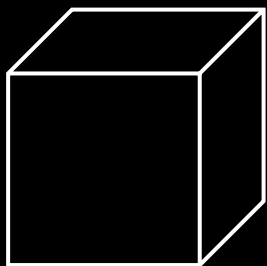
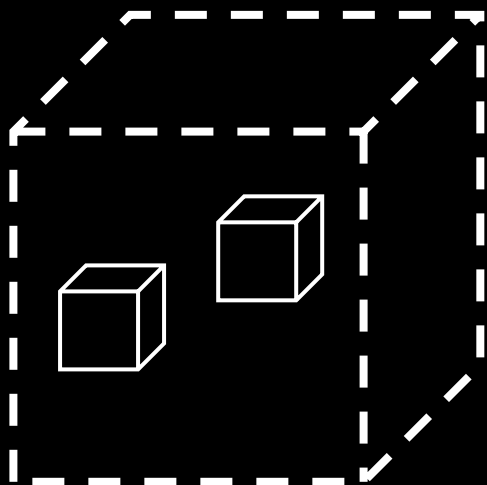


YAROK

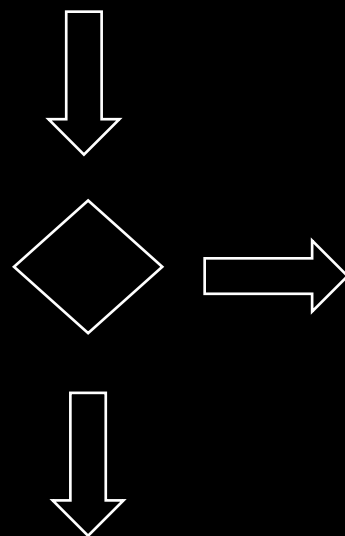
Yet Another Robot Framework



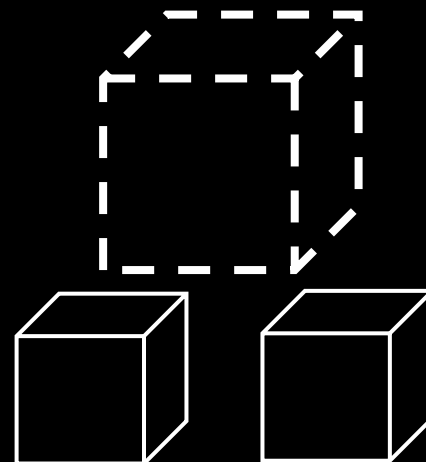
*Python Components
(template +
controller)*



*Flexible composing
mechanism*



*Extended MJCF:
flexible composing,
control flow macros*



*Code once, run
everywhere (sim
and real hardware)*

```
1 @component(  
2     tag="my-robot-arm",  
3     defaults={  
4         'interface_mjc': MyRobotArmMJC,  
5         'interface_hw': MyRobotArmHW,  
6     }  
7     template= ...  
8 class MyComponent:  
9     ...
```

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

```
1  from my_shared_components import MyGripper, MyTouchSensor
2
3  components=[
4      MyGripper, MyTouchSensor
5  ]
6  template="""
7      <my-gripper name="gripper1">
8          <my-sensor name="sensor1" parent="finger1_tip"></my-sensor>
9      </my-gripper>
10     ...
11 """
12 class MyComponent:
13
14     def __init__(gripper1: MyGripper, sensor1: MySensor):
15         ...
```

Set up complex robots by composing multiple components.
Get instance references through dependency injection.

```
1 <for each='range(rows)' as='z'>
2   <for each='range(cols)' as='x'>
3     <body pos="
4       ${0.5 + 0.082*x if z % 2 == 0 else 0.58}
5       ${0.48 if z % 2 == 0 else 0.4 + 0.082*x}
6       ${0.061*z}"
7       euler="0 0 ${0 if z % 2 == 0 else 1.57}">
8       <freejoint/>
9       <geom class='tt-block'/>
10    </body>
11  </for>
12 </for>
```

Expression evaluation and additional tags for dynamic templating.



