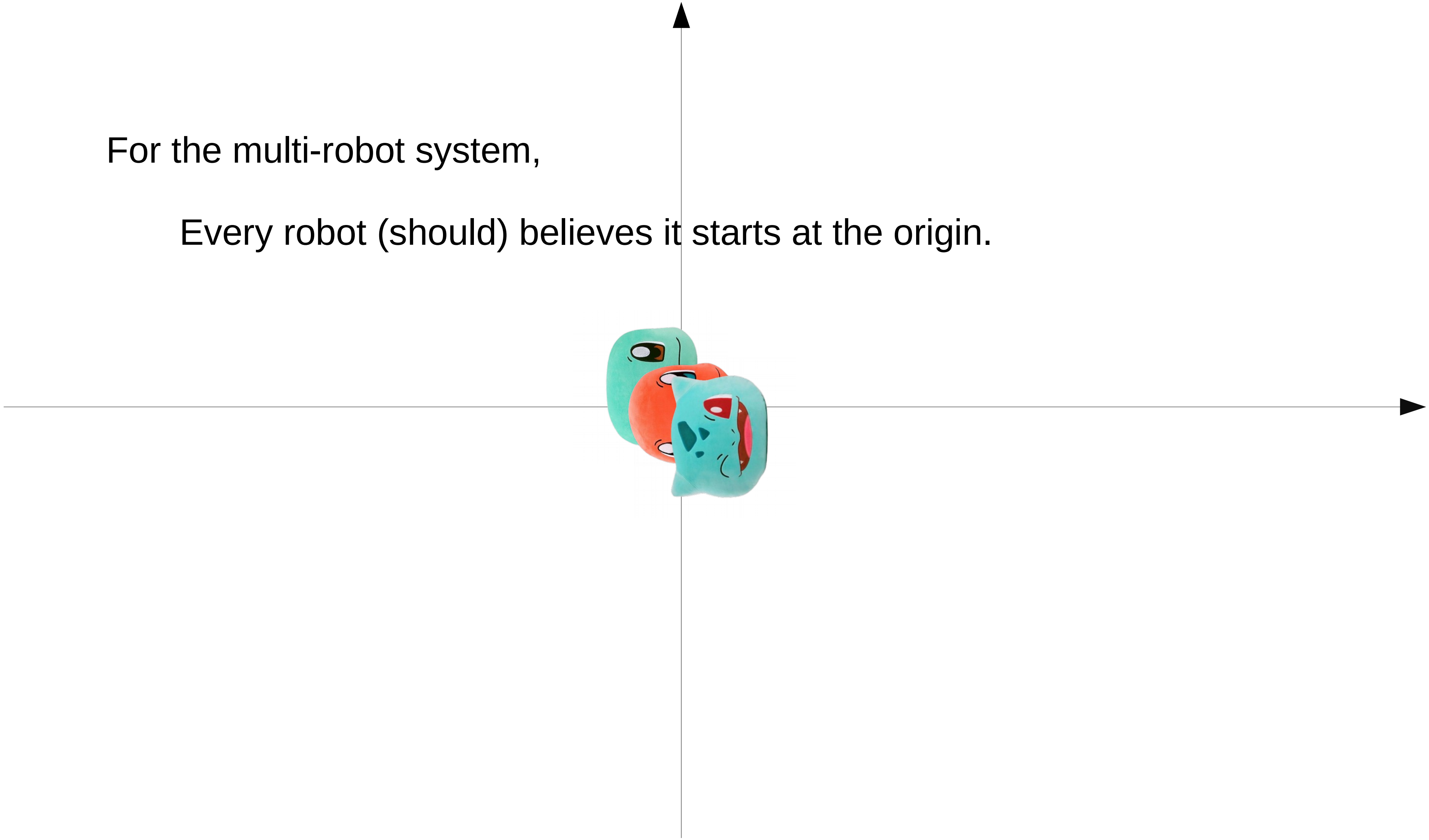
An explanation sheet of iSAM multi-robot SLAM tutorial

- 2019.11.28

- Giseop Kim, IRAP lab, KAIST
- paulgkim@kaist.ac.kr

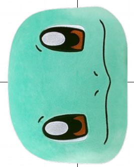
For the multi-robot system,

Every robot (should) believes it starts at the origin.



For the multi-robot system,

Every robot (should) believes it starts at the origin.



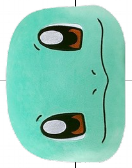
Because actually they do not know other’s existence before encounters

This example shows only the robot 1 exists at time 0.

```
// First, robot 1 is generated and starts to operate
bool is_base_session = true;
Robot2D robot_1(multislam, is_base_session); // multi sesssion ver
multi_robots.push_back(&robot_1);
is_base_session = false; // off the flag (because anchor graph also has the only prior)
```

For the multi-robot system,

Every robot (should) believes it starts at the origin.



Because actually they do not know other’s existence before encounters

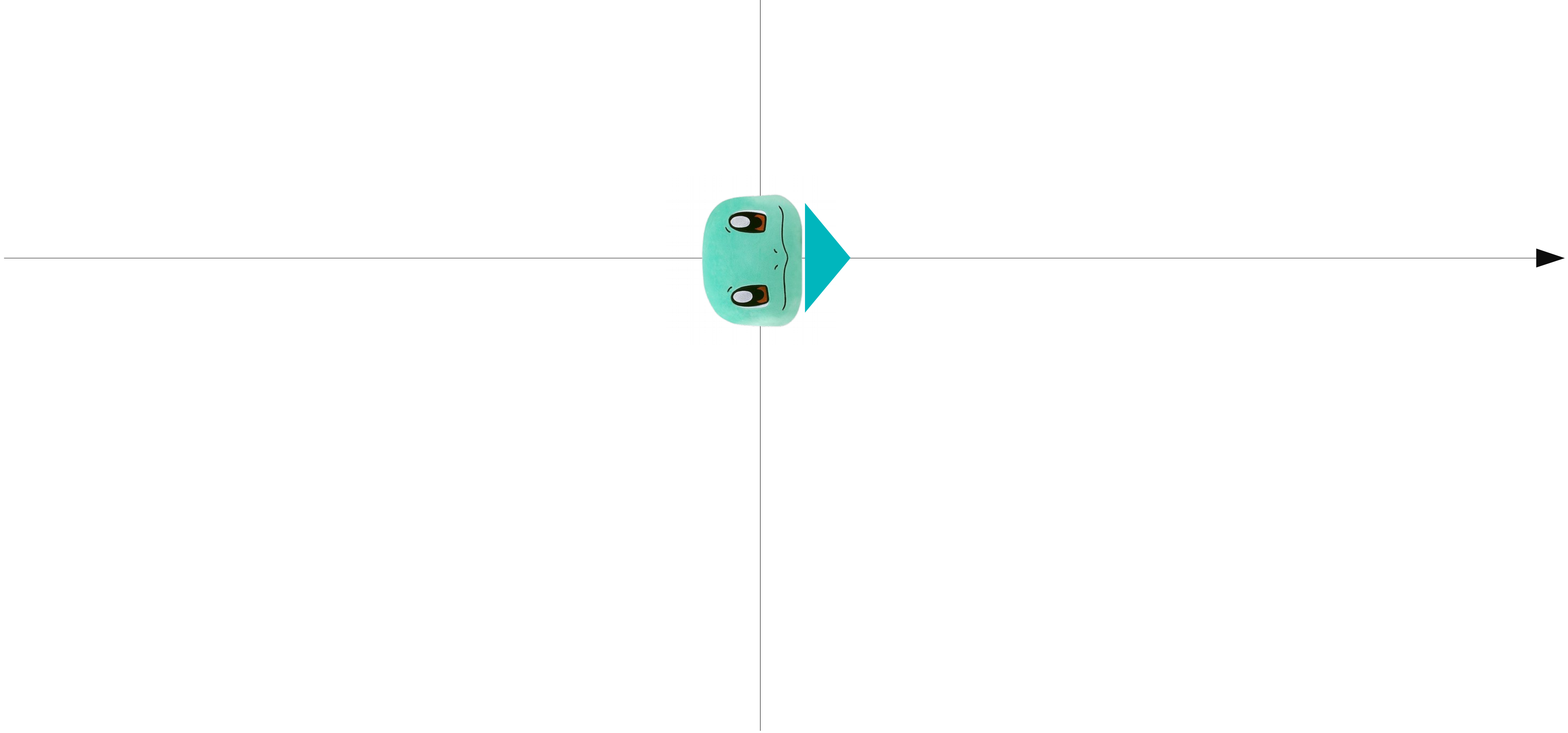
This example shows only the robot 1 exists at time 0.

```
// First, robot 1 is generated and starts to operate
bool is_base_session = true;
Robot2D robot_1(multislam, is_base_session); // multi sesssion ver
multi_robots.push_back(&robot_1);
is_base_session = false; // off the flag (because anchor graph also has the only prior)
```

In this multi-robot example, we assume the first appeared robot’s frame is the global.

