From Research Code to Running Systems

Cross-Platform Robotics and ML Workflows

Tobias Fischer
Peter Corke

From Research Code to Running Systems

Robotics papers increasingly depend on full software stacks

Bucket list:

- ROS, Python, C++, CUDA, simulators, ML tooling, and data
- Cross-platform robotics and ML workflows (Linux + Windows + MacOS + ...)
- Support for many programming languages: Python, C++, Rust, R, ...

From Research Code to Running Systems

Robotics papers increasingly depend on full software stacks

Bucket list:

- ROS, Python, C++, CUDA, simulators, ML tooling, and data
- Cross-platform robotics and ML workflows (Linux + Windows + MacOS + ...)
- Support for many programming languages: Python, C++, Rust, R, ...

Today: make those systems runnable, repeatable, and portable.

The Problem: Research Code Becomes Infrastructure

▓▓▓ Hands up if you have lost time to...

- a missing dependency
- a mystery dataset or model download
- "works on my machine"

The Problem: Research Code Becomes Infrastructure

▓▓ Hands up if you have lost time to...

- a missing dependency
- a mystery dataset or model download
- "works on my machine"

▓▓ The thesis

Tooling is part of how research is shared, reused, and extended. Developing and sharing should be a single workflow, not separated.

The Reality of Many Research Repositories

```
git clone https://github.com/some_repo/project

# Ubuntu 20.04 only
sudo apt install ...

pip install torch==1.12
pip install some_package

# Compile custom OpenCV

# Download model manually:
https://drive.google.com/...

# Tested on my machine :)
```

The Reality of Many Research Repositories

```
git clone https://github.com/some_repo/project

# Ubuntu 20.04 only
sudo apt install ...

pip install torch==1.12
pip install some_package

# Compile custom OpenCV

# Download model manually:
https://drive.google.com/...

# Tested on my machine :)
```

What if the README had one command?

This Talk Is Executable

The slides are also the demo environment.

▒▒▒ During the talk

- arrows: navigate
- control + e: run code
- control + r: reset slide
- ?: show keybindings
- esc or control + c: exit

▒▒▒ Running it yourself

Use a terminal and raw git clone (avoid GitHub Web or GitHub Desktop):

```
git clone https://github.com/petercorke/
          ICRA2026-modern-software-tools.git
cd ICRA2026-modern-software-tools
pixi run presentation
```

This Talk Is Executable

The slides are also the demo environment.

During the talk

- arrows: navigate
- control + e: run code
- control + r: reset slide
- ?: show keybindings
- esc or control + c: exit

Running it yourself

Use a terminal and raw git clone (avoid GitHub Web or GitHub Desktop):

```
git clone https://github.com/petercorke/
          ICRA2026-modern-software-tools.git
cd ICRA2026-modern-software-tools
pixi run presentation
```

Only run executable snippets from presentations you trust!

Pixi: From Instructions to Artifacts ⚡

Pixi is a fast project manager built on the conda-forge ecosystem.

Think: conda packages, modern project workflow.

Pixi: From Instructions to Artifacts ⚡

Pixi is a fast project manager built on the conda-forge ecosystem.

Think: conda packages, modern project workflow.

▒▒ The old way: Instructions

- README.md
- shell scripts
- oral tradition
- "ask the previous student"

▒▒ Now: shared artifacts

- pixi.toml
- pixi.lock
- pixi run <task>
- CI-friendly commands

> The shift is from instructions to runnable artifacts.

Start a Project

```
pixi init

————————————————— [finished] —————————————————

✔ Created
/home/runner/work/ICRA2026-modern-software-tools/ICRA2026-modern-softwar
e-tools/tobi/pixi.toml
```

Start a Project

```
pixi init
```

```
————————————————— [finished] —————————————————
```

```
✔ Created
/home/runner/work/ICRA2026-modern-software-tools/ICRA2026-modern-softwar
e-tools/tobi/pixi.toml
```

Add PyTorch

```
pixi add python pytorch torchvision
```

```
————————————————— [finished] —————————————————
```

```
✔ Added python >=3.14.5,<3.15
✔ Added pytorch >=2.11.0,<3
✔ Added torchvision >=0.26.0,<0.27
```

Fast Environments Change Behavior ⚡

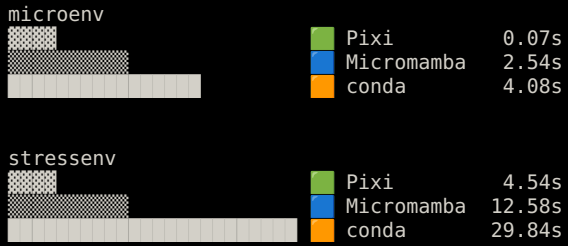
If changing environments takes minutes, people avoid changing environments.

