Try to Start It! The Challenge of Reusing Code in Robotics Research

Enric Cervera

***Abstract*—This paper reviews the source code published with the papers of a flagship robotics research conference, 2017 International Conference on Robotics and Automation (ICRA). The aim is to investigate whether the code is actually useful, i.e. can be reused by an interested reader without much effort. The interest is twofold: for one side, it makes possible to replicate and validate the results of the research; for another side, it facilitates new progress on the field, since researchers can build new systems on top of existing work. Unfortunately, reusing code is not as straightforward as it could seem, and there is a need for tools that alleviate the effort for integrating someone else's code into the own user's system. We propose the use of Docker, a Linux container technology, to turn the source code repositories into executable images, that can be run and tested locally, in an isolated environment, without the need of a costly integration with the host system.**

***Index Terms*—Software, Middleware and Programming Environments;**

I. INTRODUCTION

KNOWLEDGE sharing is a cornerstone in the progress of science. Publication of new ideas for review and criticism by peers in conferences and journals is the widely-accepted way to validate and establish the research works of the scientific community.

In the spirit of transparency and cooperation, the sharing of source code is slowly but steadily becoming a common practice in the robotics community, not only to enforce the replicability of research, but to overcome the increasing complexity of modern robotic systems, allowing the researchers to focus on new developments, not on each time having to reinvent the wheel.

Nevertheless, reusing someone else's code can be a challenging experience with an uncertain ending: things may integrate silently, smoothly; or the process may become a time-consuming, painstaking, epic fail.

In this work, we analyze the difficulties in reusing the source code available in public repositories provided by current robotics research papers. We have looked into the proceedings of a recent flagship conference in robotics, 2017 IEEE International Conference on Robotics and Automation (ICRA), and tested the source code made available with the papers.

Our aim is not to replicate the experimental results of the papers, but to test the usability of the code, as posed in the following

---

Question: Can a user build, install, and run the code without errors, by following the instructions given in the code repository?

Our results indicate that the shared code and documentation needs to be improved for a more effective use by the community. We present a protocol and platform, with minimal additions over the common practices, to achieve a simpler and less traumatic experience in the reuse of robotics source code.

II. STATE OF THE ART

Reproducibility of robotics research has gained increasing attention in recent years [31], and reusing the own authors' code is possibly the most straightforward way for reproducing the experiments and results of conference papers.

The robotics community has steadily evolved towards the use of off-the-shelf programming frameworks which allow the researchers to share a common development base, some of the most popular are ROS [32], YARP [33], or OROCOS [34].

Interestingly, the use of such frameworks has not led to a simplification of the reproduction process: systems have become increasingly complex, combining different tools, and a number of dependencies, which are frequently hidden.

Efforts for automating tasks and addressing the above-mentioned challenges have been proposed: Lier's Cognitive Interaction Toolkit [35], [36] is an integrated tool chain that incorporates the development, reproduction, and refinement process of robotic systems.

SwarmRob is another toolkit recently proposed [37], which uses a holistic approach based on operating-system-level virtualization for dealing with the problem of reproducibility and sharing of experimental artifacts.

Cloud software platforms are used by some robotics journals, e.g. Code Ocean in Robotics and Automation Magazine [38]. Besides submitting the article, the authors upload the complete source code to the cloud, so it can be shared and run by interested readers.

The ROS build farm is another example of cloud-based system for automatic builds and deployment [39], continuous integration and autodocumentation. It is well suited for mature software packages, but the learning curve is steep, possibly not worth for experimental code.

Manuscript received: July, 20, 2018; Revised September, 23, 2018; Accepted October, 23, 2018.

This paper was recommended for publication by Editor Tamim Asfour upon evaluation of the Associate Editor and Reviewers' comments. This paper describes research done at the Robotic Intelligence Laboratory. Support for this laboratory is provided in part by Ministerio de Economia y Competitividad (DPI2015-69041-R), by Generalitat Valenciana (PROMETEOII/2014/028) and by Universitat Jaume I (P1-1B2014-52).

ORCID id: 0000-0002-5386-8968. The views and opinions expressed in this article are those of the author, not affiliated with any of the projects mentioned. Enric Cervera is with the Robotic Intelligence Laboratory, Jaume-I University, 12071 Castelló, Spain. ecervera@uji.es

Digital Object Identifier (DOI): see top of this page.

TABLE I: Platform specifications (Operating System, Middleware, Programming Language, and Software Dependencies) for the code published in the repositories of ICRA 2017 papers. This information has been extracted from the README files of the repositories.