Example
component
interface for
MuJoCo

```
1  from yarok import interface
2
3  @interface(
4      defaults={
5          'conf1': 'some_value'
6      }
7  )
8  class MyGripperInterfaceMJC:
9      def __init__(self, mjc: InterfaceMJC, config: ConfigBlock):
10         self.conf1 = config['conf1']
11         self.mjc = mjc # wraps MuJoCo Python API
12
13         def move(q):
14             self.q = q
15
16         def step(): # called every update step
17             # handles referencing/scope for multiple component instances
18             current_q = self.interface.sensordata()
19             self.mjc.set_ctrl('joint1', self.q)
20
21             # for cameras, converts depth to meters
22             rgb, depth = self.mjc.read_camera('camera1')
23
```



```

1  from yarok import Platform, PlatformMJC
2
3  Platform.create({
4      'world': MyExperimentWorldComponent,
5      'behaviour': SomeBehaviour,
6      'defaults': {
7          'environment': 'sim' # or 'real'
8      },
9      'environments': {
10         'sim': {
11             'platform': PlatformMJC,
12             'components': {
13                 '/gripper1': {
14                     'confl': 'value1'
15                     # ... etc
16                 }
17             }
18         }
19     }
20 }).run()

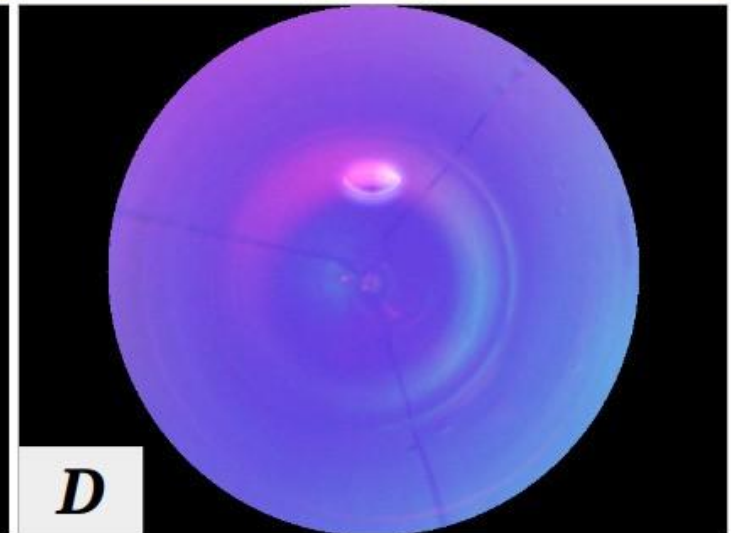
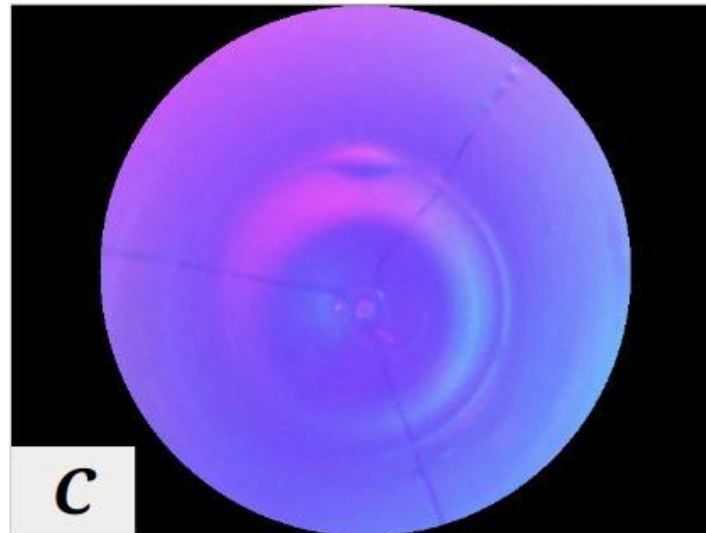
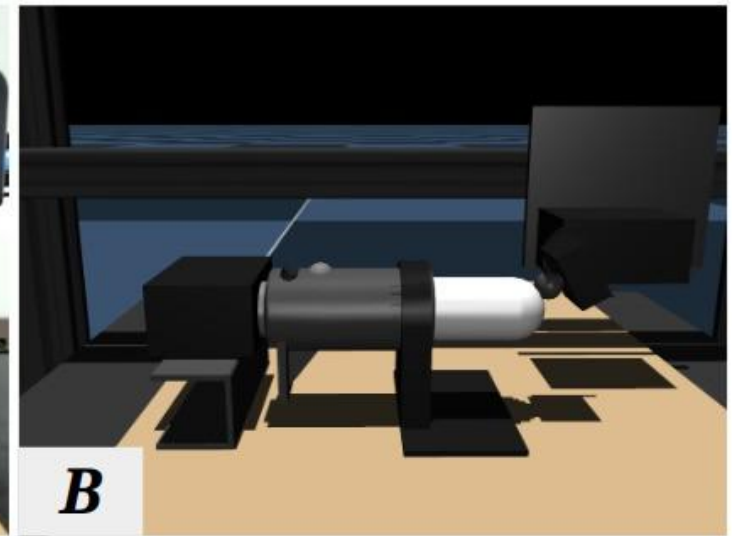
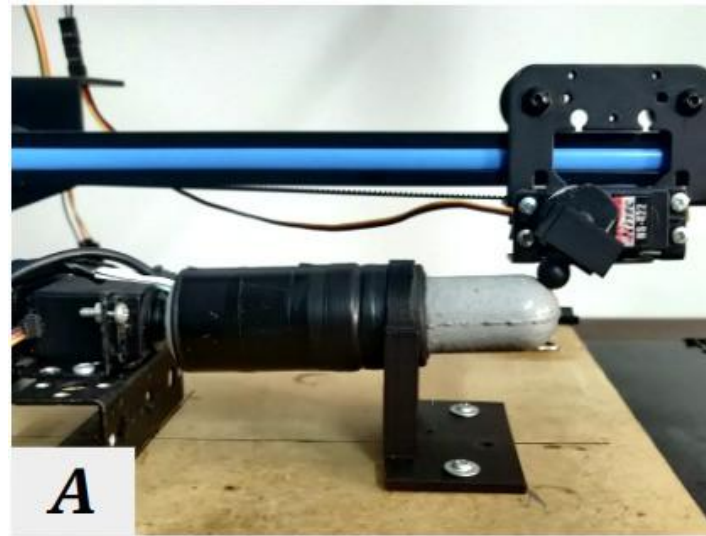
```

