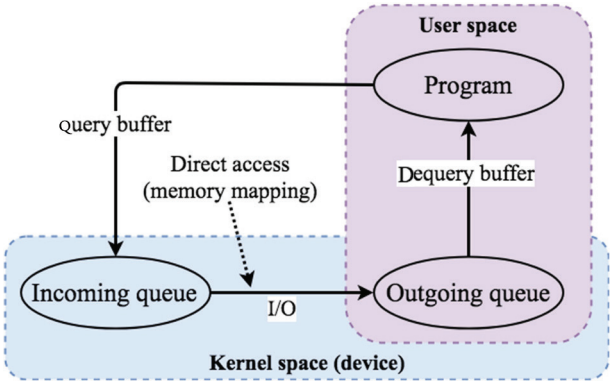

Figure 4 | Capturing frames from a V4L2 device: requesting a recent video frame (query buffer) and retrieving it from the device (Dequery buffer)

The process of streaming is as follows:

- Put the buffer into the device's incoming queue (query buffer).
- Wait for the device to fill in the buffer with data, i.e., capture a video frame.
- Retrieve the buffer from the outgoing queue (dequery buffer).

Several issues were faced at the development stage. The most important one is an inability to get frames captured time (timestamp). This issue was sorted out by placing the information extraction step after the queuing and before the dequeuing operations.

3.2. ROS Package

The development of the video capturing (streaming) ROS package was conducted under ROS Kinetic distribution.

Our ROS package makes use of the developed V4L2 video streaming layer in order to capture video frames from the robot's cameras in real-time. Following ROS conventions of publishers and subscribers, one publisher can deliver data to multiple subscribers through topics, and the system becomes modular as it consists of multiple interchangeable parts. Thus, our video streaming package is comprised of publishers (one publisher for each camera), and we can easily integrate camera calibration for either stereo pair or each camera individually. Figure 5 demonstrates the high-level architecture of the developed ROS package.

Robot operating system publisher transmits video frames and camera information (such as frame resolution and pixel format) to subscribers, e.g., camera calibration node. Internally it uses V4L2 layer (Section 3.1) in order to get the most recent frames from a camera.

Parameters that are used in order to control streaming parameters of the ROS publisher node include:

- Frame rate – number of frames captured per second (e.g., 60 fps).
- Frame resolution – width and height of images in pixels (e.g., 640 × 480).
- Image format, or pixel format (e.g., "yuyv422").
- Path to the camera, or URL (e.g., "/dev/video0").

To run the video streaming node *rosrun* command is used along with defined camera parameters. ROS launch file was created for a convenient usage, so it is possible to start streaming from multiple cameras and view captured frames (*image_view* ROS package).

4. EXPERIMENTS

In this section, we provide performance evaluation results of the created ROS package. Comparison with an alternative solution for video streaming, OpenCV based ROS package, is presented. Robot's stereo vision system calibration is accomplished using circle grid and checkerboard pattern calibration techniques.

4.1. ROS Package Benchmark

For a mobile robot in order to be able to operate using its battery it is essential to minimize power consumption. Launched programs and applications consume system resources, such as CPU and memory, which are strictly limited in case of a mobile robot, hence it should be considered. In our benchmark we used only one camera of the Servosila Engineer, the rear view monocular camera. In order to estimate CPU and memory consumption *pidstat* utility was utilized. It outputs both intermediate and final results. Measurements were taken in the idle (no subscribers) and non-idle (with one subscriber) modes (Figure 6).

As an alternative we used an OpenCV based video streaming ROS package – *video_stream_opencv* [10]. However, in our system it did not work due to unsupported raw image format issues in OpenCV 3. Thus, it was decided to accomplish tests with an older version, OpenCV 2.4. In this scenario it was not possible to switch from an RGB to the raw image capturing mode (without image format conversion). The developed ROS package shows encouraging results in comparison to the OpenCV based one (Figure 7).