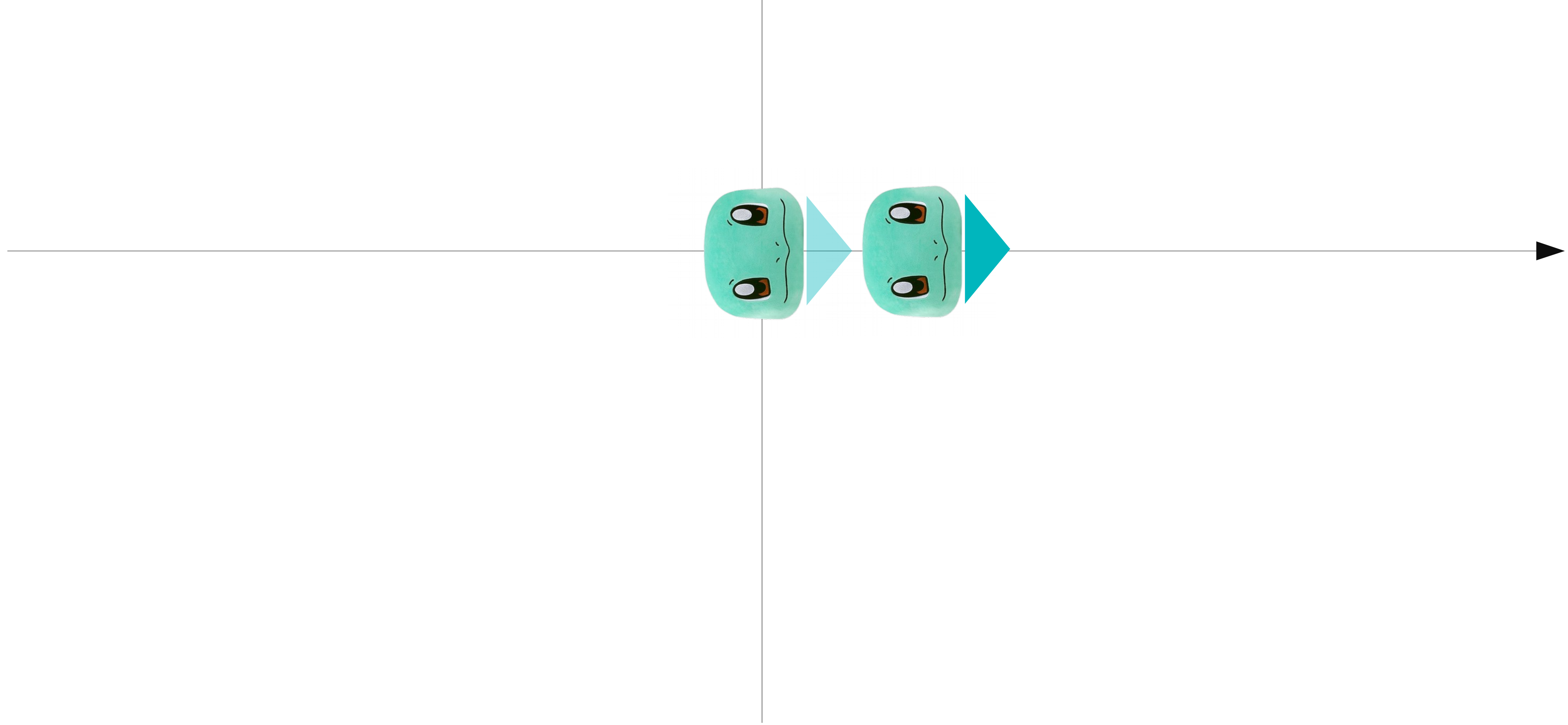
```cpp
// First, robot 1 is generated and starts to operate
bool is_base_session = true;
Robot2D robot_1(multislam, is_base_session); // multi sesssion ver
multi_robots.push_back(&robot_1);
is_base_session = false; // off the flag (because anchor graph also has the only prior)
```

Ok. now the robot 1 is generated and it starts at the origin. (others have not appeared yet.)



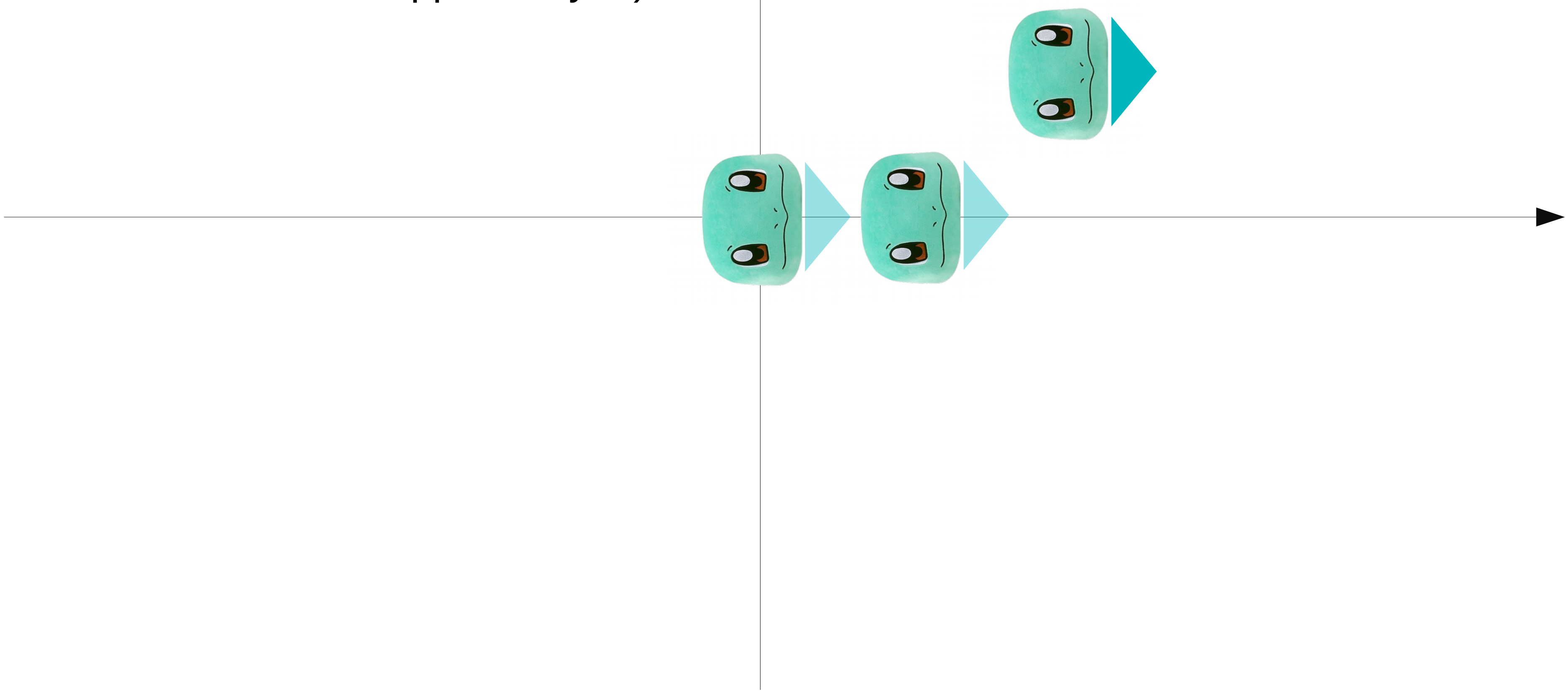
```
// The robot 1 moves a single step in x
robot_1.addOdometryFactor(Pose2d(one_step, 0.0, 0.0));
```