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
reparameterization for
collecting large dataset

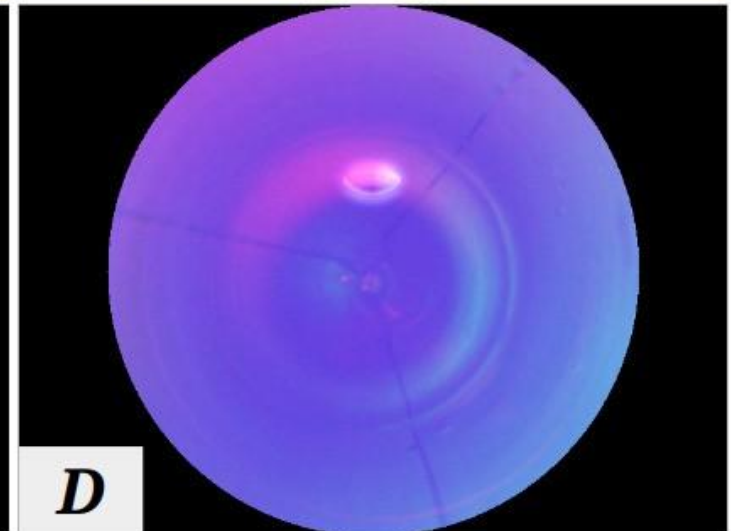
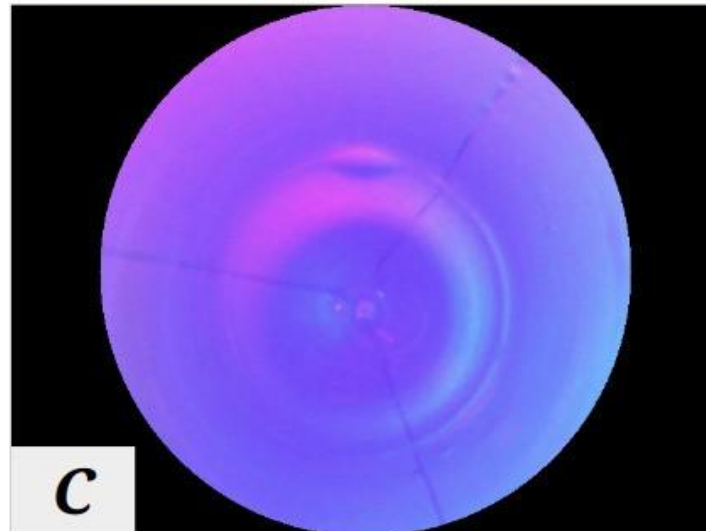
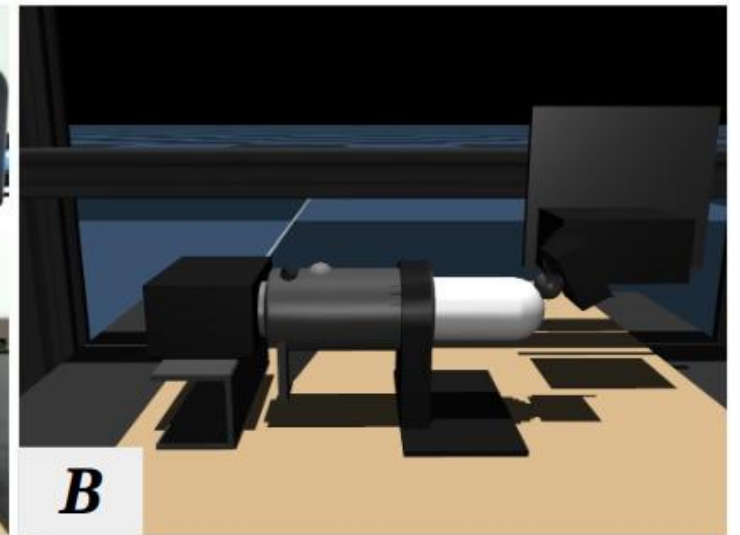
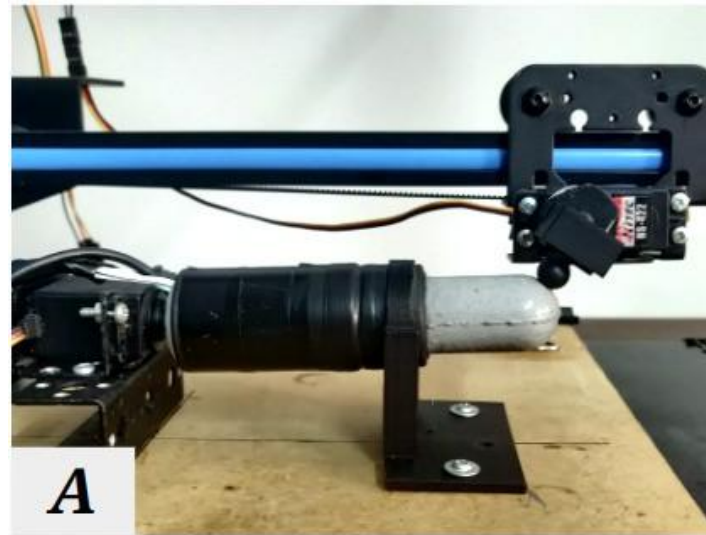


Use case. *“Beyond Flat Gelsight Sensors: Simulation of Optical Tactile Sensors of Complex Morphologies for Sim2Real” RSS 2023*

Shared sim/real
behaviour

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

Simulation configs
reparameterization for
collecting large dataset



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

yarok 0.0.27

`pip install yarok`

✓ Latest version

Released: May 24, 2023

YAROK - Yet another robot framework

Navigation

Project description

Release history

Download files

Project description

YAROK - Yet another robot framework

pypi

v0.0.27

Is a **Python** library for interfacing with sensors and actuators, and composing robots and robotic environments. Aimed for quick prototyping, learning, researching, etc. Simulation first (MuJoCo).



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