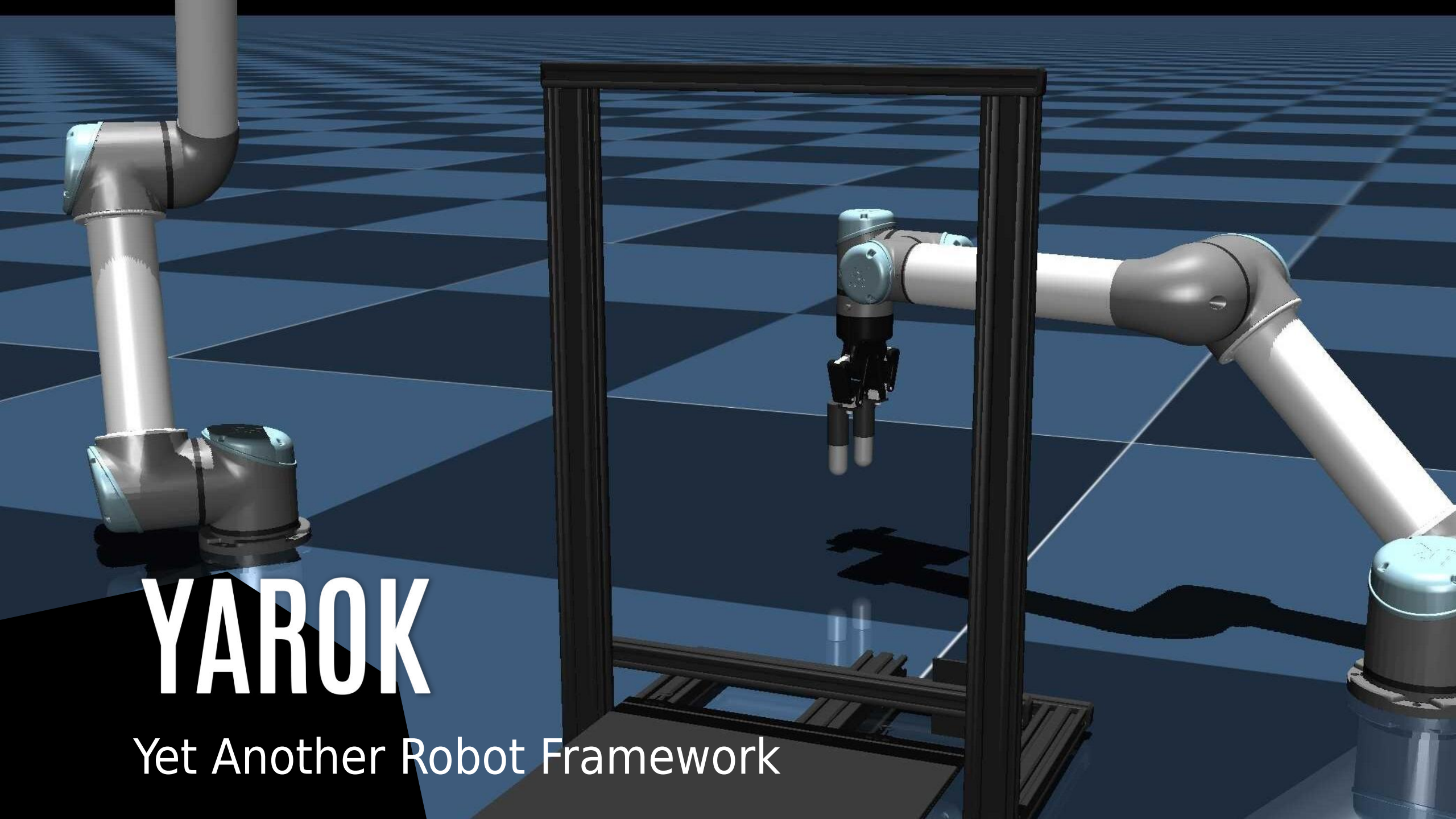
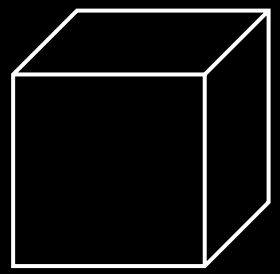
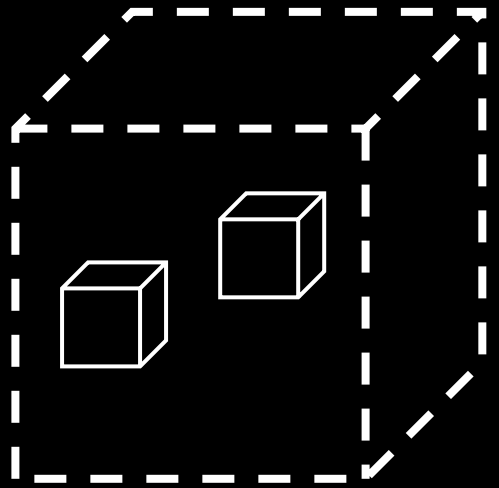
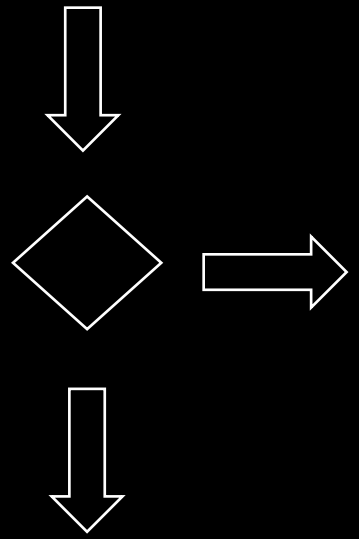