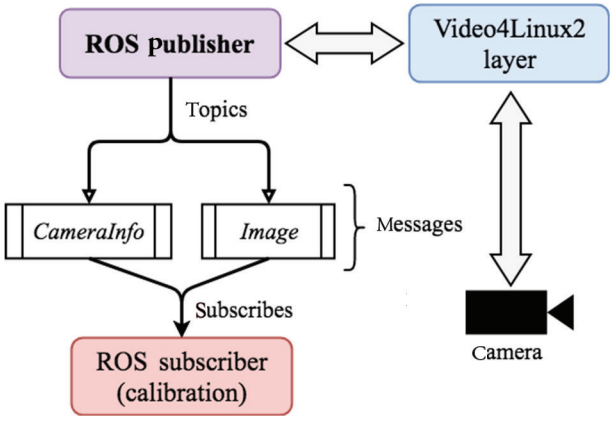

Figure 5 | High-level architecture of the developed video streaming ROS package. ROS publisher delivers video frames and camera information to subscribers from a camera using V4L2

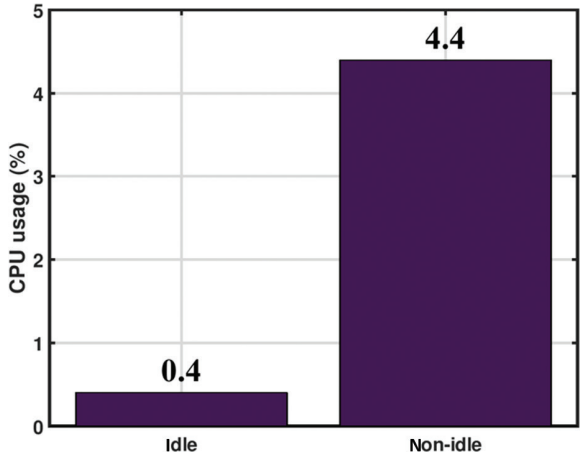

Figure 6 | Average CPU usage of our ROS package from the monocular rear view camera in the idle and non-idle modes

Statistics were estimated by the same *pidstat* utility for the rear view monocular camera of the robot. An average CPU usage of our ROS package implementation in the idle mode was close to 0 (0.4%). With a subscriber (non-idle mode) the resulting CPU usage was about 4.4%. As for the OpenCV based solution, *video_stream_opencv* demonstrated values of 31.6% in the idle and 35.1% in the non-idle modes. Thus, in the idle mode the OpenCV based ROS package continues to load the CPU.

Memory consumption is negligible in both cases – the buffer size is equal to four frames. In our package we can adjust the buffer size. However, even if it is set to the lower values, there is no benefit from it in terms of memory consumption and performance. Higher values only increase memory consumption and capturing latency.

4.2. Stereo Pair Calibration

Ideally, cameras of a stereo pair should be perfectly aligned along the same horizontal line passing through the principal points. Misalignment can lead to the potential growth of the error in measurements [11]. However, in some cases there are various hardware imperfection issues, which could present initially or appear due to severe operation conditions. Moreover, distortion issues do not enable to implement some algorithms, which require accurate measurements. In our case there is a radial distortion (Figure 8) with straight lines in the 3D world being distorted into curved ones. Radial distortion is spreading starting from the image's principal point.

Stereo vision system of our robot is comprised of two monocular cameras DFM 22BUC03-ML that are located in the robot's head (Figure 2 – left image). Left and right cameras of the stereo pair are displaced relative to each other with the right camera being higher. Figure 9 demonstrates distortion and displacement resulting issues of the stereo pair.

Camera calibration consists of extrinsic and intrinsic parameters estimation along with calculation of distortion coefficients [12].

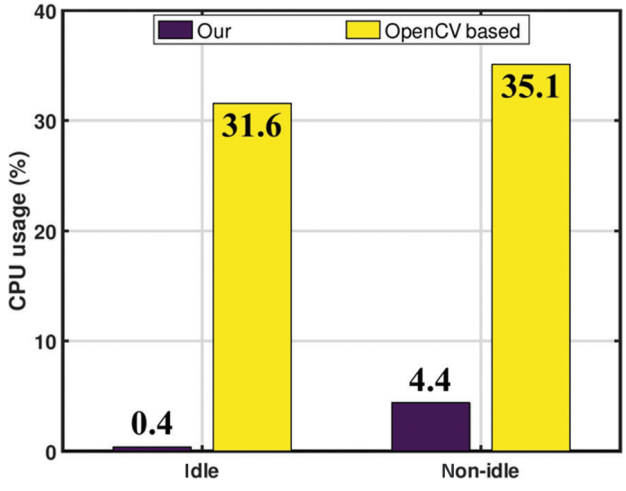

Figure 7 | Average CPU usage of our ROS package in comparison to the *video_stream_opencv* in the idle and non-idle modes for the rear view monocular camera