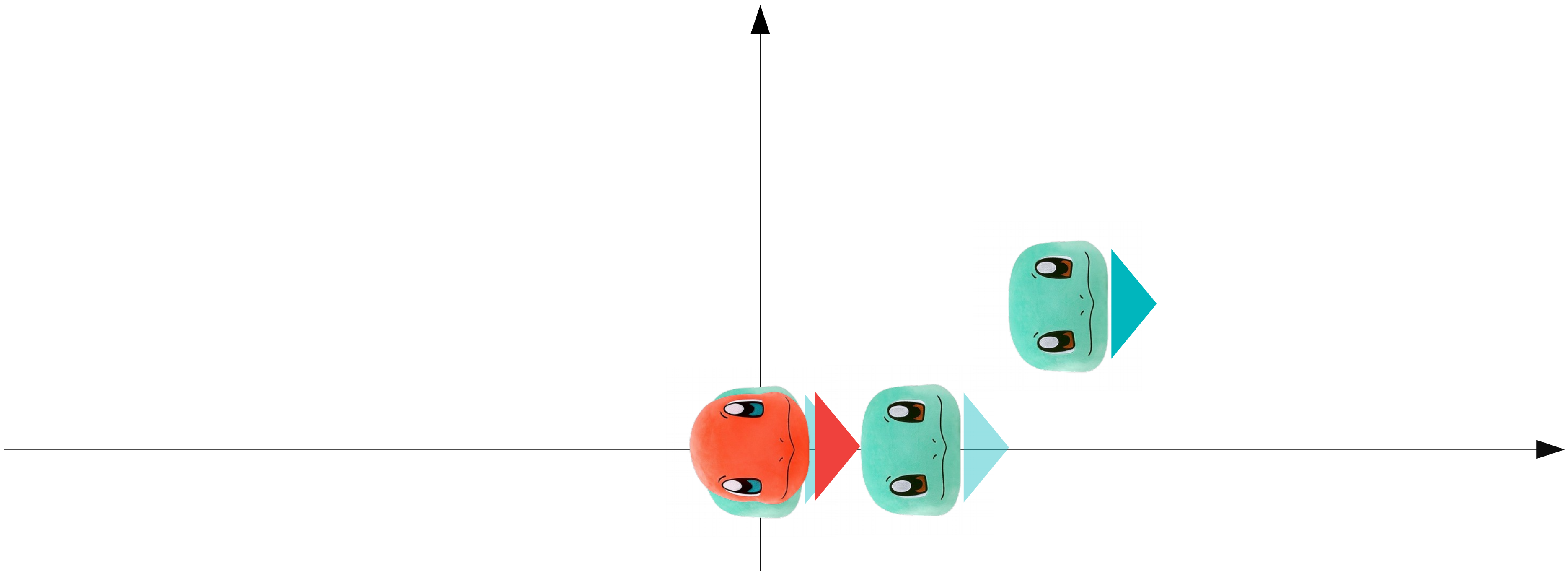
The robot 1 moved 1 step in x. (others still have not appeared yet.)



```
// The robot 1 moves again 2 steps more in x and 1 step in y
robot_1.addOdometryFactor(Pose2d(2.0*one_step, one_step, 0.0));
```

The robot 1 moved again 2 step in x and 1 step in y.
(others still … have not appeared yet.)





```
// While the robot 1 is moving,
// A robot 2, the other robot, is generated and starts to operate
Robot2D robot_2(multislam);
multi_robots.push_back(&robot_2);
```

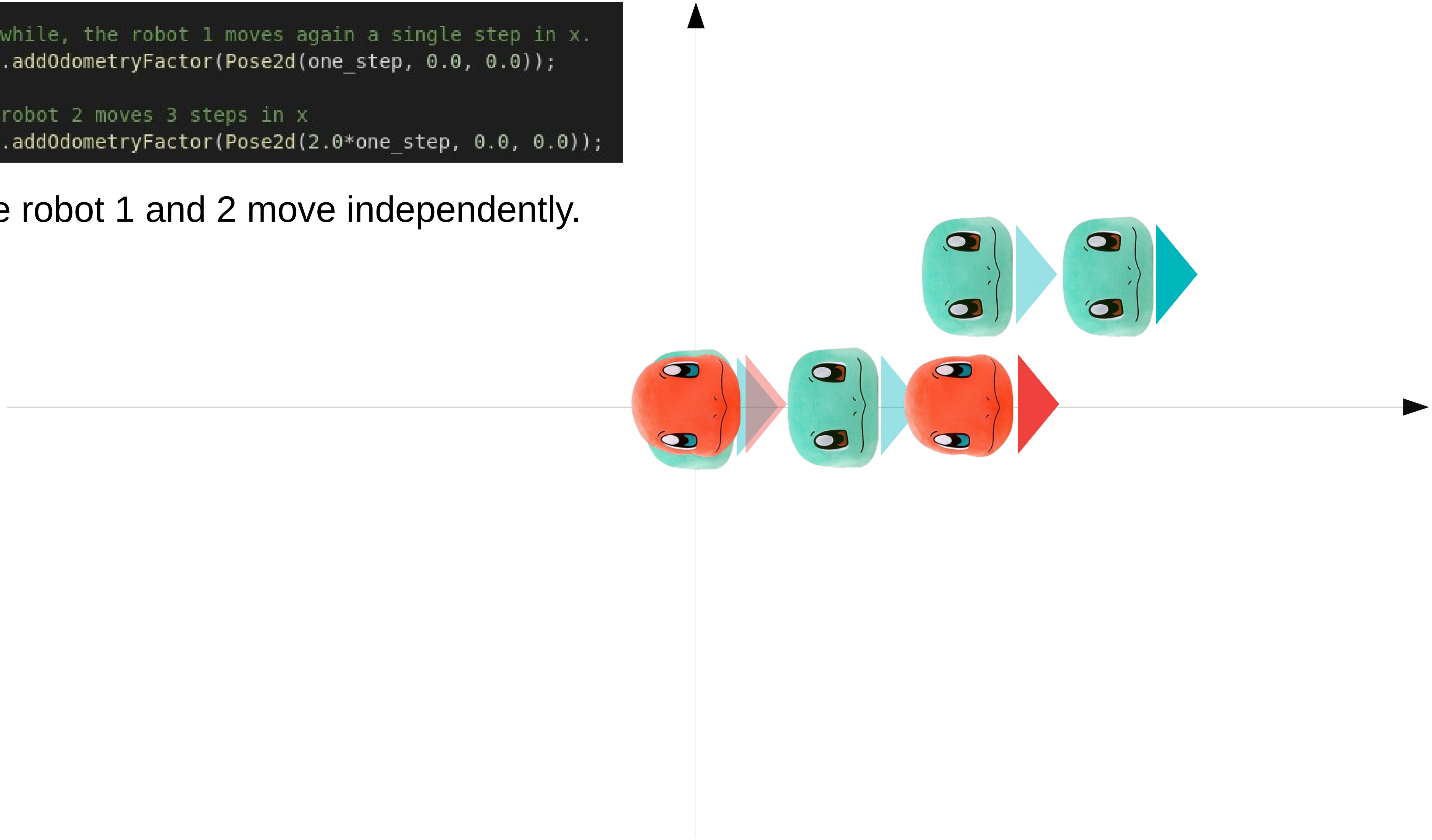
Suddenly the robot 2 appeared.
But robot 2 does not know the robot 1 yet. (i.e., the robot 1 not seen by the robot 2)

Therefore the robot 2 should starts its own origin. (not know others frame yet)

```
// Meanwhile, the robot 1 moves again a single step in x.
robot_1.addOdometryFactor(Pose2d(one_step, 0.0, 0.0));

// The robot 2 moves 3 steps in x
robot_2.addOdometryFactor(Pose2d(2.0*one_step, 0.0, 0.0));
```

