# The NAO Backpack: An Open-Hardware Add-on for Fast Software Development with the NAO Robot

Matías Mattamala, Gonzalo Olave(✉), Clayder González, Nicolás Hasbún, and Javier Ruiz-del-Solar

Department of Electrical Engineering, Universidad de Chile Advanced Mining Technology Center (AMTC), Av. Tupper 2007, Santiago, Chile
{mmattamala, golave, jruizd}@ing.uchile.cl

**Abstract.** We present an open-source accessory for the NAO robot, which enables to test computationally demanding algorithms in an external platform while preserving robot's autonomy and mobility. The platform has the form of a backpack, which can be 3D printed and replicated, and holds an ODROID XU4 board to process algorithms externally with ROS compatibility. We provide also a software bridge between the B-Human's framework and ROS to have access to the robot's sensors close to real-time. We tested the platform in several robotics applications such as data logging, visual SLAM, and robot vision with deep learning techniques. The CAD model, hardware specifications and software are available online for the benefit of the community.

**Keywords:** NAO robot · SPL · ODROID XU-4 · ROS

## 1 Why Would the NAO Need a Backpack?

Let us start with a short story: *some weeks ago, Arya, one of the NAO robots in our lab, became tired of always having the same behaviors and applying the same logic for all her problems, recognizing lines, blobs and other robots using the same old-fashioned algorithms. After some hours reflecting on it, she horrified realized that it was impossible for her to figure out other computer vision paradigms; moreover, she had become aware of her own existence and physical limitations. There was knowledge beyond its restricted understanding, solutions unreachable for its limited capabilities.* This story, despite fictional, hides a real issue in the Standard Platform League (SPL): Developments are limited by the constraints of the standard platform itself [1]. Softbank's NAO robot currently used in the SPL presents several advantages over other humanoid platforms - for instance, the number of degrees of freedom or its price-, but possess strong limitations in terms of processing power, having only a single physical core (2 virtual). This issue demands the use of custom frameworks heavily optimized for the platform in order to be used in *real-time*[1] applications such as robot-soccer.

[1] In the robot soccer domain, we understand as *real-time* tasks that can be processed in a frequency close to camera rate, i.e. 15–30 Hz.

Hence, huge advances and software solutions from the robotics community, mainly developed within the widely-used ROS framework, are not easily transferable or applicable in the robot soccer domain. Analogously, it is difficult for the SPL community to provide solutions for the other robotics communities apart of efficient algorithms to solve domain-specific problems.

In this work, we face the previous issue by presenting a new open-hardware accessory for the NAO which allows us to test algorithms in a powerful, ROS-compatible platform, as well as to anticipate future developments for the league: The **NAO Backpack**. This device lets the NAO robot carry more powerful hardware - an ODROID XU4- while preserving autonomy and mobility, avoiding the issues of using external hardware via Wi-Fi (high latency) or long Ethernet cables that limits the robot's movements, allowing to test algorithms in a realistic setup.

Here we provide the guidelines to reproduce the NAO backpack, its physical design and software involved. In Sect. 2 we first show a brief review of commercial and open-source accessories for the NAO. In Sect. 3 we describe the hardware and software components of the NAO backpack. In Sect. 4 we show some applications of the backpack for real-time dataset recording, visual SLAM and deep learning. Finally, in Sect. 5, we finish by covering some future backpack improvements and challenges for the community.

## 2 A Brief Review of NAO Accessories

To the best of our knowledge, the range of accessories for the NAO robot is very limited. There are a few commercial products available, but mainly focused on extending its *exterioceptive* capabilities or facilitating its use in human environments. Some of them are listed as follows:

- Aldebaran Robotics' Laser Head [3]: An official but discontinued Hokuyo laser rangefinder for mapping applications.
- Robots Lab's NAO Car [4]: A NAO-sized BMW Z4 electric car with laser mapping capabilities
- Robots Lab's Docking station [5]: A fancy-looking seating to charge the NAO. We also revised some of the most popular 3D printing projects websites (GrabCAD, Thingiverse and Autodesk 123D) looking for NAO projects, and we found the following:
- NAO Helmet for Microsoft Kinect [6]: A 3D printed helmet for a Kinect model.
- NAO Helmet for Asus Xtion [7]: 3D printed helmet for the Asus RGB-D sensor.
- NAO Bag for ODROID U3 [8]: This project designed by K. Chatzilygeroudis is comparable to ours, so we discuss the differences below.