Fast Environments Change Behavior ⚡

If changing environments takes minutes, people avoid changing environments.

If it takes seconds, environments become part of iteration.

Environment Solve Time



train.py

```python
import torch
import torch.nn.functional as F

print("PyTorch version:", torch.__version__)
print("CUDA available:", torch.cuda.is_available())
print("MPS available:", torch.backends.mps.is_available())

y = torch.tensor([1.0]); x = torch.tensor([1.1])  # target and input
w = torch.tensor([2.2], requires_grad=True)
b = torch.tensor([0.0], requires_grad=True)

z = x * w + b; a = torch.sigmoid(z)
loss = F.binary_cross_entropy(a, y)
loss.backward()

print("loss:", loss.item())
```

[finished]

```
PyTorch version: 2.11.0
CUDA available: False
MPS available: False
loss: 0.0851878821849823
```

Tasks Turn Commands into Workflows

Hands up: who has a command in their lab that nobody wants to type from memory?

Tasks Turn Commands into Workflows

Hands up: who has a command in their lab that nobody wants to type from memory?

▓▓▓ Define the workflow

```
pixi task add start "python train.py"
--depends-on download-mnist
```

```
──────────────────── [finished] ────────────────────
```

```
✔ Added task `start`: python train.py,
depends-on = 'download-mnist'
```

Tasks Turn Commands into Workflows

Hands up: who has a command in their lab that nobody wants to type from memory?

▓▓▓ Define the workflow

```
pixi task add start "python train.py"
--depends-on download-mnist
```

```
──────────────────── [finished] ────────────────────
```

```
✔ Added task `start`: python train.py,
depends-on = 'download-mnist'
```

```
[tasks]
...
download-mnist = {
  cmd = "python -c
'torchvision.datasets.MNIST("data",
download=True)'",
  outputs = ["data/MNIST"]
}
```

Tasks Turn Commands into Workflows

Hands up: who has a command in their lab that nobody wants to type from memory?

▓▓ Define the workflow

```
pixi task add start "python train.py"
--depends-on download-mnist
```

```
──────────────────── [finished] ────────────────────
```

```
✔ Added task `start`: python train.py,
depends-on = 'download-mnist'
```

```
[tasks]
...
download-mnist = {
  cmd = "python -c
'torchvision.datasets.MNIST("data",
download=True)'",
  outputs = ["data/MNIST"]
}
```

▓▓ Run it

```
pixi run start
```

```
────────── [finished] ──────────
```

```
✨ Pixi task
(download-mnist): python -c
'from torchvision.datasets
import MNIST; MNIST("data",
download=True)' 2>/dev/null

✨ Pixi task (start): python
train.py
PyTorch version: 2.11.0
```

Tasks Turn Commands into Workflows

Hands up: who has a command in their lab that nobody wants to type from memory?

▓▓ Define the workflow

```
pixi task add start "python train.py"
--depends-on download-mnist
```

```
──────────────────── [finished] ────────────────────
```

```
✔ Added task `start`: python train.py,
depends-on = 'download-mnist'
```

```
[tasks]
...
download-mnist = {
  cmd = "python -c
'torchvision.datasets.MNIST("data",
download=True)'",
  outputs = ["data/MNIST"]
}
```

▓▓ Run it

```
pixi run start
```

```
────────── [finished] ──────────
```

```
✨ Pixi task
(download-mnist): python -c
'from torchvision.datasets
import MNIST; MNIST("data",
download=True)' 2>/dev/null

✨ Pixi task (start): python
train.py
PyTorch version: 2.11.0
```

▓▓ Run it again

```
pixi run start
```

```
────────── [finished] ──────────
```

Mixing Ecosystems

Modern robotics often needs:

Robotics stack

- ROS
- OpenCV
- CUDA
- native libraries

Mixing Ecosystems

Modern robotics often needs:

Robotics stack

- ROS
- OpenCV
- CUDA
- native libraries

ML stack

- PyTorch
- Transformers
- experiment tools
- Python packages

Mixing Ecosystems

Modern robotics often needs:

Robotics stack

- ROS
- OpenCV
- CUDA
- native libraries

ML stack

- PyTorch
- Transformers
- experiment tools
- Python packages

Conda-forge + PyPI Together

```toml
[dependencies]
python = "3.11"
pytorch = "*"

[pypi-dependencies]
transformers = "*"
```

Mixing Ecosystems

Modern robotics often needs:

Robotics stack

- ROS
- OpenCV
- CUDA
- native libraries

ML stack

- PyTorch
- Transformers
- experiment tools
- Python packages

Conda-forge + PyPI Together

```toml
[dependencies]
python = "3.11"
pytorch = "*"

[pypi-dependencies]
transformers = "*"
```

No boundary between ecosystems ✨

Robotics Meets ML: RoboStack

Hands up if you have ever installed a VM or old Ubuntu just to get ROS running.

Robotics Meets ML: RoboStack

Hands up if you have ever installed a VM or old Ubuntu just to get ROS running.

RoboStack enables ROS on:

- Linux
- macOS
- Windows

through the conda-forge ecosystem.

Robotics Meets ML: RoboStack

Hands up if you have ever installed a VM or old Ubuntu just to get ROS running.

RoboStack enables ROS on:

- Linux
- macOS
- Windows

through the conda-forge ecosystem.

```
pixi workspace channel add https://prefix.dev/robostack-rolling
pixi add ros-rolling-desktop
```

```
──────────────────────────── [finished] ────────────────────────────

✔ Added https://prefix.dev/robostack-rolling
 WARN Skipped running the post-link scripts because
`run-post-link-scripts` = `false`
    - bin/.librsvg-pre-unlink.sh

To enable them, run:
    pixi config set --local run-post-link-scripts insecure

More info:

https://pixi.sh/latest/reference/pixi_configuration/#run-post-link-scrip
ts
```

ROS Desktop App from a Locked Environment

```
pixi run ros2 run turtlesim turtlesim_node
```

ROS Desktop App from a Locked Environment

```
pixi run ros2 run turtlesim turtlesim_node
```

```
pixi run ros2 topic pub /turtle1/cmd_vel \
  geometry_msgs/msg/Twist \
  "{linear: {x: 2.0}, angular: {z: 1.8}}"
```

ROS Desktop App from a Locked Environment

```
pixi run ros2 run turtlesim turtlesim_node
```

```
pixi run ros2 topic pub /turtle1/cmd_vel \
  geometry_msgs/msg/Twist \
  "{linear: {x: 2.0}, angular: {z: 1.8}}"
```

Start with a tiny demo. Then scale the same idea to real stacks.

One Environment

ROS + PyTorch + OpenCV
+ Transformers + custom research code

- One Environment

  ROS + PyTorch + OpenCV
  + Transformers + custom research code

- One lockfile.

One Environment

ROS + PyTorch + OpenCV
+ Transformers + custom research code

One lockfile.

One Machine, Multiple ROS Distros

```
# Add another ROS distro
pixi workspace channel add https://prefix.dev/robostack-humble
pixi add --feature humble ros-humble-desktop

# Run different environments
pixi run -e humble   ros2 run turtlesim turtlesim_node
pixi run -e rolling  ros2 run rviz2 rviz2

# We still support ROS1 Noetic :)
```

Build Your Own ROS / C++ / Python Packages

▓▓▓ Create a ROS package

```
pixi run ros2 pkg create \
  --build-type ament_cmake \
  --node-name icra_node \
  icra_ros_package
```

```
──────────────────── [finished] ────────────────────
```

```
going to create a new package
package name: icra_ros_package
```

Build Your Own ROS / C++ / Python Packages

Create a ROS package

```
pixi run ros2 pkg create \
  --build-type ament_cmake \
  --node-name icra_node \
  icra_ros_package
```

```
──────────────────── [finished] ────────────────────

going to create a new package
package name: icra_ros_package
```