YAROK
Yet Another Robot Framework



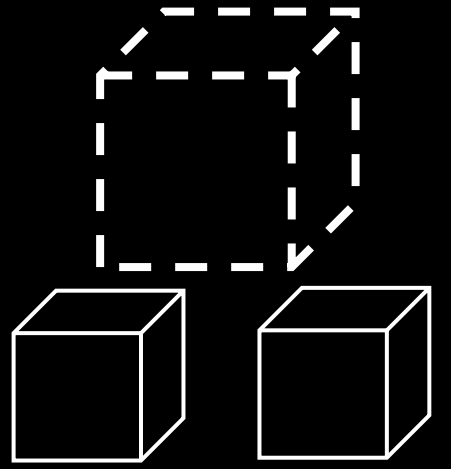
Python Components (template + controller)



Flexible composing mechanism



Extended MJCF: flexible composing, control flow macros



Code once, run everywhere (sim and real hardware)

```
@component(
    tag="my-robot-arm",
    defaults={
        'interface_mjc': MyRobotArmMJC,
        'interface_hw': MyRobotArmHW,
    }
    template= ...
class MyComponent:
        ...
```

A Component is a Python class with an MJCF template, interfaces for handling sim and real hardware, and config variables.

```python
from my_shared_components import MyGripper, MyTouchSensor

components=[
    MyGripper, MyTouchSensor
]
template="""
    <my-gripper name="gripper1">
        <my-sensor name="sensor1" parent="finger1_tip"></my-sensor>
    </my-gripper>
    ...
"""
class MyComponent:

    def __init__(gripper1: MyGripper, sensor1: MySensor):
        ...
```

Set up complex robots by composing multiple components.

Get instance references through dependency injection.

```
<for each='range(rows)' as='z'>
    <for each='range(cols)' as='x'>
        <body pos="
            ${0.5 + 0.082*x if z % 2 == 0 else 0.58}
            ${0.48 if z % 2 == 0 else 0.4 + 0.082*x}
            ${0.061*z}"
            euler="0 0 ${0 if z % 2 == 0 else 1.57}">
            <freejoint/>
            <geom class='tt-block'/>
        </body>
    </for>
</for>
```

Expression evaluation and aditional tags for dynamic templating.