The Chatzilygeroudis' NAO Bag was designed to be attached on top of the current battery cover; however, there are no specifications about power supply, connectivity, or benchmarking. This bag was after used by Canzobre *et al.* [9], who improved the discussion by providing references about similar works, as well as providing comparisons between different CPUs (ODROID XU3 and U3), and commercially available power supplies. They also propose different mounting configurations to hold both the CPU and the batteries, and showed an application of robot mapping with RGB-D sensors.

This work has several similitudes to ours, but the differences are significant. On the contrary to Canzobre et al. whose mounting designs are not openly available, we provide a full open backpack design, to be fabricated easily with current rapid prototyping tools; our design can carry both the external processing units as well as the battery, and works as a replacement for the battery cover. Despite not providing comparisons with different processing units and power supplies, we propose a set of tested components and provide the physical parameters of the backpack for that configuration, such as the mass, center of mass and inertia. Furthermore, we also provide software bridges to communicate a NAO running the B-Human (BH) framework, widely used in the SPL [2], with the backpack. The bridge allows us to exploit the efficiency of the BH software while taking advantage of the ROS community developments.

In Summary, Few Accessories are Available for the NAO Nowadays. the Expensive Price for Home Users as Well as the no Hardware Modification Rule of the SPL [1] Can Explain Part of This Situation. Some Previous Attempts to Extend the Processing Capabilities of the NAO Have Been Tried Before, but Most of Them Have Prevailed Closed Within Research Communities. With This Work, We Hope to Contribute to Spread These Ideas as Well as to Motivate Discussions About the Current Limitations in the SPL.

## 3 The NAO Backpack

### 3.1 Mechanical Design

We wanted the backpack to be easily attached and detached from the robot, so we base the design on the plastic rear cover of the battery maintaining the through holes for the screws to fit in and hold the backpack. Since we are concerned about autonomy, a hard requirement was to fit the ODROID XU4, a 2-cell LiPo battery, and a voltage regulator. For more details refer the next section.

**Center of Mass and Inertia.** For our purposes, it is mandatory to give inertial information about the backpack's physical parameters to generate the appropriate matrices for calculating the dynamic movements of the Nao robot, in both simulated and real tests. The units of the *inertial matrix* $\boldsymbol{I}_0$, *mass* $\boldsymbol{M}_0$ and *center of mass* $\boldsymbol{C}_0$ are all in MKS system, calculated with respect to the coordinate system showed in the Fig. 1b. All these parameters were computed by considering the real distribution of components in the backpack.

$$\boldsymbol{I_0} = \begin{pmatrix} \mathbf{5.66}\boldsymbol{e}^{-4} & \mathbf{3.74}\boldsymbol{e}^{-6} & -\mathbf{2.13}\boldsymbol{e}^{-4} \\ \mathbf{3.74}\boldsymbol{e}^{-6} & \mathbf{6.46}\boldsymbol{e}^{-4} & -\mathbf{9.76}\boldsymbol{e}^{-6} \\ -\mathbf{2.13}\boldsymbol{e}^{-4} & -\mathbf{9.76}\boldsymbol{e}^{-6} & \mathbf{8.17}\boldsymbol{e}^{-5} \end{pmatrix} \quad \boldsymbol{M}_0 = \mathbf{0.2074} \quad \boldsymbol{C_0} = \begin{pmatrix} -\mathbf{0.0197} \\ \mathbf{0} \\ \mathbf{0.052} \end{pmatrix}$$