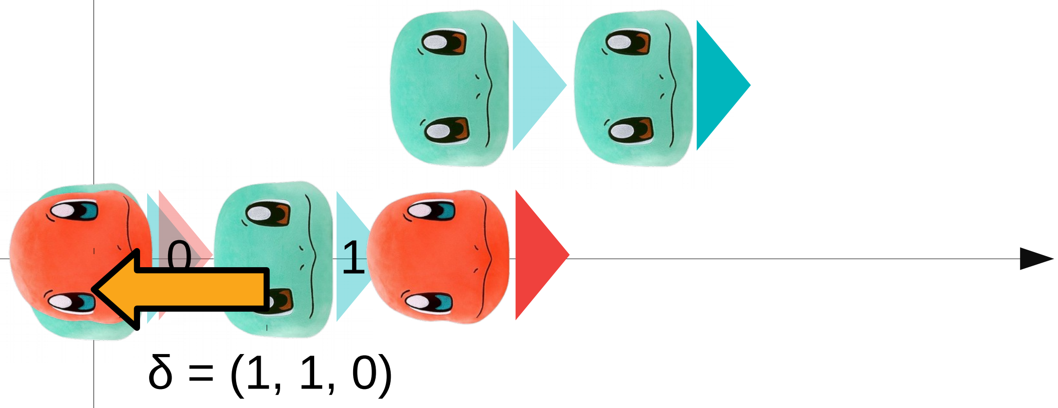
The robot 1 and 2 move independently.



At that time, the robot 1 noticed (encountered) the robot 2!
And it realized also the relative between them is (1, 1, 0).

i.e., a node 0 of the robot 2
= a node 1 of the robot 1 ⊕ (1,1,0)

Inter-robot loop is added.




```
// The robot 1 and 2 encountered (i.e., inter-loop detected)
// - Assume the node 1 of the robot 1 and the node 0 of the robot 2 has a relative transformation (1,1,0)
int loop_node_target;
int loop_node_matched;
Pose2d loop_relative;

loop_node_target = 1;
loop_node_matched = 0;
loop_relative = Pose2d(1., 1., 0.);
robot_1.addInterLoopFactor(loop_node_target, robot_2, loop_node_matched, loop_relative);
```

If we close the loop

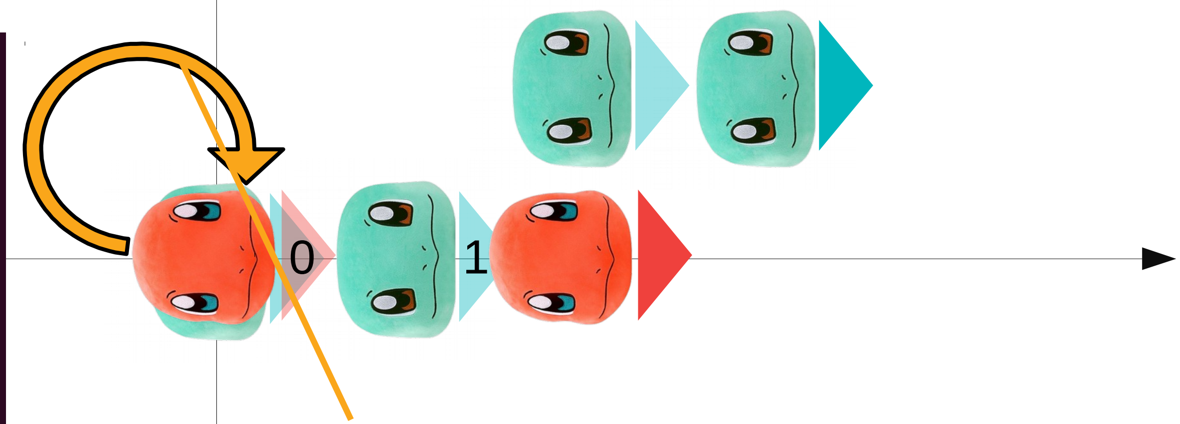
```
// optimization including the anchor node graph
cout << endl << "optimize the multi-session graph ..." << endl;
multislam->batch_optimization(); // == robot_1.batchOptimizationMultiSlam()
```

The result is:
(i.e., anchor node pose is optimized)
== node 0 of the robot 2 to node o of the robot 1

```
optimize the multi-session graph ...

----- The optimized graph -----
Robot 1 graph:
  anchor transform wrt the global: (0, 0, 0)
  x(0): robot frame (0, 0, 0) / global frame (0, 0, 0)
  x(1): robot frame (1, 0, 0) / global frame (1, 0, 0)
  x(2): robot frame (3, 1, 0) / global frame (3, 1, 0)
  x(3): robot frame (4, 1, 0) / global frame (4, 1, 0)
 graph saved as: ./robot_1_opt_1.graph

Robot 2 graph:
  anchor transform wrt the global: (2, 1, 0)
  x(0): robot frame (0, 0, 0) / global frame (2, 1, 0)
  x(1): robot frame (2, 0, 0) / global frame (4, 1, 0)
 graph saved as: ./robot_2_opt_1.graph
```



Anchor Δ = (2, 1, 0)
for all nodes in the robot 2

If we close the loop

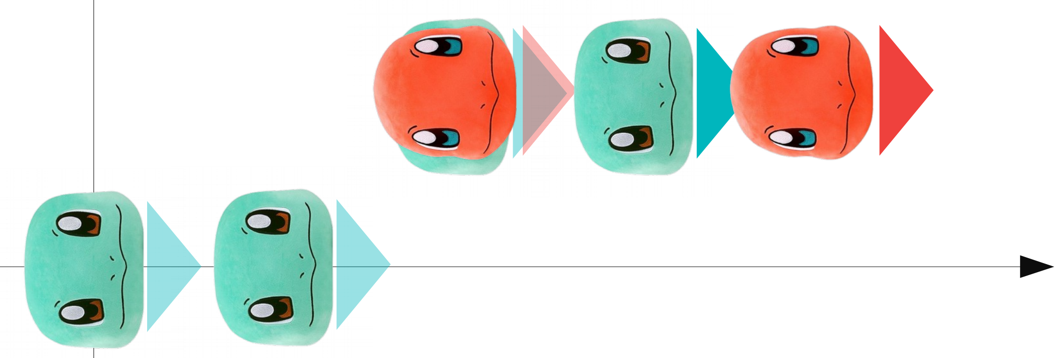
```
// optimization including the anchor node graph
cout << endl << "optimize the multi-session graph ..." << endl;
multislam->batch_optimization(); // == robot_1.batchOptimizationMultiSlam()
```

The result is:
(i.e., anchor node pose is optimized)

```
optimize the multi-session graph ...

----- The optimized graph -----
Robot 1 graph:
  anchor transform wrt the global: (0, 0, 0)
  x(0): robot frame (0, 0, 0) / global frame (0, 0, 0)
  x(1): robot frame (1, 0, 0) / global frame (1, 0, 0)
  x(2): robot frame (3, 1, 0) / global frame (3, 1, 0)
  x(3): robot frame (4, 1, 0) / global frame (4, 1, 0)
 graph saved as: ./robot_1_opt_1.graph

Robot 2 graph:
  anchor transform wrt the global: (2, 1, 0)
  x(0): robot frame (0, 0, 0) / global frame (2, 1, 0)
  x(1): robot frame (2, 0, 0) / global frame (4, 1, 0)
 graph saved as: ./robot_2_opt_1.graph
```

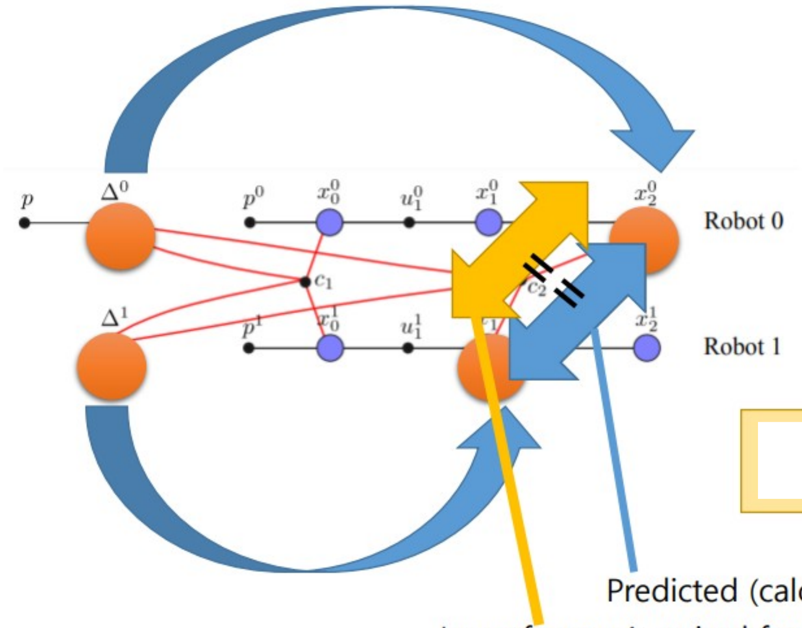


The robot 2's poses
are transformed into the global frame (i.e., robot 1)
using the relative between anchor node 1 to 2.