Repository	OS	Middleware	Language	Dependencies
Self-triggered-mechanism [1]			Matlab	
PGO-Laginit [2]			Matlab	mMath, manopt
Fast-SeqSLAM [3]			Matlab, C++	
EKF-SLAM-on-Manifold [4]			Matlab	
LT-Algorithm [5]			Matlab	
lwAnn [6]			Java	
sun-bcnn [7]			Python	caffe, lmdb, cv2
delta-execution-models [8]			Python, C++	Visual Studio, Unreal Engine
bpvo [9]			C++ 11	Eigen 3.2+, OpenCV 2.11
driving-in-the-matrix [10]	Linux		Python	CUDA 8, nvidia docker
learning-forces [11]	Linux		C++, Python	Keras, SciPy, Theano, plotly, PyOpenGL
PUMP [12]	Unix		CUDA C	
rrd_slam [13]	Ubuntu		C++	OpenCV 2.4.8
MSGD [14]	macOS Sierra		C++	
SSM_linearArray [15]	Ubuntu 14.04		C++ 11	Pangolin, OpenCV, Eigen3, BLAS, LAPACK
crazyswarm [16]	Ubuntu 16.04		Python, C, Matlab	git, swig, numpy, yaml, matplotlib
jps3d [17]		ROS	C, C++	Eigen3, yaml-cpp
autonomy_hri [18]		ROS	C++	
skimap_ros [19]		ROS	C++	OpenMP, Eigen3, OpenCV 2.4, Boost
atom_mapping [20]		ROS	C++	
VI-MEAN [21]		ROS	C++	OpenCV, Eigen, Ceres, OpenChisel, camodocal
gps [22]		ROS	Python 2.7	numpy, matplotlib, scipy, boost, protobuf
team_acrv_2016 [23]		ROS Indigo	C++	MoveIt!, PCL, Baxter SDK
Incremental_DuDe_ROS [24]		ROS Indigo	C++	CGAL, Freeglut, MPFR, OpenCV
superquadric-grasping [25]		YARP	C++	IPOPT, OpenCV
costar_stack [26]	Ubuntu 14.04	ROS Indigo	C++, Python	Git, Catkin Build Tools, OpenCV 2.4 nonfree
human_robot_collaboration [27]	Ubuntu 14.04	ROS Indigo	gcc-4.9	NLOPT
segmap [28]	Ubuntu 14.04	ROS Indigo	C++	python-wstool, doxygen, autoconf
bayesian-object-tracking [29]	Ubuntu 14.04	ROS Indigo	C++ 11	Eigen 3.2.1
smartwatches_apps [30]	Ubuntu 14.10	ROS Indigo	Java, Python	

Some of the previous approaches use Docker, a well-established container technology that has proven useful for reproducible research in many scientific domains [40]. Integration of Docker with robotic frameworks (e.g. ROS) has also been proposed [41].

In this work, we extend the use of Docker to a set of representative robotics research papers, illustrating its usefulness in an academic setting. Unlike other frameworks [35], [37], the proposed method does not interfere with the common development workflow, since it smoothly integrates the authors' code repository with the automatic generation of Docker images. Moreover, it proves to be useful in a wide range of robotic domains.

Another advantage over cloud-based platforms like Code Ocean is that the software runs locally on the user's computer, performing not only numerical computations, but also rich user interactions with a full-featured graphical interface (e.g. rviz in ROS).

III. CASE STUDY

In this work we analyze the source code of 30 papers listed in Table I, which were selected among the ~800 papers published at ICRA2017, including Robotics and Automation Letters (RA-L), available at IEEE Xplore[1]. An automatic text search was carried out among all the PDFs of the conference, looking for the keywords of four of the main public repositories of open source code, namely *bitbucket*, *github*, *gitlab*, and *sourceforge*. The hits were then manually revised to confirm that the source code was available.

The vast majority of source code is published through the *github* platform[2]. Only two papers used *bitbucket* [42], [43] and none of them used neither *gitlab* nor *sourceforge*. For the reproducibility of our study, only the *github* platform has been considered.

With further inspection, we discarded those papers which presented datasets [44], [45], [46] or complex hardware/software platforms [47], [48]. We also discarded one paper that presented a completely empty repository [49], i.e. without any source code: upon questioning by an interested reader, the author answered that he was cleaning up the code before releasing it[3].

While this final selection is not exhaustive, nor perhaps representative, we believe that it is an interesting sample of the conference papers that make their source code available for the community. The set consists of 30 papers, 5 of them being also published in RA-L.

The themes of the papers vary among different robotic fields, and some of the most frequent keywords are:

[1] https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7960754

[2] https://www.github.com

[3] https://github.com/aspek1/JointCurvatureOptimisation/issues/1

collision avoidance [20][48], helicopters [16][42], human-robot interaction [11][18][30], industrial robots [26][27], learning [22][43], mapping [4][5][9], mobile robots [1][6][18][19][20][30][43][48], object detection [3][10], path planning [12][19], pose estimation [2][25][49], robot vision [3][6][14][20][21][24][25][48], sensor fusion [18][29], service robots[8][11], SLAM [2][4][9][13][14][20][24][48], stereo image processing [20][21][25], and visual tracking [9][29].

For the software to be used by the community, it must be properly documented. As shown in Table I, some information (operating system, dependencies) is not always provided by the authors, but every paper includes at least a URL to a public repository, where the source code is stored, along with additional information that is essential for putting the code at work.

We now review some of these additional data, specifically the information regarding the platform (operating system and other software dependencies). We also discuss some statistics about the repositories: how recently have been updated and how much are shared.

A. Platform Specifications

This information has been extracted from the README files of the repositories.