Figure 8 | A picture taken from the front left monocular camera of the Servosila Engineer mobile robot (grayscale)

Figure 9 | Pictures taken from the left and right cameras of the robot's stereo pair (the center of the right camera is higher than the left camera)

Intrinsic and extrinsic, parameters enable to transform objects in the 3D world into the 2D image plane. For the stereo pair we need to calculate their relative (spatial) relation. Obtained parameters are used to align images from both cameras along the same plane (stereo rectification). In our case we utilized 2D patterns, so the process of calibration included tracking of a planar pattern, which was located at different orientations relatively to the camera.

In calibration with a checkerboard pattern and circle grid *camera_calibration* ROS package was used [13]. It subscribes to left and right camera topics each of which conveys video frames and camera information. In order to stream from the stereo pair our developed node was used. First, calibration with a checkerboard pattern was executed (Figure 10).

Calibration with a checkerboard pattern showed acceptable accuracy with the best result of 0.33 pixels error. Next, calibration with a 6×6 circle grid was carried out (Figure 11). In this case the best result was the error of 0.18 pixels. However, even though the difference between circle grid and checkerboard pattern calibration results is invisible to a human eye, measurements accuracy may increase when using a circle grid.

Calibration process of the misaligned stereo pair resulted in partial information loss. Figure 12 demonstrates how visual information could be lost after the distortion elimination and rectification processes. Approximately 10% of pixels were lost.

The final result of the stereo pair calibration is shown in Figure 13. Calibration corrected the misalignment of the stereo pair cameras. Radial distortion was mitigated and curved lines from the initial images became straight.

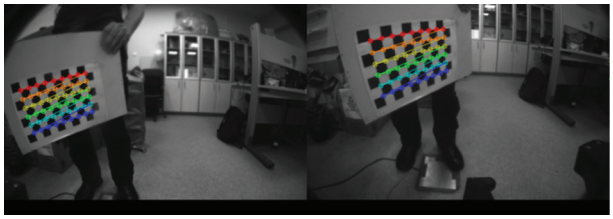

Figure 10 | Calibration of the robot's stereo pair with a checkerboard pattern in ROS (images are shown in grayscale)

Figure 11 | Calibration of the robot's stereo pair with a circle grid in ROS (images are shown in grayscale)

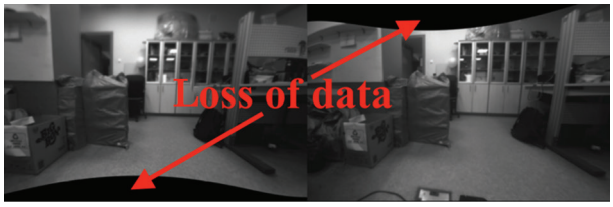

Figure 12 | Loss of data after calibration of the robot's stereo pair (rectification process). Black areas represent the lost pixels

Figure 13 | Servosila Engineer's stereo pair calibration result (rectified and undistorted images). Part of loss pixels is truncated

5. CONCLUSION AND FUTURE WORK

In this work we developed the real-time video capturing ROS package based on Video4Linux2 multimedia API for a low-level interaction with Servosila Engineer's cameras. The created ROS package demonstrated good results in terms of CPU usage and memory consumption enabling to capture and stream video in real-time. Comparison with the alternative OpenCV based ROS package using one monocular rear view camera demonstrated that our solution both in idle and non-idle modes, when there are no clients and there is a client respectively, performs better in terms of CPU usage: 0.4% vs 31.6% and 4.4% vs 35.1% in the idle and non-idle modes respectively.

Using created ROS package robot's stereo pair calibration in the presence of hardware imperfection was executed. A circular grid and checkerboard pattern were used. A circle grid based calibration showed more accurate results compared to a checkerboard pattern: 0.18 vs. 0.33-pixel error. As a result of calibration approximately 10% of pixels were lost due to the rectification and undistortion processes. Unfortunately, the loss of data in inevitable during such procedures. As a part of our future work, we plan to develop an real time streaming protocol (RTSP) server for teleoperation purposes.