BTW, the detailed principle of anchor node optimization for multi-robot SLAM is here.

Refer the original paper for details,
10 ICRA, Been Kim, et al, multiple relative pose graphs for robust cooperative mapping

$$X^* = \arg\min_X \left\{ \sum_{r=0}^{R-1} \left(\|\mathbf{p}^r - \mathbf{x}_0^r\|_\Sigma^2 + \sum_{i=1}^{M_r} \|f_i(\mathbf{x}_{i-1}^r, \mathbf{u}_i^r) - \mathbf{x}_i^r\|_{\Lambda_i^r}^2 \right) + \underbrace{\sum_{j=1}^{N} \left\| h_j(\mathbf{x}_{i'_j}^{r'_j}, \mathbf{c}_j) - \mathbf{x}_{i_j}^{r_j} \right\|_{\Gamma_j}^2}_{\text{loop}} \right\}$$



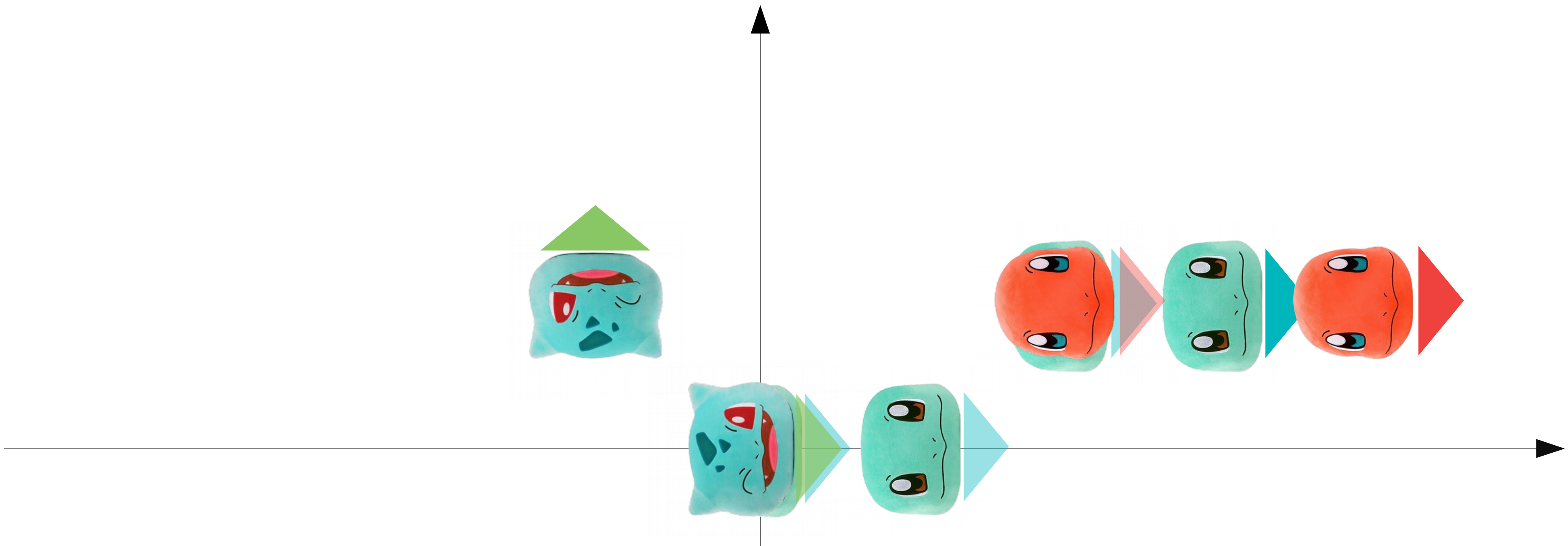
$$\sum_{j=1}^{N} \left\| h'_j(\mathbf{x}_{i'_j}^{r'_j}, \mathbf{c}_j, \Delta^{r_j}, \Delta^{r'_j}) - \mathbf{x}_{i_j}^{r_j} \right\|_{\Gamma_j}^2$$

Loop factor anchor node version

Such that $\mathbf{c} = (\Delta^r \oplus \mathbf{x}_i^r) \ominus \left(\Delta^{r'} \oplus \mathbf{x}_{i'}^{r'}\right)$

Predicted (calculated) from current (noisy) graph
Loop factor: Acquired from marker, icp, or PnP ... etc.