Firstly, the unambiguous specification of the Operating System (OS) is not common practice: only 8 repositories state both the OS name and version (Ubuntu 14.04, Ubuntu 16.04, macOS Sierra), whereas in 4 cases a more or less generic OS is mentioned (Linux, Unix, Ubuntu), and 18 repositories do not provide any information at all.

It can be argued that some application software is multi-platform and can be run on top of different OS without modification, e.g. Matlab. In fact, none of the repositories that use this language mention the OS or any other dependency. In any case, none of them specifies the Matlab version neither, which may cause improper functioning, as will be presented later.

For the rest of specifications (middleware, programming language, dependencies) the information is not complete neither: there are 14 works that use a middleware (ROS in all but one case) but only half of them specify the ROS distribution (Indigo). The version of the programming language is mentioned in few cases too: only one 1 of 8 cases for the Python language, and 4 of 18 cases for the C++ language. Libraries are mostly mentioned without version: for OpenCV, only 4 of 9 works specify some version, but one of them is incorrect (2.11, which probably refers to 2.4.11).

B. Statistics of Repositories

Some information extracted from the repositories is analyzed: according to the time of the last update depicted in Fig. 1.a, 45% of the repositories have not been updated in the last year, and other 25% have not been updated in the last six months.

One indicator of the success in sharing a code repository would be the number of clones (downloads), but the statistic is only available to the administrators of the repository. A public indicator is its number of forks, or copies of the repository, shown in Fig.1.b: more than half of the analyzed repositories (61.3%) have less than 10 forks.

The third plot, depicted in Fig. 1.c, combines both measurements: it represents the histogram of update time in three intervals (days, tens of days, hundreds of days) for three different classes of repositories, depending on their number of forks (units, tens, hundreds). Interestingly, this plot indicates that highly-shared repositories are most likely to be updated: since more people is using the code, it makes sense that bugs are detected, and requests for update are sent to the developer.

Nevertheless, the main conclusion is that a majority of code repositories are not frequently updated, nor shared actively by other users.

IV. Methodology

Software installation is a time-consuming process. New software may generate conflicts with existing applications, and compatibility issues may arise between the requisites and the installed configuration.

Isolation between the tested software and the host machine is a desirable feature. In this study, we rely on Docker, a Linux container technology [50] for creating isolated execution environments. Containers can be regarded as lightweight virtual machines, which run on top of the host operating system, without the need of a native installation.

Docker is gaining popularity for constructing repeatable and reproducible environments, enabling any user to run and ship portable applications [41].

Figure 2 depicts our proposal for the distribution of a readily-executable version of the code in the ICRA papers: the source code repositories are forked, generating new snapshots of the software. Based on these snapshots, Docker images are automatically generated. A web index is build for easy accessing the code repository[4]. Any user can download and run an image in his/her computer.

There is an exception in this framework: repositories of Matlab code are not executed in a Docker image, but directly on a Matlab native install. Though it can be installed in a Docker image, Matlab is a proprietary platform, hence a license is necessary and the software cannot be redistributed without permission.

A. The Matlab platform

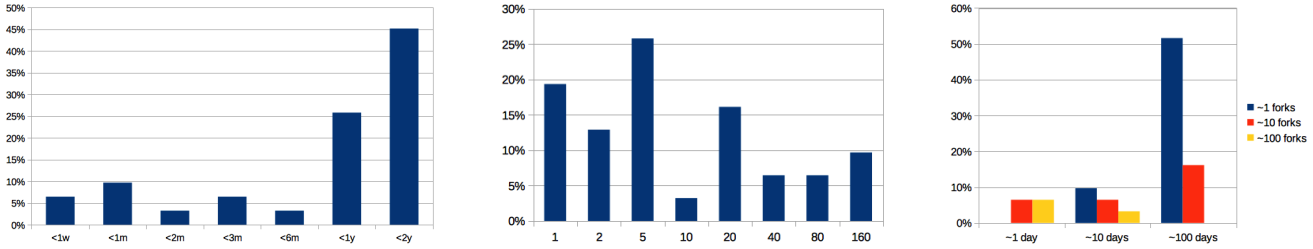
Matlab is a software for engineering and mathematics, working on different Operating Systems, which is extensively used in robotics [51][52].

We have tested the ICRA source code with the following Matlab configuration:

- Matlab 8.5.0.197613 (R2015a)
- 64-bit
- Ubuntu 14.04.5

Unfortunately, the code of the five papers that use Matlab cannot be successfully executed, as listed in Table II.

[4]https://icra2017.github.io/



(a) Last update (b) Number of forks (c) Updates and forks

Fig. 1: Histograms of repositories based on (a) the time since the last update, (b) the number of forks, and (c) the order of magnitude of updates and forks.