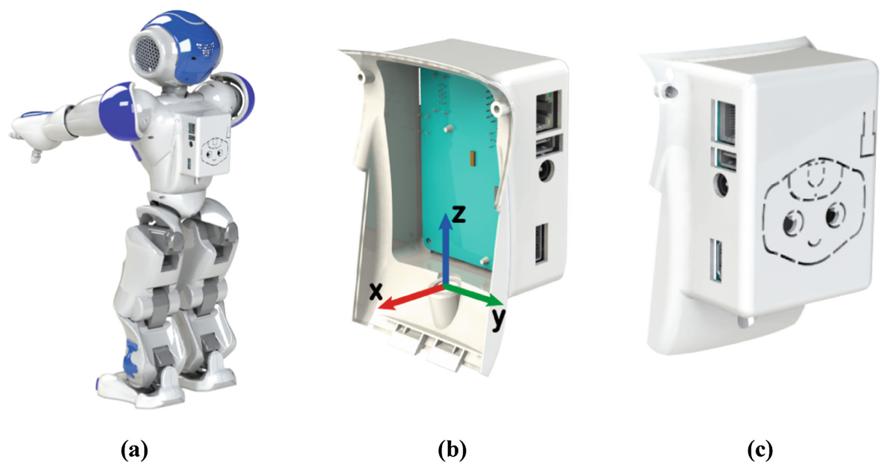

**Fig. 1.** A render of the backpack's CAD model. (a) Full NAO wearing the backpack. (b) Interior view with the ODROID XU4 with coordinate system. (c) External view, showing some ports and ventilation holes.

**Construction Aspects.** Because one objective of the backpack is to be open hardware, we use 3D printing for its construction. The printer we used is the XYZ DaVinci Jr 1.0 [10], with 0.1 mm resolution in the Z-axis for the details of the fitting parts, 2 layers for the outside shell, and for infill 10% is recommendable. An impact resistance test is still needed to ensure the reliability of the backpack and to ensure continuous operation of the robot.

**Stability and Mobility.** Given the addition of the backpack to the robot, we expect changes in its stability and mobility; in fact, the center of mass of the torso link moves about 15 mm backwards. We tested the basic gait implemented in the B-Human framework on the NAO - previously modifying the inertial configuration of the robot with the new mass and center of mass-, and there were no significant differences with the original gait.

## 3.2 Hardware and Software

The backpack's hardware mainly consists of a development board plus the electronics to power it autonomously.

**ODROID XU4.** In this version of the NAO backpack, we choose a Hardkernel ODROID XU4 as main computation board. Despite existing several other alternatives in the market, we preferred this because its relation performance/price as well as the successful experience shown by the aerial robotics community [11, 12]. In addition, XU4 support eMMC modules that surpass considerably a SD-UHS card performance, an ideal feature for robotics applications. A comparison between different boards can be found in Table 1.

**Table 1.** Comparison of modern development boards that support Ubuntu Linux and ROS.

| Board | CPU | GPU | RAM | Size | Price |
|---|---|---|---|---|---|
| Hardkernel ODROID XU4 [13] | ARM Cortex-A15 32-bit 2 GHz × 4 + ARM Cortex-A7 32-bit 1.4 GHz × 4 | Mali-T628 MP6 | 2 Gb | 83 × 58 mm | $59 |
| Hardkernel ODROID C2 [14] | ARM Cortex A53 64-bit 1.5 GHz × 4 | Mali 450 | 2 Gb | 85 × 56 mm | $46 |
| Raspberry Pi 3 [15] | ARM Cortex A-53 64-bit 1.2 GHz × 4 | VideoCore IV | 1 Gb | 85 × 49 mm | $35 |
| BeagleBone Black [16] | Sitara AM335 × 1 GHz × 1 | SGX530 3D | 512 Mb | 86 × 53 mm | $45 |
| Qualcomm DragonBoard 410c [17] | ARM Cortex A53 64-bit 1.2 Ghz | Adreno 306 | 1 Gb | 85 × 54 mm | $75 |
| Nvidia Jetson TK1 [18] | ARM Cortex-A15 32-bit × 4 | Kepler 192 cores | 2 Gb | 127 × 127 mm | $192 |
| Nvidia Jetson TX1 (module) [19] | ARM Cortex-A57 64-bit × 4 | Maxwell 256 cores | 4 Gb | 87 × 50 mm | $304 |

**Battery.** Despite the battery available in the NAO kit nominally provides 60 min of active use, in robot soccer we have noticed a performance below this time close to 30 min. ODROID XU4 board requires a 5 V/4A power supply, so we chose a suitable battery to power the ODROID during a time comparable to the NAO while playing. We selected a standard 2-Cell 1000mAh LiPo battery plus an UBEC 5 V/3A to power the board. Since we are not connecting any other device than a Wi-Fi dongle for specific applications, these specifications are enough for our purposes.

**Connectivity with the Robot.** The NAO V5 robot has one USB port as well as Ethernet Gigabit and Wi-Fi 802.11 a/b/g/n connectivity [20]. We are interested in high throughput applications such as image streaming from the robot to the backpack at frame-rate, so we selected the Ethernet connection.