Add it to Pixi

```
[workspace]
preview = ["pixi-build"]

[dependencies]
ros-rolling-icra-ros-package = { path = "
icra_ros_package/package.xml" }
```

Build Your Own ROS / C++ / Python Packages

Create a ROS package

```
pixi run ros2 pkg create \
  --build-type ament_cmake \
  --node-name icra_node \
  icra_ros_package
```

```
──────────────────── [finished] ────────────────────

going to create a new package
package name: icra_ros_package
```

Add it to Pixi

```
[workspace]
preview = ["pixi-build"]

[dependencies]
ros-rolling-icra-ros-package = { path = "
icra_ros_package/package.xml" }
```

Build the package

```
# The equivalent to colcon
build!
pixi install
```

```
────────── [finished] ──────────

 ┌─ Running build for
recipe:
ros-rolling-icra-ros-package
-0.0.0-hb0f4dca_0
```

Build Your Own ROS / C++ / Python Packages

Create a ROS package

```
pixi run ros2 pkg create \
  --build-type ament_cmake \
  --node-name icra_node \
  icra_ros_package
```

```
──────────────────── [finished] ────────────────────
```

```
going to create a new package
package name: icra_ros_package
```

Add it to Pixi

```
[workspace]
preview = ["pixi-build"]

[dependencies]
ros-rolling-icra-ros-package = { path = "
icra_ros_package/package.xml" }
```

Build the package

```
# The equivalent to colcon
build!
pixi install
```

```
────────── [finished] ──────────
```

```
 ┌─ Running build for
recipe:
ros-rolling-icra-ros-package
-0.0.0-hb0f4dca_0
```

Does it work?

```
pixi run ros2 run
icra_ros_package icra_node
```

Build Your Own ROS / C++ / Python Packages

Create a ROS package

```
pixi run ros2 pkg create \
  --build-type ament_cmake \
  --node-name icra_node \
  icra_ros_package
```

```
──────────────────── [finished] ────────────────────
```

```
going to create a new package
package name: icra_ros_package
```

Add it to Pixi

```
[workspace]
preview = ["pixi-build"]

[dependencies]
ros-rolling-icra-ros-package = { path = "
icra_ros_package/package.xml" }
```

Build the package

```
# The equivalent to colcon
build!
pixi install
```

```
────────── [finished] ──────────
```

```
 ┌─ Running build for
recipe:
ros-rolling-icra-ros-package
-0.0.0-hb0f4dca_0
```

Does it work?

```
pixi run ros2 run
icra_ros_package icra_node
```

Beyond ROS

- pure CMake projects
- Python packages
- Rust crates
- git repositories

> Research code becomes installable, shareable infrastructure.

Cross Platform Reproducibility

- Same repository
- Same lockfile
- Same command
- Different machine
- Different OS - note: no root required!

Cross Platform Reproducibility

- Same repository
- Same lockfile
- Same command
- Different machine
- Different OS - note: no root required!

 macOS → Linux → HPC

```
# Add other platforms
pixi workspace platform add linux-64 win-64
```

Cross Platform Reproducibility

- Same repository
- Same lockfile
- Same command
- Different machine
- Different OS - note: no root required!

macOS → Linux → HPC

```
# Add other platforms
pixi workspace platform add linux-64 win-64
```

```
# Copy this demo to Linux/HPC via ssh/scp
# Runs ssh/scp only when ICRA_REMOTE_HOST is set
python scripts/remote_demo.py prepare
```

```
# Optional presenter remote: run the same Pixi task on Linux/HPC
# Prints each remote command before executing it
python scripts/remote_demo.py run
```

Case Study: VSLAM-Lab

Large-scale visual SLAM framework for benchmarking, composability, reproducibility, and easy onboarding.

▓▓ Before 😬

README.md

1. Install ROS, ...
2. Build OpenCV, ...
3. Download models
4. Download datasets
5. Configure CUDA
6. Build dependencies
7. Pray

Case Study: VSLAM-Lab

Large-scale visual SLAM framework for benchmarking, composability, reproducibility, and easy onboarding.