Optimize Δ1, not $\mathbf{x}_{i'}^{r'}$
to make these to be equal

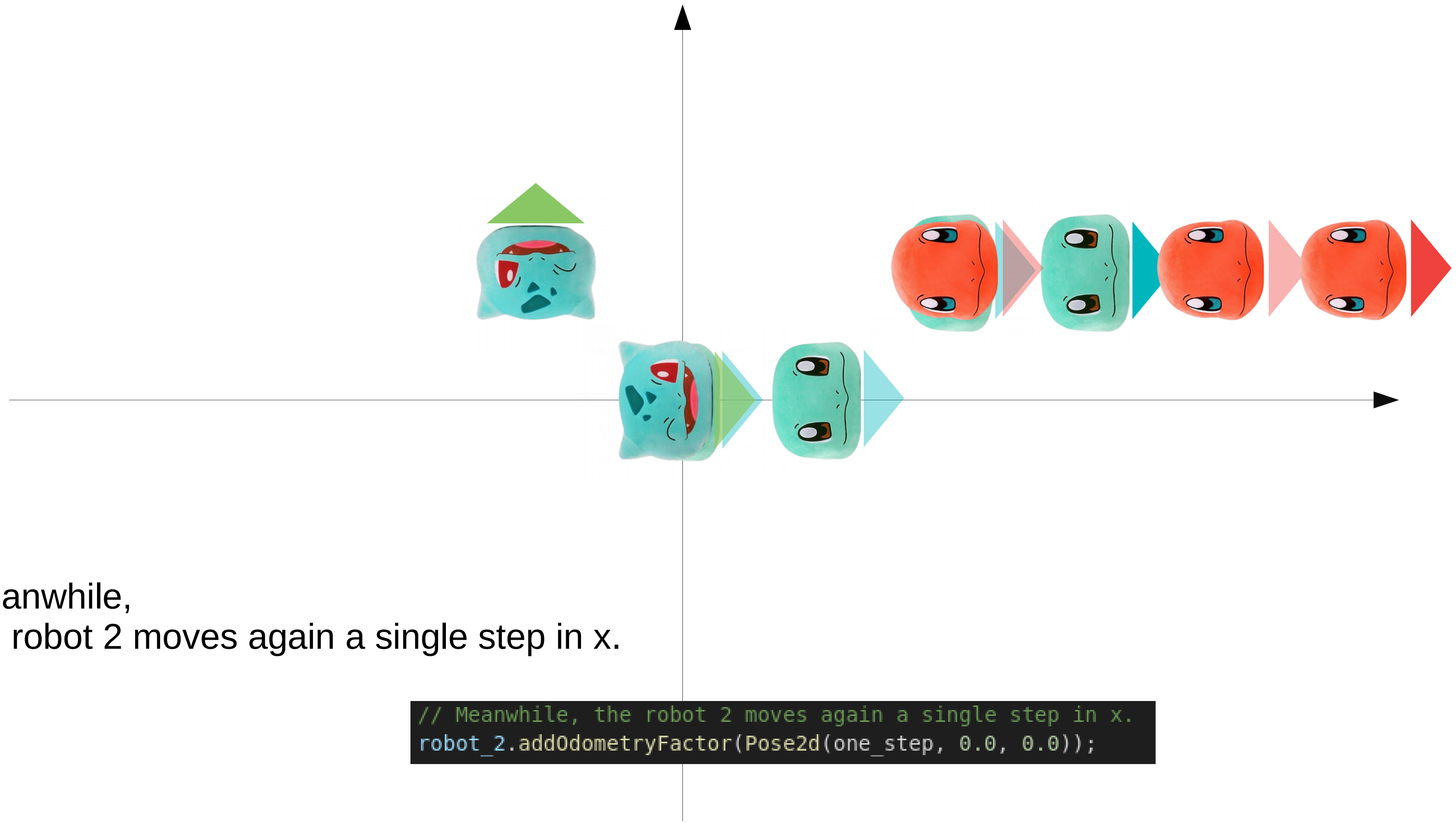


While the robot 1 and 2 being optimized,

Now a robot 3 is appeared.
And it moves
a negative step in x,
a step in y,
and turn left.

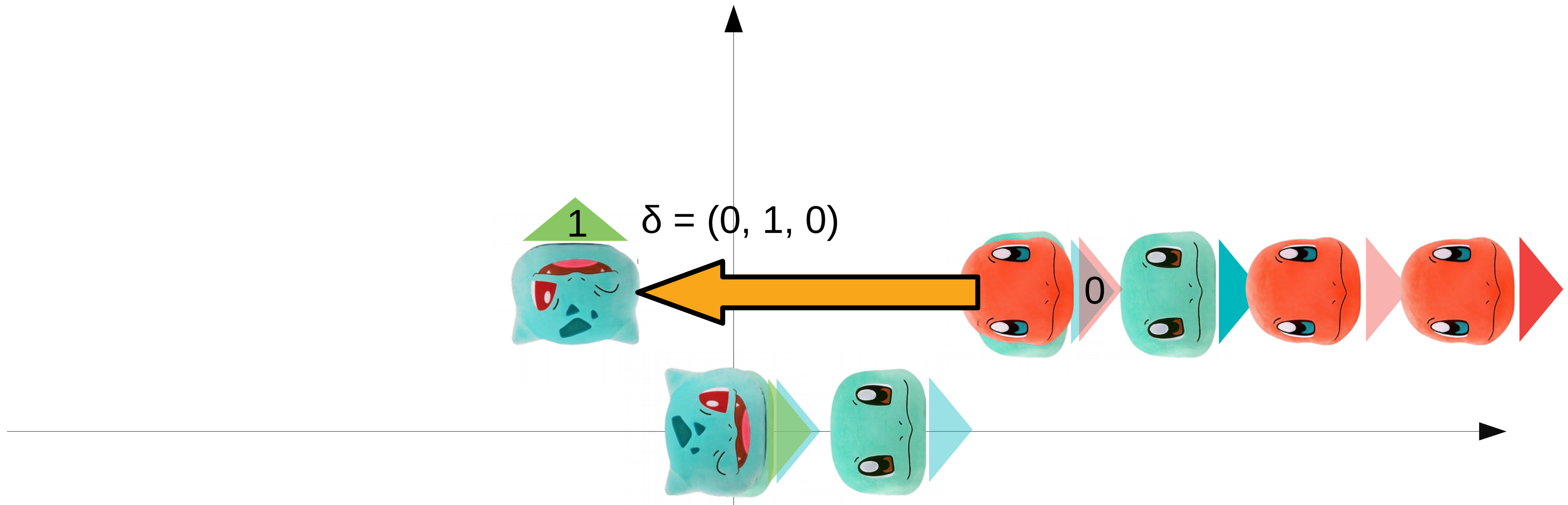
```
// While the robot 1 and 2 are going around,
// The other robot, a robot 3 is generated and starts to operate
Robot2D robot_3(multislam);
multi_robots.push_back(&robot_3);

// The robot 3 moves -1 steps in x, 1 step in y, and rotates 90 deg
robot_3.addOdometryFactor(Pose2d( -1.0*one_step, one_step, angle_turn_left));
```



Meanwhile,
the robot 2 moves again a single step in x.

```
// Meanwhile, the robot 2 moves again a single step in x.
robot_2.addOdometryFactor(Pose2d(one_step, 0.0, 0.0));
```

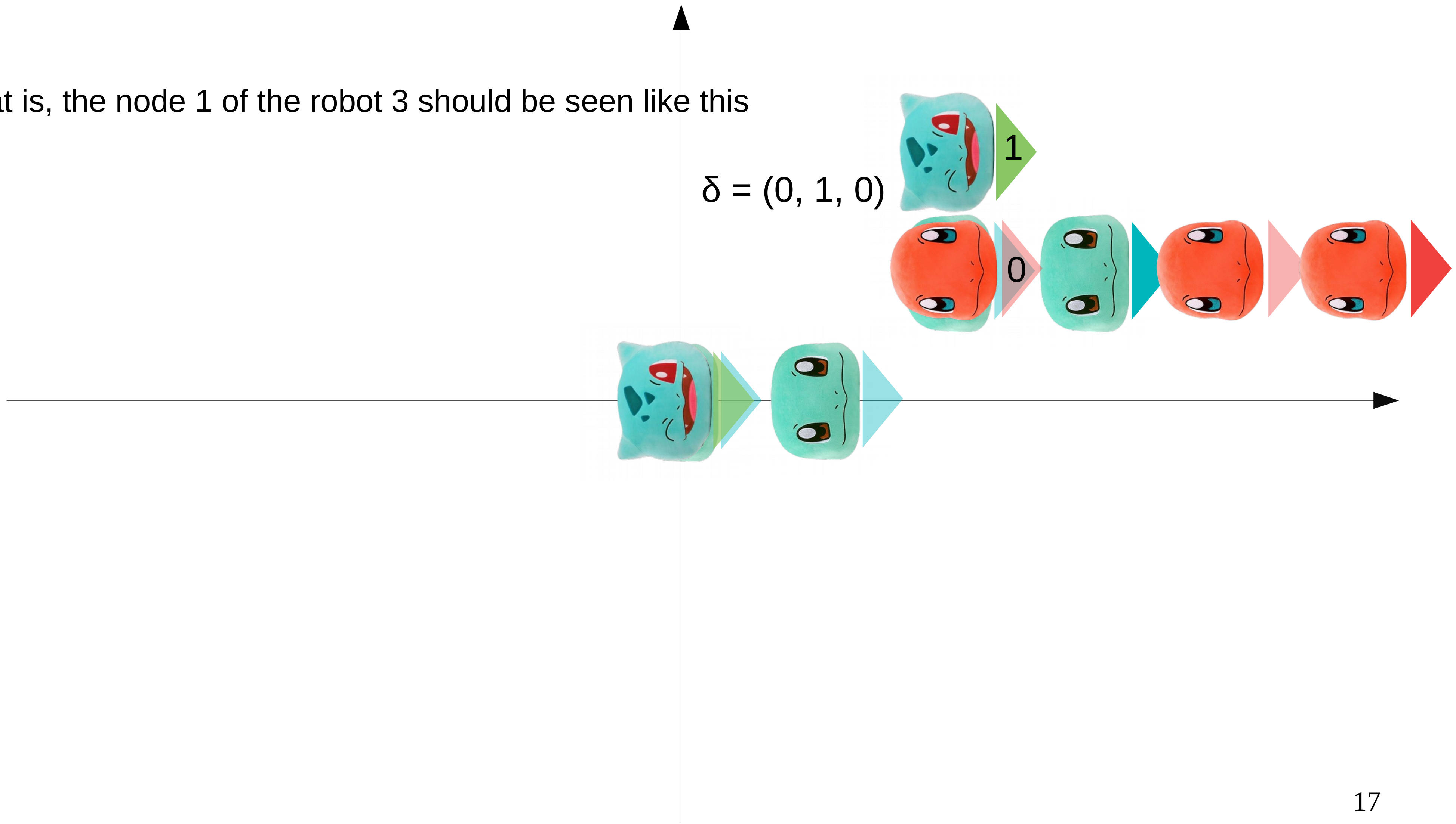


Then,
The robot 2 and 3 encountered (i.e., inter-loop detected)
Assume the node 0 of the robot 2 and the node 1 of the robot 3 has a relative transformation (0,1,0)

i.e., a node 1 of the robot 3
= a node 0 of the robot 2 ⊕ (0, 1, 0)