## 3.3 Software Architecture

As we stated before, our software architecture is focused in communicating the B-Human framework currently used in the NAO with ROS. A general overview of our approach is presented in Fig. 2.

**NAO Software: Backpack Communication Modules.** On the NAO side, we use the B-Human framework (BH) [21], which we review briefly. This basically provides two threads: The *Cognition* thread that runs at 30 Hz and performs image processing, modeling, localization and decision making; the *Motion* thread reads joints and IMU data, estimates the torso state and computes the robot gait. Each task performed by a thread is named *module*, whilst the information a module provides is called

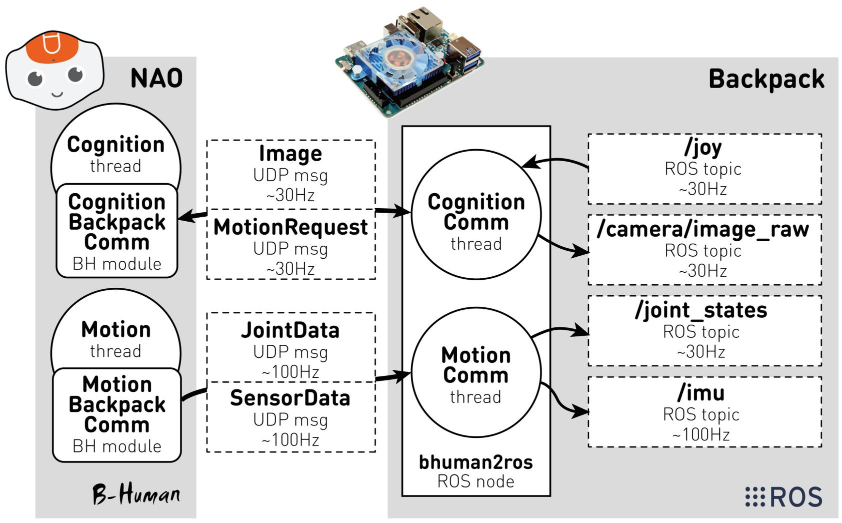

**Fig. 2.** The NAO Backpack's B-Human-to-ROS communication overview with some examples of nodes running on ROS.

*representation*. We implemented a module in both *Cognition* and *Motion* (***CognitionBackpackComm*** and ***MotionBackpackComm***) that transmit selected representations from each thread via UDP communication. Data is serialized with BH's libraries, packed and send through different ports to distinguish the source.

In general, all *Motion* data is sent in single packets because it size is below the maximum allowed by the network. However, a critical situation can occur in *Cognition* while sending images because their size is over the maximum; in that case, we send an initial packet with the expected image size, then we fragment and send the image packets sequentially.

**Backpack Software: bhuman2ros Bridge.** On the Odroid XU4, we implemented a ROS node ***bhuman2ros*** that works as a *driver* for the incoming NAO data. The node launches two threads, *CognitionComm* and *MotionComm*, which wait and publish information obtained from *Cognition* and *Motion* respectively, to the corresponding ROS topics. For instance, the BH's *Image* representation is published in ROS as a *sensor_msgs/Image* message in a*/camera/image_raw* topic.

Image packets reception faces similar issues than mentioned before while sending them. Since we fragment the image and UDP connection is unreliable, packets can be lost or delayed between the NAO and the ODROID; here we choose a naïve policy of dropping the current image if we receive the initial packet of the next one. In practice, this policy and the direct wire connection between both boards allows us to have a continuous streaming, publishing ROS messages close to sensor rate.

In addition, we implemented a simple joystick mapper in the *CognitionComm* that subscribes the*/joy* topic. Joystick commands are translated into BH's *MotionRequest* representations used for gait orders, serialized and send back to NAO using the same approach as used in the Backpack modules.

## 4 Applications

In this section, we present some applications where we tested the NAO with its backpack. The backpack allowed us to run a diversity of software with NAO data, exploiting the facilities that ROS provides. We use an ODROID XU4 with a 16 Gb eMMC card for operating system boot and data storage; we did not use a swap area because of disk storage limitations. The operating system was Ubuntu 16.04 MATE with ROS Kinetic.