▓▓ Before 😬

README.md

1. Install ROS, ...
2. Build OpenCV, ...
3. Download models
4. Download datasets
5. Configure CUDA
6. Build dependencies
7. Pray

▓▓ After ✨

```
git clone
https://github.com/VSLAM-LAB/VSLAM-LAB
.git > /dev/null 2>&1
pixi run -m ./VSLAM-LAB/pixi.toml demo
orbslam2 eth table_3 mono
```

Case Study: VSLAM-Lab

Large-scale visual SLAM framework for benchmarking, composability, reproducibility, and easy onboarding.

▓▓ Before 😬

README.md

1. Install ROS, ...
2. Build OpenCV, ...
3. Download models
4. Download datasets
5. Configure CUDA
6. Build dependencies
7. Pray

▓▓ After ✨

```
git clone
https://github.com/VSLAM-LAB/VSLAM-LAB
.git > /dev/null 2>&1
pixi run -m ./VSLAM-LAB/pixi.toml demo
orbslam2 eth table_3 mono
```

> Multi-page setup instructions become executable workflows!

Why Pixi for Robotics? ⚙️

Built-in project feature	Pixi	Conda	Pip	Poetry	uv
Installs Python	✅	✅	❌	❌	✅
Native libraries	✅	✅	❌	❌	❌
Lockfiles	✅	via tools	❌	✅	✅
Task runner	✅	❌	❌	❌	❌
Workspace management	✅	❌	❌	✅	✅
Fast solver	✅	slower	n/a	❌	✅

Why Pixi for Robotics? ⚙️

Built-in project feature	Pixi	Conda	Pip	Poetry	uv
Installs Python	✅	✅	❌	❌	✅
Native libraries	✅	✅	❌	❌	❌
Lockfiles	✅	via tools	❌	✅	✅
Task runner	✅	❌	❌	❌	❌
Workspace management	✅	❌	❌	✅	✅
Fast solver	✅	slower	n/a	❌	✅

> The point is not one feature.

> The point is having the right combination for robotics.

Why Not Docker? 

▒▒ Containers are excellent for

- deployment
- cloud workflows
- CI/CD
- reproducible services
- production isolation

Dockerfile
apt-get
pip install
CUDA setup
X11 forwarding
volume mounts

Why Not Docker? 

▓▓ Containers are excellent for

- deployment
- cloud workflows
- CI/CD
- reproducible services
- production isolation

```
Dockerfile
apt-get
pip install
CUDA setup
X11 forwarding
volume mounts
```

▓▓ But robotics research often needs

- native GUI applications
- low-friction hardware access
- host networking and shared memory behavior
- ROS + ML composability
- rapid iteration across repositories
- cross-platform desktop workflows

```
pixi.toml
pixi.lock
pixi run start
```

Why Not Docker? 

▓▓ Containers are excellent for

- deployment
- cloud workflows
- CI/CD
- reproducible services
- production isolation

```
Dockerfile
apt-get
pip install
CUDA setup
X11 forwarding
volume mounts
```

▓▓ But robotics research often needs

- native GUI applications
- low-friction hardware access
- host networking and shared memory behavior
- ROS + ML composability
- rapid iteration across repositories
- cross-platform desktop workflows

```
pixi.toml
pixi.lock
pixi run start
```

▓▓ Our goal

Native, reproducible workflows with minimal setup friction.

Limitations and Trade-offs

Reproducible does not mean effortless. It means the effort is captured instead of rediscovered.

Limitations and Trade-offs

Reproducible does not mean effortless. It means the effort is captured instead of rediscovered.

1. Some robotics software is still Linux-first. Drivers, GUIs, middleware, and hardware access need testing.
2. Packaging moves work earlier. Old libraries and unusual builds likely need patches (our RoboStack builds have 100s!).
3. Pixi is one layer. Docker, HPC modules, and embedded toolchains still matter.
4. Lockfiles capture complexity. They do not make complex systems simple.

Limitations and Trade-offs

Reproducible does not mean effortless. It means the effort is captured instead of rediscovered.

1. Some robotics software is still Linux-first. Drivers, GUIs, middleware, and hardware access need testing.
2. Packaging moves work earlier. Old libraries and unusual builds likely need patches (our RoboStack builds have 100s!).
3. Pixi is one layer. Docker, HPC modules, and embedded toolchains still matter.
4. Lockfiles capture complexity. They do not make complex systems simple.