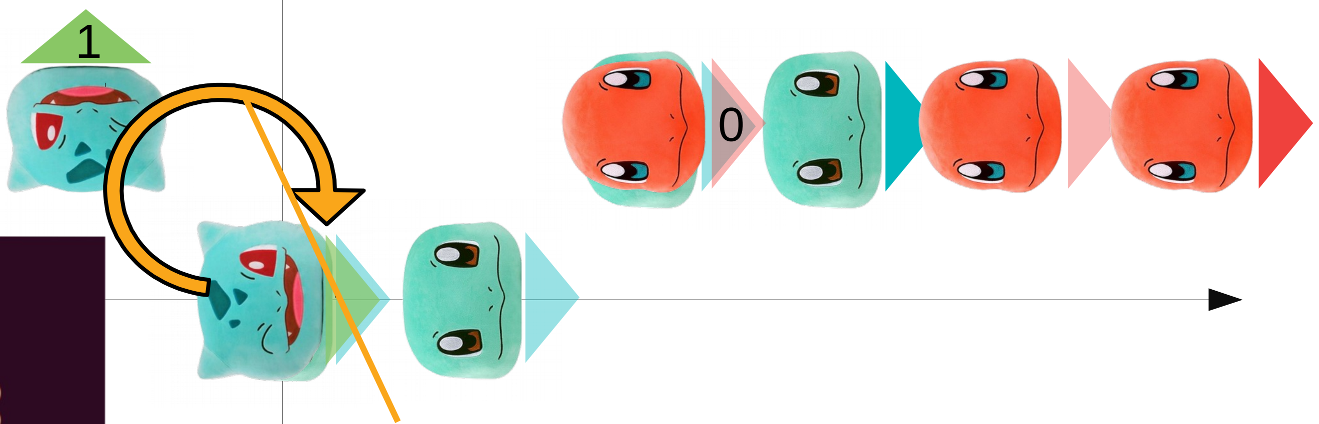
```
// The robot 2 and 3 encountered (i.e., inter-loop detected)
// - Assume the node 0 of the robot 2 and the node 1 of the robot 3 has a relative transformation (0,1,0)
loop_node_target = 0;
loop_node_matched = 1;
loop_relative = Pose2d(0., 1., 0.);
robot_2.addInterLoopFactor(loop_node_target, robot_3, loop_node_matched, loop_relative);
```

That is, the node 1 of the robot 3 should be seen like this



```cpp
// optimization including the anchor node graph
cout << endl << "optimize the multi-session graph ..." << endl;
multislam->batch_optimization(); // == robot_2.batchOptimizationMultiSlam()
```

Optimize the overall slam graph again ...



Δ = (1, 1, - 90deg)
for all nodes in the robot 3

```
optimize the multi-session graph ...

----- The optimized graph -----
Robot 1 graph:
  anchor transform wrt the global: (0, 0, 0)
  x(0): robot frame (0, 0, 0) / global frame (0, 0, 0)
  x(1): robot frame (1, 0, 0) / global frame (1, 0, 0)
  x(2): robot frame (3, 1, 0) / global frame (3, 1, 0)
  x(3): robot frame (4, 1, 0) / global frame (4, 1, 0)
 graph saved as: ./robot_1_opt_2.graph

Robot 2 graph:
  anchor transform wrt the global: (2, 1, 0)
  x(0): robot frame (0, 0, 0) / global frame (2, 1, 0)
  x(1): robot frame (2, 0, 0) / global frame (4, 1, 0)
  x(2): robot frame (3, 0, 0) / global frame (5, 1, 0)
 graph saved as: ./robot_2_opt_2.graph

Robot 3 graph:
  anchor transform wrt the global: (1, 1, -1.5708)
  x(0): robot frame (0, 0, 0) / global frame (1, 1, -1.5708)
  x(1): robot frame (-1, 1, 1.5708) / global frame (2, 2, 0)
 graph saved as: ./robot_3_opt_2.graph
```

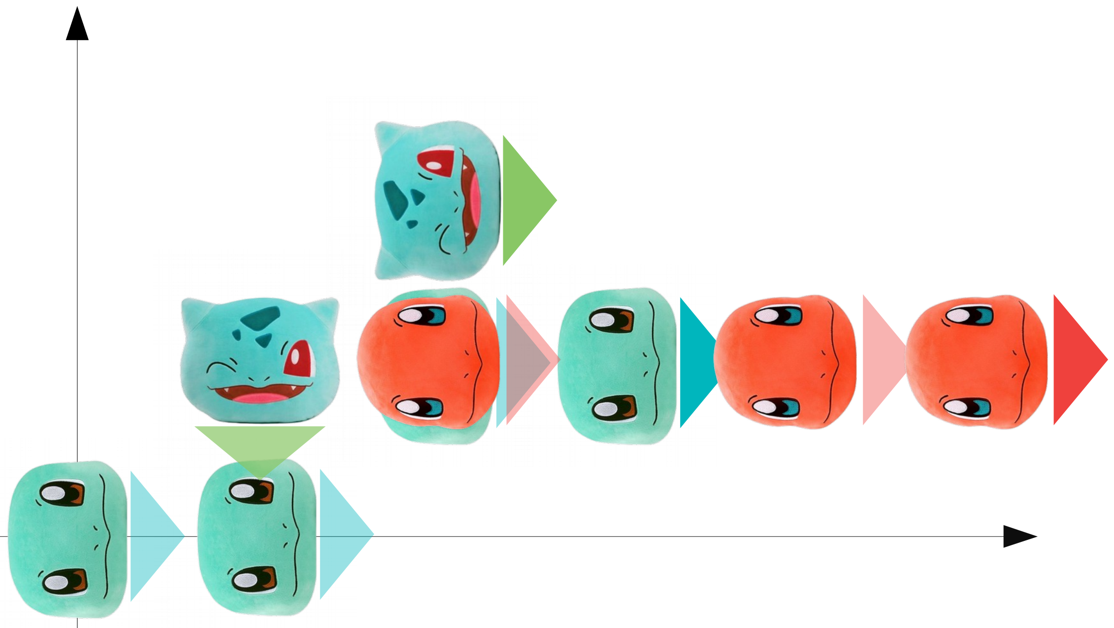
The result is:
(i.e., anchor node pose is optimized)
== node 0 of the robot 3 to node o of the robot 1

```
optimize the multi-session graph ...

----- The optimized graph -----
Robot 1 graph:
  anchor transform wrt the global: (0, 0, 0)
  x(0): robot frame (0, 0, 0) / global frame (0, 0, 0)
  x(1): robot frame (1, 0, 0) / global frame (1, 0, 0)
  x(2): robot frame (3, 1, 0) / global frame (3, 1, 0)
  x(3): robot frame (4, 1, 0) / global frame (4, 1, 0)
 graph saved as: ./robot_1_opt_2.graph

Robot 2 graph:
  anchor transform wrt the global: (2, 1, 0)
  x(0): robot frame (0, 0, 0) / global frame (2, 1, 0)
  x(1): robot frame (2, 0, 0) / global frame (4, 1, 0)
  x(2): robot frame (3, 0, 0) / global frame (5, 1, 0)
 graph saved as: ./robot_2_opt_2.graph

Robot 3 graph:
  anchor transform wrt the global: (1, 1, -1.5708)
  x(0): robot frame (0, 0, 0) / global frame (1, 1, -1.5708)
  x(1): robot frame (-1, 1, 1.5708) / global frame (2, 2, 0)
 graph saved as: ./robot_3_opt_2.graph
```



The robot 3's poses
are transformed into the global frame (i.e., robot 1)
using the relative between anchor node 1 to 3.