Example

component

interface for

MuJoCo

```python
from yarok import interface

@interface(
    defaults={
        'conf1': 'some_value'
    }
)
class MyGripperInterfaceMJC:
    def __init__(self, mjc: InterfaceMJC, config: ConfigBlock):
        self.conf1 = config['conf1']
        self.mjc = mjc # wraps MuJoCo Python API

    def move(q):
        self.q = q

    def step(): # called every update step
         # handles referencing/scope for multiple component instances
         current_q = self.interface.sensordata()
         self.mjc.set_ctrl('joint1', self.q)

         # for cameras, converts depth to meters
         rgb, depth = self.mjc.read_camera('camera1')

```

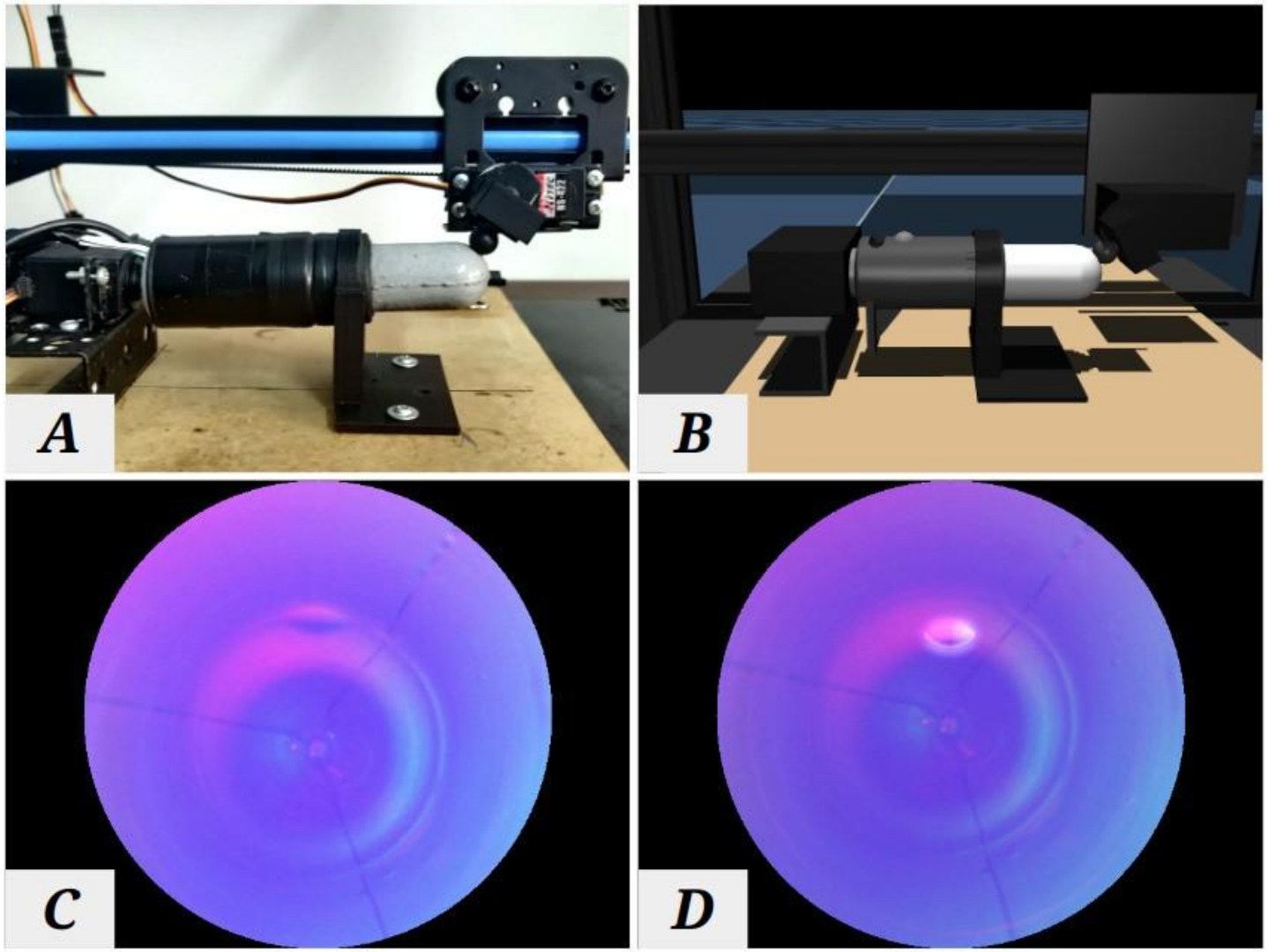
```
from yarok import Platform, PlatformMJC

Platform.create({
    'world': MyExperimentWorldComponent,
    'behaviour': SomeBehaviour,
    'defaults': {
        'environment': 'sim' # or 'real'
    },
    'environments': {
       'sim': {
            'platform': PlatformMJC,
            'components': {
                '/gripper1': {
                    'conf1':'value1'
                     # ... etc
                }
            }
        }
    }
}).run()
```