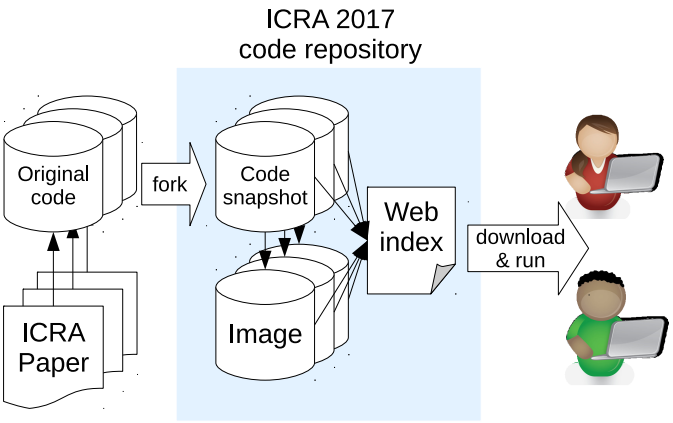


Fig. 2: Architecture of the proposed code repository, consisting of archived forks of the original code, Docker images, and a web index for easy access.

TABLE II: Problems in the execution of Matlab code.

Repository	Problem	Fix
Self-triggered-mechanism [1]	No instructions	Yes
PGO-Laginit [2]	No instructions	No
Fast-SeqSLAM [3]	No instructions	Yes
EKF-SLAM-on-Manifold [4]	`Undefined function or variable 'so3_log'`	Yes
LT-Algorithm [5]	`Undefined function or variable 'circ_dist'`	Yes

In three cases, the instructions in the repositories do not explain how to run the code. In other two cases, the execution was interrupted by the runtime errors listed in Table II.

After some debugging, we have succeeded in fixing four of the five repositories: some demonstration scripts were identified in [1][3]; a proper definition of the path was necessary in [4]; and a missing dependency (Circular Statistics Toolbox) was installed in [5].

The remaining repository [2] consists of the code of a function, without any demonstration script or any other information about how to run it.

The fixed repositories with examples, including figures, are available at https://icra2017.github.io/.

B. The Docker platform

Building a Docker image requires the definition of a Dockerfile, a text document that contains all the commands a user could call on the command line to assemble an image.

Docker images are not built from scratch, but from standard base images, e.g. Debian, Ubuntu, or ROS installs.

On top of the base image, the user installs the libraries and other dependencies, then compiles and builds the code, and the result is a complete image that can be executed on the Docker runtime in any supported host (Windows, Linux, Mac). An example of Dockerfile is shown in Fig. 3.

```
9 ■■■■■ Dockerfile
...  ...  @@ -0,0 +1,9 @@
          1  +FROM ubuntu:14.04
          2  +
          3  +RUN apt-get update && apt-get install -y \
          4  +       build-essential git \
          5  +       && rm -rf /var/lib/apt/lists
          6  +
          7  +RUN git clone https://github.com/ICRA2017/MSGD.git
          8  +
          9  +RUN cd MSGD && make all
```

Fig. 3: Dockerfile for the MSGD repository [14].

Automated building of Docker images from GitHub can be configured, in such a way that a commit to the code repository triggers the generation of the Docker image. The success or fail of the process is displayed automatically with badges, enabling the user to continuously monitor the proper development of the software cycle.

Upon success, the image is stored in the DockerHub cloud for further downloading by users[5]. Figure 4 shows an example of execution in a terminal: the image `icra2017/msgd` is downloaded and executed; inside the container the commands for running the demonstration are executed, and the results are generated.

It must be emphasized that the user only needs to install the Docker software in her computer, which is free and available in Windows, Mac, and Linux[6]. All the libraries and research code

[5]https://hub.docker.com/u/icra2017/
[6]https://www.docker.com/community-edition

```
ecervera — root@355080b20f7b: /MSGD — docker run -it --rm icra2017/msgd — 80×49
khiraki:~ ecervera$ docker run -it --rm icra2017/msgd
Unable to find image 'icra2017/msgd:latest' locally
latest: Pulling from icra2017/msgd
28bfaceaff9b: Already exists
ac540055f2f8: Already exists
2965585ef8b8: Already exists
2416bb9f3ad2: Already exists
93b55a6a6807: Already exists
5cd5532da043: Pull complete
b61db151b0db: Pull complete
6f24d1460649: Pull complete
Digest: sha256:f375f16c2fa2747b0e798ef10a7cd365e1e47c9a58515d2040abcabb8ec59772
Status: Downloaded newer image for icra2017/msgd:latest
root@355080b20f7b:/# cd MSGD
root@355080b20f7b:/MSGD# mkdir result_folder
root@355080b20f7b:/MSGD# ./msgd -in ./datasets/WGB2a-1.g2o -out result_folder/

MSGD Copyright (C) 2015-2017 Chao Gao, University of Cambridge.
This program comes with ABSOLUTELY NO WARRANTY;
This is free software, and you are welcome to redistribute it
under certain conditions. See LICENSE.txt.

*****************************************************************
 Input File                                    = ./datasets/WGB2a-1.g2o
 Output Folder                                 = result_folder/
 Iterations                                    = 100
 Show Progress                                 = No
 Enable Edge Randomnization                    = Yes
 Set Constraint Covariances to Idendity Matrix = No
 Save Result Graph For Every Iteration         = No
*****************************************************************

Load data from file ...
Done
Number of Poses: 26262
Number of Constraints: 10382
Number of Mini-batches: 29
Saving graph: initial
Done.

Before Optimization:  global error = 2.68694e+06   error/constraint = 258.808
**** Optimization Start ****
**** Optimization Done ****
After Optimization:  global error = 300477   error/constraint = 28.9421
Total Time = 1.79049 s.
Saving graph: final
Done.

root@355080b20f7b:/MSGD#
```

Fig. 4: Execution of MSGD code [14] in Docker.

runs inside the container, transparently, without interferences with the host.