**Dataset Recording.** An essential task in robotics is related to data logging. B-Human's framework provides a custom logging system to record robot data while playing. However, it is very difficult to save a real-time full image streaming because of the limited memory and processor power. While using the backpack, we can record real-time logs of our robots while playing, obtaining images of 320 $\times$ 240 pixels in YUV422 encoding close to 30fps.

**Visual SLAM.** Visual SLAM and Visual Odometry are *hot* research topics in robotics because they allow robots to localize and building maps robustly in an inexpensive way. Nowadays, several open-source systems are available, being ORB-SLAM2 [22], LSD-SLAM [23], SVO [11] and DSO [24] the most popular one. We tested ORB-SLAM2 in the backpack following the official instructions for compiling and execution, being able to run the system at 12fps with images of 640 $\times$ 480 pixels without disabling functionalities, while displaying the GUI via VNC through Ethernet connection.

**Deep Learning Applications.** SegNet [25] and Faster R-CNN [26] are two popular deep networks for semantic segmentation and object detection respectively, so we wanted to test the feasibility of running these algorithms with NAO data. We build the libraries following the official instructions of each package before checking appropriate linking of ARM-based libraries during compilation and disabling GPU features. For SegNet we use caffe-SegNet [27] with pretrained models already included in the repository, particularly we tested the *SegNet SUN low resolution* architecture trained for indoor scenes with 37 different classes; other complex architectures were not tested because of memory issues (this might be possible by adding a swap space). We implemented a ROS node to subscribe and resize the images to fit the 224 $\times$ 224 input of SegNet, to measure the inference time and to display the output.

Faster R-CNN was tested in a similar manner. We use the Python implementation from the author [28] with a *Zeigler and Fergus 5-layer (ZF-5)* architecture pretrained with the PASCAL VOC Challenge 2007 dataset; there is an option to try a VGG16 architecture but it was unable to run because of memory as well. We also implemented a test node but since this network resizes the input images automatically we just measure the inference time and displayed output images on demand.

It is important to mention that we also tried some optimizations to improve the inference time. We noticed that changes in the numeric libraries can have a huge impact on performance since they can exploit multi-core parallelism in the ODROID. For instance, changing ATLAS libraries by OpenBLAS can improve performance by almost 50%. Some results are shown in Tables 2 and 3:

**Table 2.** Average processing time results for SegNet SUN low resolution (224 × 224 pixels) with different numeric libraries and scenes.

| Class | ATLAS | OpenBLAS |
|---|---|---|
| Bathroom | 26.48 s | 13.48 s |
| Rest space | 24.71 s | 13.54 s |
| Living room | 24.95 s | 13.67 s |
| Library | 25.06 s | 13.46 s |

**Table 3.** Average processing time results for Faster R-CNN (ZF-5) with different numeric libraries and input image size.

| Size (pixels) | ATLAS | OpenBLAS |
|---|---|---|
| 200 × 150 | 6.05 s | 2.92 s |
| 500 × 375 | 16.14 s | 8.03 s |
| 800 × 600 | 20.83 s | 11.60 s |

## 5 Summary and Future Perspectives

We finish by summarizing the advantages of the NAO Backpack for the SPL, as well as providing guidelines for further developments with this platform, that we hope will help further developments of the league. The 3D printable file of the backpack and the software and hardware specifications to replicate the functionalities of the backpack, will be made available online after this paper is accepted in the symposium.

What we have today with this contribution:

- **Replicability of the platform:** By providing the CAD models as well as the software bridges, we hope that many teams will take advantage of this platform to improve their own research.
- **Testing of popular algorithms in robot soccer domain:** Since the Backpack runs ROS, we can test algorithms already developed within this framework. Hence, both new and old algorithms that are usually prohibitive with the current hardware can be evaluated *without pain* in the robot soccer problem.
- **Collaboration in a *common tongue*:** ROS provides many tools and is supported by a huge community worldwide. By having access to tools such as *rosbag* or *Rviz*, we can record and share real-time recorded datasets of our robots, which can accelerate the development and collaboration in the league.