Yarok loads the appropriate component interfaces at runtime (following the environment platform)

Shared sim/real behaviour

Interfaces for implementing sensor simulation model, and remaining sim and real hardware.

Simulation configs reparemeterization for collecting large dataset

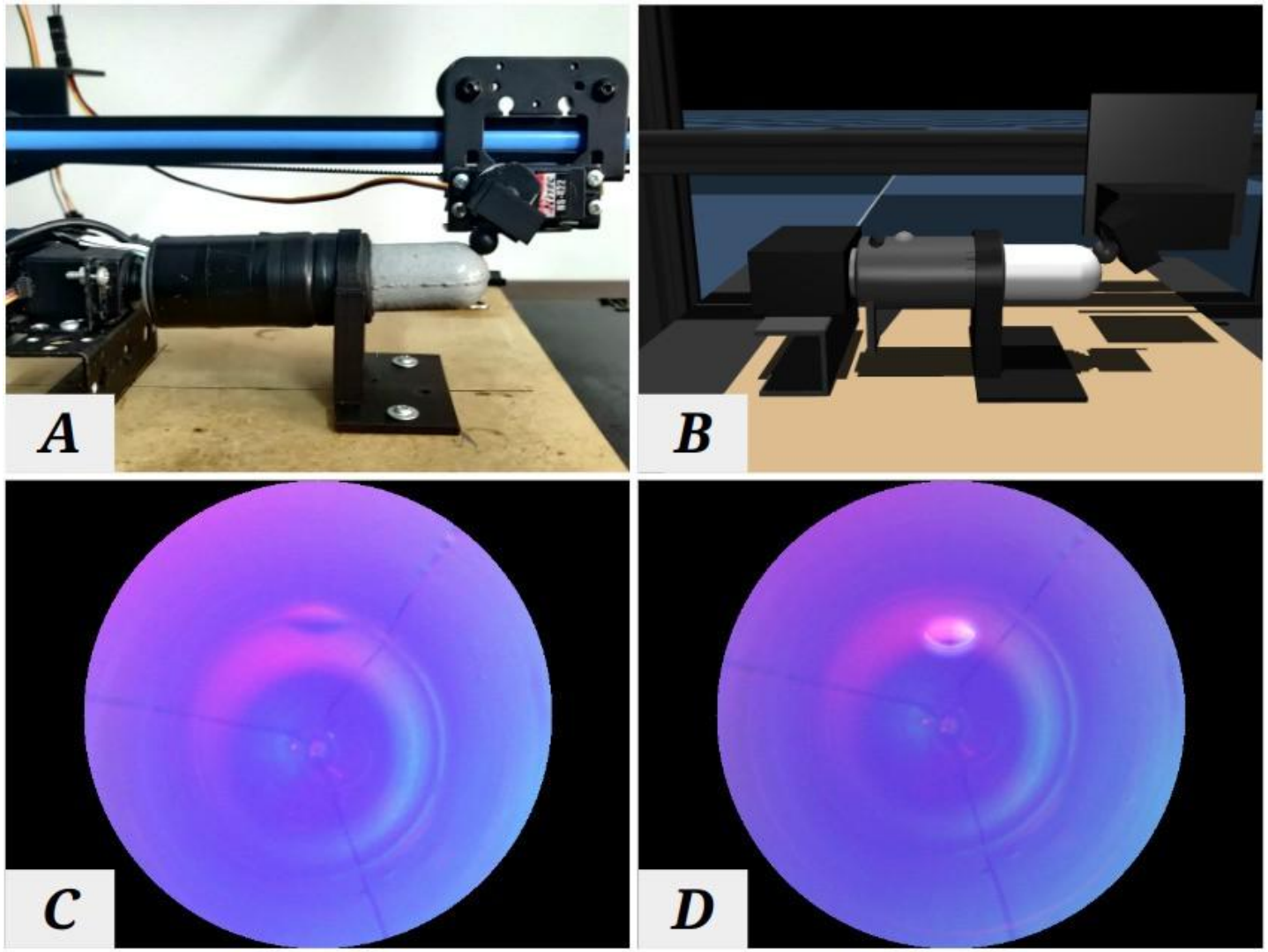


***Use case**. "Beyond Flat Gelsight Sensors: Simulation of Optical Tactile Sensors of Complex Morphologies for Sim2Real" RSS 2023*

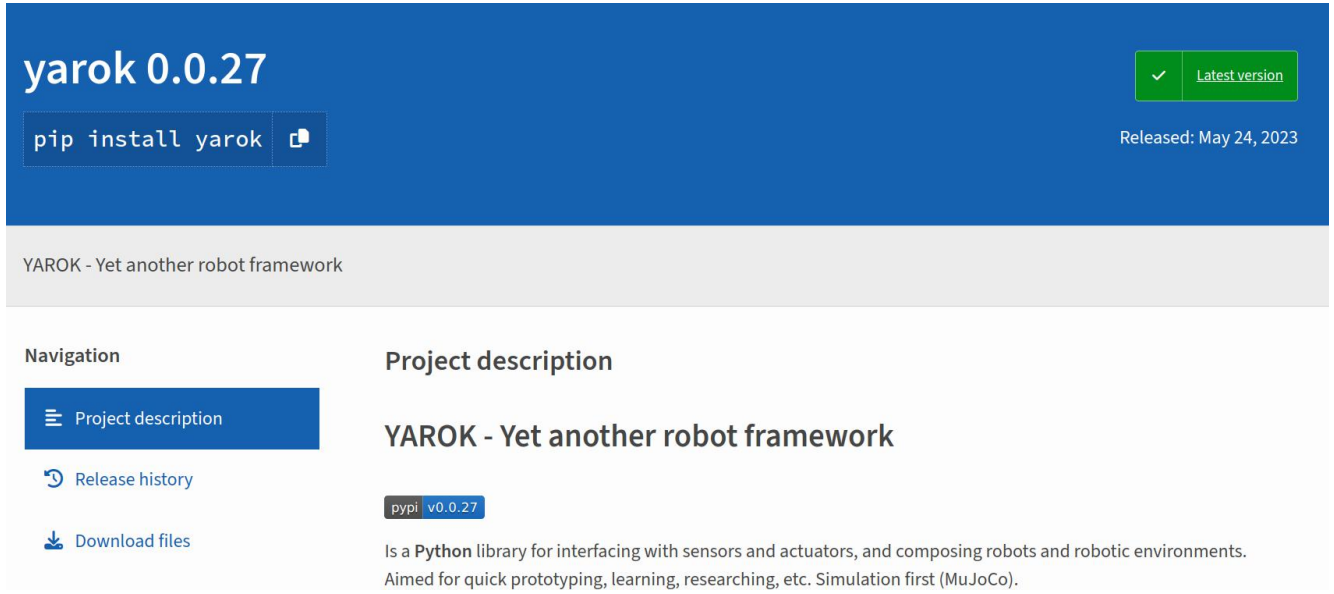
Shared sim/real behaviour

Interfaces for implementing sensor simulation model, and remaining sim and real hardware.

Simulation configs reparemeterization for collecting large dataset



***Use case**. "Beyond Flat Gelsight Sensors: Simulation of Optical Tactile Sensors of Complex Morphologies for Sim2Real" RSS 2023*



*If you would like to use or contribute to Yarok, find us on **GitHub** or **Pipy***