V. PROTOCOL AND RESULTS

We consider a protocol of three phases in the process of executing the code in a repository. Each of these phases has to be adequately documented for allowing the correct installation and execution of the code:

1. Prerequisites Setting up the operating system and installing the library dependencies.

2. Build Building the software, i.e. compiling, linking, and generating a executable program.

3. Run Running the executable.

In the repositories of this study, some phases are more documented than others: only 30% of the repositories include the instructions for installing the prerequisites; 60% of the works include building instructions; and only 33% explain how to run the software.

According to our review, there are only 4 works [14][16][17][22] providing the interested reader with all the necessary information.

On the other side, half of the published repositories can be extremely hard to use due to the partial documentation. In fact, in a few cases, there is absolutely no information at all [18][30].

Figure 5 depicts the result for each repository. Those with insufficient documentation are marked in yellow: in most cases, a library dependency is mentioned but the installation process is not described, and only a link to the library developer is included. In many cases, installing a library is not straightforward, as it may have additional dependencies. In addition, the choice of version is not trivial since the success of the building procedure may depend on the use of a specific version.

The existence of instructions is a necessary yet not sufficient condition for a successful building process. The experimental software is frequently updated, and the documentation may lag behind the latest software changes.

In addition, some aspects may not be documented because they are taken for granted in the developer's platform. In many repositories using ROS, the operating system is not indicated, possibly because the default option (Ubuntu) is used. The ROS distribution is not specified neither: in this case, one could assume that a Long-Term Support (LTS) distribution is used, but such distributions are launched on a 2-year basis.

In summary, the building process is likely to fail due to the differences between the platforms of the code developer and the user. Figure 5 depicts a diagram of the progress in the test of the repositories: 8 repositories failed to build, and only in 1 of 3 repositories a documented example could be run without errors.

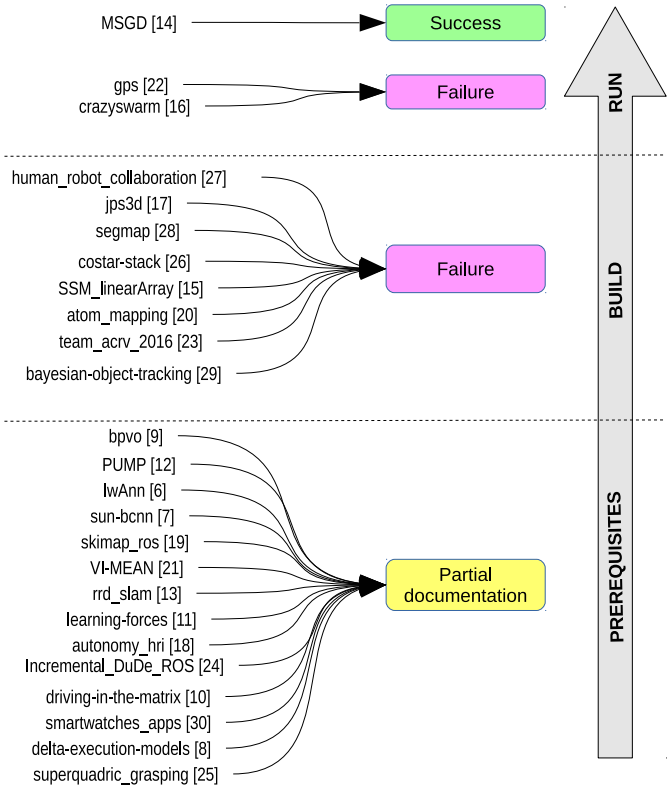


Fig. 5: Setup and execution phases: result of each repository.

The errors in the building and running process are listed in Table III. In most cases, the error is produced during the configuration process (Cmake) because a library dependency was not documented. In two other cases (costar-stack, team_acrv_2016), the error is caused by a wrong order of the installation instructions.

Regarding the runtime errors, in the first case [22] some running dependencies for the graphical output (PyQt4) were not mentioned in the documentation, hence not installed. Moreover, the default version of one of the libraries (matplotlib) had some deprecated functions that caused a run error. For fixing this issue, an older version (1.5.3) needs to be installed.

In the second case [16] the name of the file in the documentation example did not correspond to any file in the installed source code. The issue can be solved by replacing it with the most similar file name in the folder (`figure8_csv.py` instead of `figure8_canned.py`).

TABLE III: Errors in the build and run phases.