Using this eq.

global node pose = anchor pose ⊕ own (local) node pose

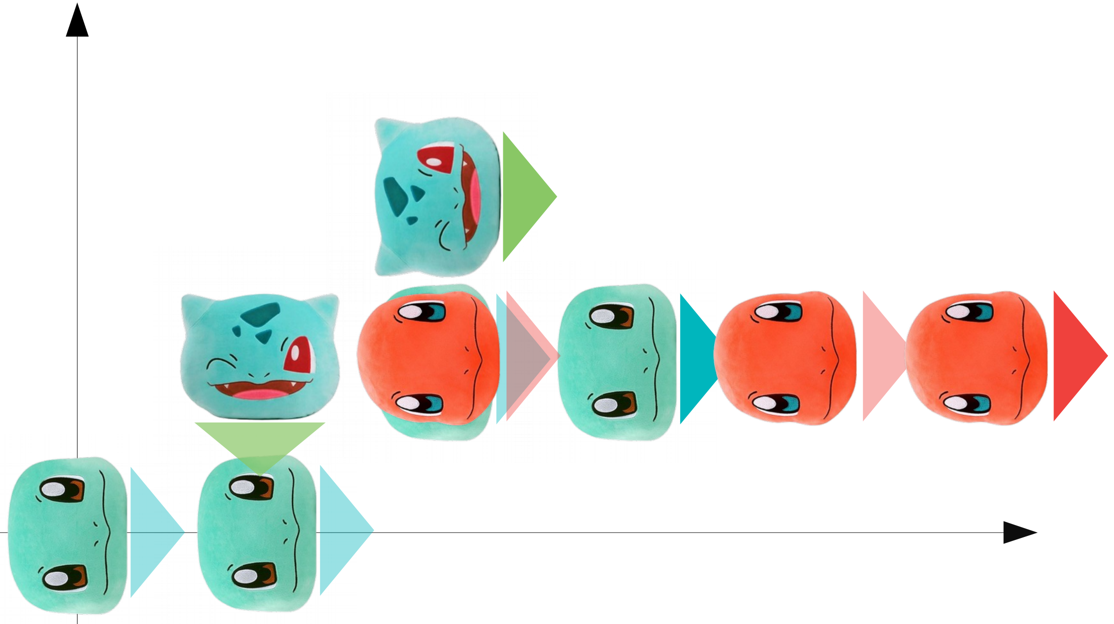
```
"/ global frame " << anchor_node_->value().oplus(nodes_.at(i).get()->value())
```

```
optimize the multi-session graph ...

----- The optimized graph -----
Robot 1 graph:
  anchor transform wrt the global: (0, 0, 0)
  x(0): robot frame (0, 0, 0) / global frame (0, 0, 0)
  x(1): robot frame (1, 0, 0) / global frame (1, 0, 0)
  x(2): robot frame (3, 1, 0) / global frame (3, 1, 0)
  x(3): robot frame (4, 1, 0) / global frame (4, 1, 0)
 graph saved as: ./robot_1_opt_2.graph

Robot 2 graph:
  anchor transform wrt the global: (2, 1, 0)
  x(0): robot frame (0, 0, 0) / global frame (2, 1, 0)
  x(1): robot frame (2, 0, 0) / global frame (4, 1, 0)
  x(2): robot frame (3, 0, 0) / global frame (5, 1, 0)
 graph saved as: ./robot_2_opt_2.graph

Robot 3 graph:
  anchor transform wrt the global: (1, 1, -1.5708)
  x(0): robot frame (0, 0, 0) / global frame (1, 1, -1.5708)
  x(1): robot frame (-1, 1, 1.5708) / global frame (2, 2, 0)
 graph saved as: ./robot_3_opt_2.graph
```



This result is equivalent to our expected result at the slide 16

The robot 3's poses
are transformed into the global frame (i.e., robot 1)
using the relative between anchor node 1 to 3.

Using this eq.

global node pose = anchor pose ⊕ own (local) node pose

```
"/ global frame " << anchor_node_->value().oplus(nodes_.at(i).get()->value())
```

Final result

- In this example, finally, the optimized robot pose is save as .graph file
  - Overall graph
    - Including anchor node pose

```
Pose2d_Factor 0 (0, 0, 0) {10000,0,0,10000,0,10000}
Pose2d_Factor 1 (0, 0, 0) {10000,0,0,10000,0,10000}
Pose2d_Pose2d_Factor 1 2 (1, 0, 0) {10,0,0,10,0,10}
Pose2d_Pose2d_Factor 2 3 (2, 1, 0) {10,0,0,10,0,10}
Pose2d_Factor 5 (0, 0, 0) {10000,0,0,10000,0,10000}
Pose2d_Pose2d_Factor 3 6 (1, 0, 0) {10,0,0,10,0,10}
Pose2d_Pose2d_Factor 5 7 (2, 0, 0) {10,0,0,10,0,10}
Pose2d_Pose2d_Factor 2 5 0 4 (1, 1, 0) {10,0,0,10,0,10}
Pose2d_Factor 9 (0, 0, 0) {10000,0,0,10000,0,10000}
Pose2d_Pose2d_Factor 9 10 (-1, 1, 1.5708) {10,0,0,10,0,10}
Pose2d_Pose2d_Factor 7 11 (1, 0, 0) {10,0,0,10,0,10}
Pose2d_Pose2d_Factor 5 10 4 8 (0, 1, 0) {10,0,0,10,0,10}
Anchor2d_Node 0 (0, 0, 0)
Pose2d_Node 1 (0, 0, 0)
Pose2d_Node 2 (1, 0, 0)
Pose2d_Node 3 (3, 1, 0)
Anchor2d_Node 4 (2, 1, 0)
Pose2d_Node 5 (0, 0, 0)
Pose2d_Node 6 (4, 1, 0)
Pose2d_Node 7 (2, 0, 0)
Anchor2d_Node 8 (1, 1, -1.5708)
Pose2d_Node 9 (0, 0, 0)
Pose2d_Node 10 (-1, 1, 1.5708)
Pose2d_Node 11 (3, 0, 0)
```

Final result

- In this example, finally, the optimized robot pose is save as .graph file
  - Overall graph
    - Including anchor node pose
  - Each robot's own nodes' pose (within each one's frame)

Robot 1

```
1 Pose2d_Factor 1 (0, 0, 0) {10000,0,0,10000,0,10000}
2 Pose2d_Pose2d_Factor 1 2 (1, 0, 0) {10,0,0,10,0,10}
3 Pose2d_Pose2d_Factor 2 3 (2, 1, 0) {10,0,0,10,0,10}
4 Pose2d_Pose2d_Factor 3 6 (1, 0, 0) {10,0,0,10,0,10}
5 Pose2d_Node 1 (0, 0, 0)
6 Pose2d_Node 2 (1, 0, 0)
7 Pose2d_Node 3 (3, 1, 0)
8 Pose2d_Node 6 (4, 1, 0)
```

Robot 2

```
1 Pose2d_Factor 5 (0, 0, 0) {10000,0,0,10000,0,10000}
2 Pose2d_Pose2d_Factor 5 7 (2, 0, 0) {10,0,0,10,0,10}
3 Pose2d_Pose2d_Factor 7 11 (1, 0, 0) {10,0,0,10,0,10}
4 Pose2d_Node 5 (0, 0, 0)
5 Pose2d_Node 7 (2, 0, 0)
6 Pose2d_Node 11 (3, 0, 0)
```

Robot 3

```
1 Pose2d_Factor 9 (0, 0, 0) {10000,0,0,10000,0,10000}
2 Pose2d_Pose2d_Factor 9 10 (-1, 1, 1.5708) {10,0,0,10,0,10}
3 Pose2d_Node 9 (0, 0, 0)
4 Pose2d_Node 10 (-1, 1, 1.5708)
```

Advantages of anchor node-based multi-robot SLAM

- This submap-like technique reduces time for convergence
- Each robot could be added at arbitrary time
- Each robot can operate on its own frame
- No gauge freedom problem (i.e., each robot has its own prior / if not, graph convergence is not stable)
- Directly can be extend to the multi-session (or long-term) SLAM

Thank you!