ACKNOWLEDGMENTS

This work was partially supported by the Russian Foundation for Basic Research (RFBR) (project ID 18-58-45017). Part of the work was performed according to the Russian Government Program of Competitive Growth of Kazan Federal University.

REFERENCES

[1] C. Brand, M.J. Schuster, H. Hirschmüller, M. Suppa, Stereo-vision based obstacle mapping for indoor/outdoor SLAM, 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Chicago, IL, USA, 2014, pp. 1846–1853.

[2] A. Buyval, A. Gabdullin, R. Mustafin, I. Shimchik, Realtime vehicle and pedestrian tracking for Didi Udacity self-driving car challenge, 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, Brisbane, QLD, Australia, 2018, pp. 2064–2069.

[3] A. Buyval, I. Afanasyev, E. Magid, Comparative analysis of ROS-based monocular SLAM methods for indoor navigation, Proceedings of the Ninth International Conference on Machine Vision 10341, 2016, p. 103411K.

[4] E. Magid, T. Tsubouchi, E. Koyanagi, T. Yoshida, Static balance for rescue robot navigation: Losing balance on purpose within random step environment, 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Taipei, Taiwan, 2010, pp. 349–356.

[5] F. Dai, Q. Kang, Y. Yue, P. Xie, Research on visualized search and rescue robot, J. Robot. Network. Artif. Life 3 (2016), 201–204.

[6] Y. Ru, H. Du, S. Li, H. Chang, Action recognition based on binocular vision, J. Robot. Network. Artif. Life 4 (2017), 5–9.

[7] Z. Zhang, Camera calibration with one-dimensional objects, IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Computer Society, IEEE, Washington, DC, USA, 2004, pp. 892–899.

[8] R. Safin, R. Lavrenov, Implementation of ROS package for simultaneous video streaming from several different cameras, 2018 International Conference on Artificial Life and Robotics, Oita, Japan, 2018, pp. 220–223.
[9] Linux Media Subsystem Documentation – Video for Linux API, Available from: https://linuxtv.org/downloads/v4l-dvb-apis/uapi/v4l/v4l2.html (Accessed 22 August 18).
[10] OpenCV based video stream ROS package, Available from: http://wiki.ros.org/video_stream_opencv (Accessed 22 August 18).
[11] W. Zhao, N. Nandhakumar, Effects of camera alignment errors on stereoscopic depth estimates, Pattern Recognit. 29 (1996), 2115–2126.
[12] J. Suriansky, M. Cmarada, Analysis of methods for camera calibration in 3D scanning systems, Annals of DAAAM for 2012 & Proceedings of the 23rd International DAAAM Symposium, Curran Associates, Inc., NY, USA, 2012, pp. 365–368.
[13] camera_calibration ROS package Wiki, Available from: http://wiki.ros.org/camera_calibration (Accessed 22 August 18).

Authors Introduction

Professor Evgeni Magid

Professor Evgeni Magid is a Head of the Intelligent Robotics Department and a Head of Intelligent Robotic Systems Laboratory at the Higher School of Information Technology and Intelligent Systems, Kazan Federal University, Russia.
He obtained his Ph.D. degree at the Intelligent Robot Laboratory, University of Tsukuba, Japan in 2011. Currently, his main interests are urban search and rescue, swarm robotics, and robotics education.

Dr. Edgar A. Martínez-García

Dr. Edgar A Martínez-García is an Associate Professor and Head of the Robotics Laboratory at the Institute of Engineering and Technology, Universidad Autonoma de Ciudad Juarez, Mexico.
He obtained his Ph.D. degree at the Intelligent Robot Laboratory, University of Tsukuba, Japan in 2005. Currently, his main interests in robotics are mathematical modelling, dynamic control, underactuated systems, and biomechanics design.

Mr. Roman Lavrenov

Mr. Roman Lavrenov is a lecturer and research associate at Intelligent Robotic Systems Laboratory, Higher School of Information Technology and Intelligent Systems, Kazan Federal University, Russia. His main interests are path planning, robot operating system, and autonomous mobile robots.

Mr. Ramil Safin

Mr. Ramil Safin is a master student and research assistant at Intelligent Robotic Systems Laboratory, Higher School of Information Technology and Intelligent Systems, Kazan Federal University, Russia. His main interest is computer vision for robotic applications.