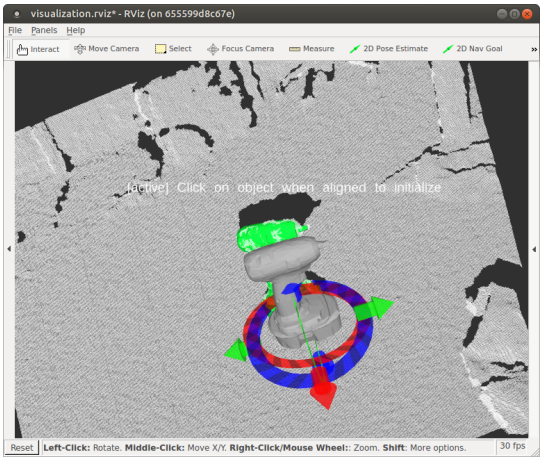
Repository	Problem	Fix
gps [22]	`ImportError: No module named PyQt4`	Yes
crazyswarm [16]	`Python: no such file`	Yes
human_robot_collaboration [27]	`apt: command [apt-get install -y ros-indigo-tf-conversions] failed`	No
jps3d [17]	`fatal error: Eigen/Geometry: No such file`	Yes
segmap [28]	`[CMakeFiles/segmatch_ros .dir/src/ segmatch_worker.cpp.o] Error 1`	No
costar-stack [26]	`Cannot locate rosdep definition for [objrecransac]`	No
SSM_linearArray [15]	`Cmake error: Could NOT find CSPARSE`	Yes
atom_mapping [20]	`CMake error: Could not find PCL`	Yes
team_acrv_2016 [23]	`~/ros_ws/src/apc_docs/ pcl_patch: No such file or directory`	No
bayesian-object-tracking [29]	`Cmake error: could not find cv_bridge`	Yes

Figure 6 shows an example of graphical user interaction in a Docker running image of the repositories `bayesian-object-tracking`[29] and `skimap-ros`[19]. For example, the user can move a marker for the initialization of the tracker. Such interaction is not possible in cloud-based environments like Code Ocean[7].

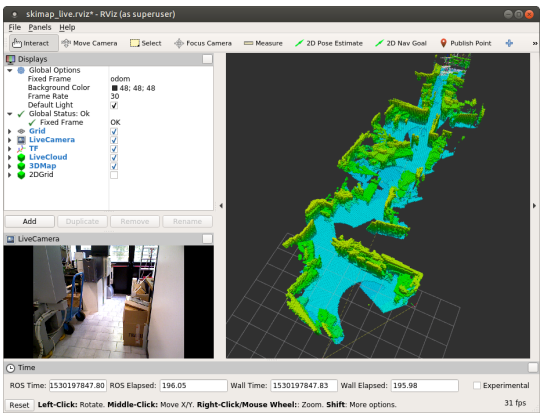
Regarding the partially documented code, we have succeeded in building working images for 5 repositories listed in Table IV. For each repository, we have chosen an appropriate base image in order to ease the installation process. The choices for available images is immense, ranging from Java to CUDA environments, and different ROS distributions. Our choice was guided by the available documentation, our previous knowledge, and trial-and-error.

As for the repositories not yet working, Table V summarizes the current situation. In two cases, the development platforms (Android, Windows) are not supported yet. The other repositories fail due to wrong or missing dependencies, which cause compilation and linking errors.

[7]https://codeocean.com/



(a) `bayesian-object-tracking`[29]



(b) `skimap-ros`[19]

Fig. 6: RViz tool running in Docker with the code of two repositories.

TABLE IV: Working images for the partially documented repositories.

Repository	Base Image
bpvo [9]	ubuntu:14.04
PUMP [12]	nvidia/cuda:8.0-cudnn6-devel-ubuntu16.04
lwAnn [6]	openjdk:7
skimap_ros [19]	nvidia/opengl:1.0-glvnd-devel-ubuntu16.04
Incremental_DuDe_ROS [24]	ros:indigo-perception

VI. CONCLUSION

Documentation is one of the main concerns in software engineering: the three severest problems are ambiguity, incompleteness, and incorrectness of content [53]. Robotics software is no exception, and the main conclusion of this study is that most of the failures are caused by incomplete or inconsistent documentation of the software.

TABLE V: Ongoing work on images for the partially documented repositories.

Repository	Problem
sun-bcnn [7]	`/opt/caffe/include/caffe/util/cudnn.hpp(112): error: too few arguments in function call`
VI-MEAN [21]	`chisel_ros: Missing resource pcl`
rrd_slam [13]	`[CMakeFiles/lsdslam.dir/src/DataStructures/Frame.cpp.o] Error 1`
learning-forces [11]	`make: *** No targets specified and no makefile found. Stop.`
autonomy_hri [18]	`Could not find the required component 'hark_msgs'`
driving-in-the-matrix [10]	`make: *** No rule to make target '/root/mxnet/ps-lite/make/ps.mk'`
smartwatches_apps [30]	Android platform
delta-execution-models [8]	Windows platform
superquadric_grasping [25]	`CMakeFiles/superquadric-grasping.dir/build.make:96: recipe for target 'CMakeFiles/superquadric-grasping.dir/src/main.cpp.o' failed`

The fact that only one code repository could be built and run off-the-shelf is discouraging. The good news is that in most cases the software can be fixed, but the required time for finding a solution can increase exponentially with the complexity of the system.

We have succeeded in building and executing the software in roughly half of the cases (16 of 30 repositories — 4 of 5 with Matlab, 12 of 25 in other cases). In the end, we believe that all the repositories can be fixed but our time and human resources were limited. Feedback from the authors would help, specially for obtaining datasets and better defining the runtime environments. The endeavor is open and collaboration from the community is indeed necessary and welcome.