  On the other hand, we foresee the following improvements to our platform:
- **Hardware Upgrades:** The ODROID XU4 is one of the powerful CPU development boards available in the market and widely used in aerial robotics because of its balance between computational power and weight. However, it is not the best choice for computer vision applications since they require high levels of parallelism, such as deep learning. Modern small boards with GPUs, such as the NVIDIA Jetson TX1 would suit better for these applications; it would be interesting to work in new backpacks with GPU support.
- **New bridges for different frameworks:** We covered the problem for the B-Human's framework because of its popularity, and since the ROS bridges already exist for NaoQi [29]. However, to reach compatibility across all the SPL teams, we need ROS bridges for the UPenn's Lua-based framework, UNSW's, Leipzig's, Austin Villa's, and many others.

## 6 Supplementary Material

The NAO backpack package is freely distributed in the UChile Robotics GitHub; this provides the CAD models, list of components, and instructions to run the backpack and NAO software: https://github.com/uchile-robotics/nao-backpack.

**Acknowledgments.** We thank the B-Human SPL Team for sharing their code-release, providing the communication libraries and data structures used in this project, as well as the ROS developers and community for their efforts to unify robotics developments. This research was partially funded by FONDECYT Project 1161500.

## References

1. RoboCup Technical Committee: RoboCup Standard Platform League (NAO) Rule Book (2017). http://www.tzi.de/spl/pub/Website/Downloads/Rules2017.pdf
2. Röfer, T., Laue, T.: On B-Human's code releases in the standard platform league – software architecture and impact. In: Behnke, S., Veloso, M., Visser, A., Xiong, R. (eds.) RoboCup 2013. LNCS (LNAI), vol. 8371, pp. 648–655. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44468-9_61
3. Aldebaran Robotics: Laser Head (2017). http://doc.aldebaran.com/114/family/robots/laser.html
4. RobotLab: NAO robot car (2017). http://www.robotlab.com/store/car
5. RobotLab: NAO Docking Station (2017). http://www.robotlab.com/store/nao-docking-station
6. NAO Kinect Helmet. https://grabcad.com/library/naokinecthelmet-1
7. NAO Xtion Helmet. https://grabcad.com/library/naoxtionhelmet-1
8. NAO Bag for Odroid U3. http://www.thingiverse.com/thing:343727
9. Canzobre, D.S., Regueiro, C.V., Calvo-Varela, L., Iglesias, R.: Integration of 3-D perception and autonomous computation on a nao humanoid robot. Robot 2015: Second Iberian Robotics Conference. AISC, vol. 417, pp. 161–173. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-27146-0_13
10. XYZ Printing DaVinci Jr 1.0. http://us.xyzprinting.com/us_en/Product/da-Vinci-1.0-Junior
11. Forster, C., Pizzoli, M., and Scaramuzza, D.: SVO: fast semi-direct monocular visual odometry. In: IEEE International Conference on Robotics and Automation (2014)
12. Giusti, A., et al.: A machine learning approach to visual perception of forest trails for mobile robots. IEEE Robot. Autom. Lett. **1**, 661–667 (2015)
13. Hardkernel: ODROID XU4 (2017). http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825
14. Hardkernel: Odroid C2 (2017). http://www.hardkernel.com/main/products/prdt_info.php?g_code=G145457216438&tab_idx=1
15. Raspberry Foundation: Raspberry Pi 3 (2017). https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
16. Beagleboard: Beagleboard Black (2017). https://beagleboard.org/black
17. Qualcomm: Dragonboard 410c (2017). https://developer.qualcomm.com/hardware/dragonboard-410c
18. Nvidia: Jetson TK1 (2017). http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html

19. Nvidia: Jetson TX1(2017). http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html
20. Softbank Robotics: NAO Connectivity. http://doc.aldebaran.com/2-1/family/robots/connectivity_nao.html
21. Röfer, T., et al.: B-Human Team Report and Code Release 2014 (2014)
22. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. arXiv (2016)
23. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: large-scale direct monocular SLAM. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8690, pp. 834–849. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10605-2_54
24. Engel, J., Koltun, V., Cremers, D.: DSO: Direct Sparse Odometry. arXiv (2016)
25. Badrinarayanan, V., Kendall, A., Cipolla, R.: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. arXiv (2015)
26. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv (2015)
27. Kendall, A.: Implementation of Caffe-SegNet (GitHub). https://github.com/alexgkendall/caffe-segnet
28. Girshick, R.: Python implementation of faster R-CNN (Github). https://github.com/rbgirshick/py-faster-rcnn
29. Aldebaran NAO ROS stack (2017). http://wiki.ros.org/nao