> The win is not zero setup.
> The win is setup you can inspect, share, and rerun.

Teasers

1. Browser-native ROS via **ROS2WASM** (*https://ros2wasm.dev/*)

Teasers

1. Browser-native ROS via **ROS2WASM** (*https://ros2wasm.dev/*)
2. Pack environments for another machine: pixi pack

Teasers

1. Browser-native ROS via **ROS2WASM** (*https://ros2wasm.dev/*)
2. Pack environments for another machine: pixi pack
3. **Cross-platform CI** (*https://github.com/ruben-arts/ros-example*)

```yaml
jobs:
  strategy:
    matrix:
      os: [ubuntu, windows, macos]
  steps:
    - name: Setup Pixi and install environment
      uses: prefix-dev/setup-pixi

    - name: Run ROS node
      run: pixi run ros2 pkg list
```

Teasers

1. Browser-native ROS via **ROS2WASM** (*https://ros2wasm.dev/*)

2. Pack environments for another machine: `pixi pack`

3. **Cross-platform CI** (*https://github.com/ruben-arts/ros-example*)

```yaml
jobs:
  strategy:
    matrix:
      os: [ubuntu, windows, macos]
  steps:
    - name: Setup Pixi and install environment
      uses: prefix-dev/setup-pixi

    - name: Run ROS node
      run: pixi run ros2 pkg list
```

4. Package missing dependencies (if it blocks your research, it blocks someone else's research, too!): `rattler-build generate-recipe pypi some-package`

Teasers

1. Browser-native ROS via **ROS2WASM** (*https://ros2wasm.dev/*)

2. Pack environments for another machine: `pixi pack`

3. **Cross-platform CI** (*https://github.com/ruben-arts/ros-example*)

```yaml
jobs:
  strategy:
    matrix:
      os: [ubuntu, windows, macos]
  steps:
    - name: Setup Pixi and install environment
      uses: prefix-dev/setup-pixi

    - name: Run ROS node
      run: pixi run ros2 pkg list
```

4. Package missing dependencies (if it blocks your research, it blocks someone else's research, too!): `rattler-build generate-recipe pypi some-package`

5. Package AI/ML models as versioned, cached dependencies: https://prefix.dev/blog/packaging-ai-ml-models-as-conda-packages

Key Insights

1. Environments are part of the method.
2. Tasks and lockfiles make projects runnable.
3. Packaging turns local code into shared infrastructure.
4. Research software is part of the contribution.

Key Insights

1. Environments are part of the method.
2. Tasks and lockfiles make projects runnable.
3. Packaging turns local code into shared infrastructure.
4. Research software is part of the contribution.

> Build systems that others can run, trust, extend, and build upon.

Key Insights

1. Environments are part of the method.
2. Tasks and lockfiles make projects runnable.
3. Packaging turns local code into shared infrastructure.
4. Research software is part of the contribution.

> Build systems that others can run, trust, extend, and build upon.

▓▓ Thank you

Tobias.Fischer@qut.edu.au & Peter.Corke@qut.edu.au

QUT Centre for Robotics

Thanks to the **prefix.dev** team for creating Pixi, **Silvio Traversaro**, **Alejandro Fontan**, **Nicolas Marticorena**, **Margaux Edwards**, my **co-authors**, open-source contributors, and the Australian Research Council.

▓▓ Questions?

References and Links

- **pixi.sh** (*https://pixi.sh*)
- **A RoboStack Tutorial: Using the Robot Operating System Alongside the Conda and Jupyter Data Science Ecosystems** (*https://doi.org/10.1109/MRA.2021.3128367*), IEEE Robotics & Automation Magazine, vol. 29, no. 2, June 2022
- **Pixi: Unified Software Development and Distribution for Robotics and AI** (*https://arxiv.org/abs/2511.04827*), arXiv:2511.04827
- **ROS2WASM: Bringing the Robot Operating System to the Web** (*https://doi.org/10.1109/ICRA55743.2025.11127821*), IEEE International Conference on Robotics and Automation (ICRA) 2025
- **VSLAM-LAB: A Comprehensive Framework for Visual SLAM Methods and Datasets** (*https://arxiv.org/abs/2504.04457*), IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2025
- **rattler.build** (*https://rattler.build*)