Nevertheless, our automated workflow for creating Docker images of the source code repositories, with all the required dependencies, allows any interested user to execute the code without the need of a time-consuming installation and configuration procedure.

The repositories of source code and Docker images are freely available at https://icra2017.github.io/.

While the presented experiment has demonstrated that Docker is a great technology for reproducing experiments in academic settings, it is not a replacement for proper software packaging technologies (apt, homebrew, pip, ...). Indeed, by means of such technologies, the installation of several packages is straightforward, yet combining two or more Docker images in a single one is not a trivial task.

The outcome of our work is two-fold: 1) it has signaled the deficient specifications of most published source code in a prestigious conference like ICRA, and 2) it has proven the feasibility of Docker (a well-established technology in industry) for running robotics software in a wide variety of academic settings.

In future steps, we aim to make further experiments for quantifying the time effort invested in the reproducibility of experiments, and the user satisfaction of docker-based vs. native approaches.

REFERENCES

[1] L. Zhou and P. Tokekar, "Active target tracking with self-triggered communications," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2117–2123.

[2] J. Briales and J. Gonzalez-Jimenez, "Initialization of 3D pose graph optimization using lagrangian duality," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5134–5139.

[3] S. M. Siam and H. Zhang, "Fast-SeqSLAM: A fast appearance based place recognition algorithm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5702–5708.

[4] T. Zhang, K. Wu, J. Song, S. Huang, and G. Dissanayake, "Convergence and consistency analysis for a 3-D Invariant-EKF SLAM," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 733–740, April 2017.

[5] V. H. Bennetts, T. P. Kucner, E. Schaffernicht, P. P. Neumann, H. Fan, and A. J. Lilienthal, "Probabilistic air flow modelling using turbulent and laminar characteristics for ground and aerial robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1117–1123, April 2017.

[6] J. Young, L. Kunze, V. Basile, E. Cabrio, N. Hawes, and B. Caputo, "Semantic web-mining and deep vision for lifelong object discovery," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2774–2779.

[7] V. Peretroukhin, L. Clement, and J. Kelly, "Reducing drift in visual odometry by inferring sun direction using a bayesian convolutional neural network," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2035–2042.

[8] A. Mitrevski, A. Kuestenmacher, S. Thoduka, and P. G. Plöger, "Improving the reliability of service robots in the presence of external faults by learning action execution models," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4256–4263.

[9] H. Alismail, M. Kaess, B. Browning, and S. Lucey, "Direct visual odometry in low light using binary descriptors," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 444–451, April 2017.

[10] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 746–753.

[11] Z. Erickson, A. Clegg, W. Yu, G. Turk, C. K. Liu, and C. C. Kemp, "What does the person feel? learning to infer applied forces during robot-assisted dressing," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 6058–6065.

[12] B. Ichter, E. Schmerling, A. a. Agha-mohammadi, and M. Pavone, "Real-time stochastic kinodynamic motion planning via multiobjective search on gpus," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5019–5026.

[13] J. H. Kim, Y. Latif, and I. Reid, "RRD-SLAM: Radial-distorted rolling-shutter direct SLAM," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5148–5154.

[14] C. Gao and R. Harle, "MSGD: Scalable back-end for indoor magnetic field-based GraphSLAM," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3855–3862.

[15] D. Su, T. Vidal-Calleja, and J. V. Miro, "Towards real-time 3D sound sources mapping with linear microphone arrays," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1662–1668.

[16] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3299–3304.

[17] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, July 2017.

[18] S. Pourmehr, J. Thomas, J. Bruce, J. Wawerla, and R. Vaughan, "Robust sensor fusion for finding hri partners in a crowd," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3272–3278.

[19] D. D. Gregorio and L. D. Stefano, "Skimap: An efficient mapping framework for robot navigation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2569–2576.

[20] D. Fridovich-Keil, E. Nelson, and A. Zakhor, "Atommap: A probabilistic amorphous 3D map representation for robotics and surface reconstruction," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3110–3117.

[21] Z. Yang, F. Gao, and S. Shen, "Real-time monocular dense mapping on aerial robots using visual-inertial fusion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4552–4559.

[22] W. Montgomery, A. Ajay, C. Finn, P. Abbeel, and S. Levine, "Reset-free guided policy search: Efficient deep reinforcement learning with stochastic initial states," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3373–3380.

[23] J. Leitner, A. W. Tow, N. Sünderhauf, J. E. Dean, J. W. Durham, M. Cooper, M. Eich, C. Lehnert, R. Mangels, C. McCool, P. T. Kujala, L. Nicholson, T. Pham, J. Sergeant, L. Wu, F. Zhang, B. Upcroft, and P. Corke, "The acrv picking benchmark: A robotic shelf picking benchmark to foster reproducible research," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4705–4712.

[24] L. Fermin-Leon, J. Neira, and J. A. Castellanos, "Incremental contour-based topological segmentation for robot exploration," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2554–2561.

[25] G. Vezzani, U. Pattacini, and L. Natale, "A grasping approach based on superquadric models," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1579–1586.

[26] C. Paxton, A. Hundt, F. Jonathan, K. Guerin, and G. D. Hager, "Costar: Instructing collaborative robots with behavior trees and vision," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 564–571.

[27] A. Roncone, O. Mangin, and B. Scassellati, "Transparent role assignment and task allocation in human robot collaboration," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1014–1021.

[28] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based place recognition in 3D point clouds," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5266–5272.

[29] C. G. Cifuentes, J. Issac, M. Wüthrich, S. Schaal, and J. Bohg, "Probabilistic articulated real-time tracking for robot manipulation," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 577–584, April 2017.

[30] E. Coronado, J. Villalobos, B. Bruno, and F. Mastrogiovanni, "Gesture-based robot control: Design challenges and evaluation with humans," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2761–2767.

[31] F. Bonsignorio and A. P. del Pobil, "Toward replicable and measurable robotics research," *IEEE Rob. Aut. Mag.*, vol. 22, no. 3, 2015.

[32] S. Cousins, B. Gerkey, and K. Conley, "Sharing software with ROS [ROS topics]," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 12–14, 2010.

[33] G. Metta, P. Fitzpatrick, and L. Natale, "YARP: yet another robot platform," *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, p. 8, 2006.

[34] H. Bruyninckx, "Open robot control software: the OROCOS project," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3. IEEE, 2001, pp. 2523–2528.

[35] F. Lier, J. Wienke, A. Nordmann, S. Wachsmuth, and S. Wrede, "The cognitive interaction toolkit–improving reproducibility of robotic systems experiments," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014, pp. 400–411.

[36] F. Lier, M. Hanheide, L. Natale, S. Schulz, J. Weisz, S. Wachsmuth, and S. Wrede, "Towards automated system and experiment reproduction in robotics," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 3298–3305.

[37] A. Pörtner, M. Hoffmann, and M. König, "Swarmrob: A toolkit for reproducibility and sharing of experimental artifacts in robotics research," *arXiv preprint arXiv:1801.04199*, 2018.

[38] F. Bonsignorio, "A new kind of article for reproducible research in intelligent robotics [from the field]," *IEEE Robotics Automation Magazine*, vol. 24, no. 3, pp. 178–182, Sept 2017.

[39] T. Foote, D. Thomas, D. Pangercic, D. Di Marco, and A. Hamann, "Docker-based build farm for ROS," in *ROSCon*, 2015.

[40] C. Boettiger, "An introduction to docker for reproducible research," *ACM SIGOPS Op Sys Rev*, vol. 49, no. 1, pp. 71–79, 2015.

[41] R. White and H. Christensen, "ROS and docker," in *Robot Operating System (ROS)*. Springer, 2017, pp. 285–307.

[42] V. Dugar, S. Choudhury, and S. Scherer, "A kite in the wind: Smooth trajectory optimization in a moving reference frame," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 109–116.

[43] S. Choudhury, A. Kapoor, G. Ranade, and D. Dey, "Learning to gather information via imitation," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 908–915.

[44] K. A. Skinner, E. Iscar, and M. Johnson-Roberson, "Automatic color correction for 3D reconstruction of underwater scenes," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5140–5147.

[45] M. Fehr, F. Furrer, I. Dryanovski, J. Sturm, I. Gilitschenski, R. Siegwart, and C. Cadena, "Tsdf-based change detection for consistent long-term dense reconstruction and dynamic object discovery," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5237–5244.

[46] B. Pfrommer, N. Sanket, K. Daniilidis, and J. Cleveland, "Penncosyvio: A challenging visual inertial odometry benchmark," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3847–3854.

[47] D. Pickem, P. Glotfelter, L. Wang, M. Mote, A. Ames, E. Feron, and M. Egerstedt, "The robotarium: A remotely accessible swarm robotics research testbed," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1699–1706.

[48] L. Paull, J. Tani, H. Ahn, J. Alonso-Mora, L. Carlone, M. Cap, Y. F. Chen, C. Choi, J. Dusek, Y. Fang, D. Hoehener, S. Y. Liu, M. Novitzky, I. F. Okuyama, J. Pazis, G. Rosman, V. Varricchio, H. C. Wang, D. Yershov, H. Zhao, M. Benjamin, C. Carr, M. Zuber, S. Karaman, E. Frazzoli, D. D. Vecchio, D. Rus, J. How, J. Leonard, and A. Censi, "Duckietown: An open, inexpensive and flexible platform for autonomy education and research," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1497–1504.

[49] A. Spek and T. Drummond, "Joint pose and principal curvature refinement using quadrics," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3968–3975.

[50] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[51] P. I. Corke, "A robotics toolbox for matlab," *IEEE Robotics & Automation Magazine*, vol. 3, no. 1, pp. 24–32, 1996.

[52] P. Corke, "Matlab toolboxes: robotics and vision for students and teachers," *IEEE Robotics & automation magazine*, vol. 14, no. 4, 2007.

[53] G. Uddin and M. P. Robillard, "How API documentation fails," *IEEE Software*, vol. 32, no. 4, pp. 68–75, July 2015.