The overview of PyICP SLAM

Giseop Kim
paulgkim@kaist.ac.kr

The overview of PyICP SLAM

```python
# @@@ MAIN @@@: data stream
for for_idx, scan_path in tqdm(enumerate(scan_paths), total=num_frames, mininterval=5.0):

    # get current information
    curr_scan_pts = Ptutils.readScan(scan_path)
    curr_scan_down_pts = Ptutils.random_sampling(curr_scan_pts, num_points=args.num_icp_points)

    # save current node
    PGM.curr_node_idx = for_idx # make start with 0
    SCM.addNode(node_idx=PGM.curr_node_idx, ptcloud=curr_scan_down_pts)
    if(PGM.curr_node_idx == 0):
        PGM.prev_node_idx = PGM.curr_node_idx
        prev_scan_pts = copy.deepcopy(curr_scan_pts)
        icp_initial = np.eye(4)
        continue

    # calc odometry
    prev_scan_down_pts = Ptutils.random_sampling(prev_scan_pts, num_points=args.num_icp_points)
    odom_transform, _, _ = ICP.icp(curr_scan_down_pts, prev_scan_down_pts, init_pose=icp_initial, max_iterations=20)

    # update the current (moved) pose
    PGM.curr_se3 = np.matmul(PGM.curr_se3, odom_transform)
    icp_initial = odom_transform # assumption: constant velocity model (for better next ICP converges)

    # add the odometry factor to the graph
    PGM.addOdometryFactor(odom_transform)

    # renewal the prev information
    PGM.prev_node_idx = PGM.curr_node_idx
    prev_scan_pts = copy.deepcopy(curr_scan_pts)

    # loop detection and optimize the graph
    if(PGM.curr_node_idx > 1 and PGM.curr_node_idx % args.try_gap_loop_detection == 0):
        # 1/ loop detection
        loop_idx, loop_dist, yaw_diff_deg = SCM.detectLoop()
        if(loop_idx == None): # NOT FOUND
            pass
        else:
            print("Loop event detected: ", PGM.curr_node_idx, loop_idx, loop_dist)
            # 2-1/ add the loop factor
            loop_scan_down_pts = SCM.getPtcloud(loop_idx)
            loop_transform, _, _ = ICP.icp(curr_scan_down_pts, loop_scan_down_pts, init_pose=yawdeg2se3(yaw_diff_deg), max_iterations=20)
            PGM.addLoopFactor(loop_transform, loop_idx)

            # 2-2/ graph optimization
            PGM.optimizedPoseGraph()
```

1-1. current data in

The overview of PyICP SLAM

```python
# @@@ MAIN @@@: data stream
for for_idx, scan_path in tqdm(enumerate(scan_paths), total=num_frames, mininterval=5.0):

    # get current information
    curr_scan_pts = Ptutils.readScan(scan_path)
    curr_scan_down_pts = Ptutils.random_sampling(curr_scan_pts, num_points=args.num_icp_points)

    # save current node
    PGM.curr_node_idx = for_idx # make start with 0
    SCM.addNode(node_idx=PGM.curr_node_idx, ptcloud=curr_scan_down_pts)
    if(PGM.curr_node_idx == 0):
        PGM.prev_node_idx = PGM.curr_node_idx
        prev_scan_pts = copy.deepcopy(curr_scan_pts)
        icp_initial = np.eye(4)
        continue

    # calc odometry
    prev_scan_down_pts = Ptutils.random_sampling(prev_scan_pts, num_points=args.num_icp_points)
    odom_transform, _, _ = ICP.icp(curr_scan_down_pts, prev_scan_down_pts, init_pose=icp_initial, max_iterations=20)

    # update the current (moved) pose
    PGM.curr_se3 = np.matmul(PGM.curr_se3, odom_transform)
    icp_initial = odom_transform # assumption: constant velocity model (for better next ICP converges)

    # add the odometry factor to the graph
    PGM.addOdometryFactor(odom_transform)

    # renewal the prev information
    PGM.prev_node_idx = PGM.curr_node_idx
    prev_scan_pts = copy.deepcopy(curr_scan_pts)

    # loop detection and optimize the graph
    if(PGM.curr_node_idx > 1 and PGM.curr_node_idx % args.try_gap_loop_detection == 0):
        # 1/ loop detection
        loop_idx, loop_dist, yaw_diff_deg = SCM.detectLoop()
        if(loop_idx == None): # NOT FOUND
            pass
        else:
            print("Loop event detected: ", PGM.curr_node_idx, loop_idx, loop_dist)
            # 2-1/ add the loop factor
            loop_scan_down_pts = SCM.getPtcloud(loop_idx)
            loop_transform, _, _ = ICP.icp(curr_scan_down_pts, loop_scan_down_pts, init_pose=yawdeg2se3(yaw_diff_deg), max_iterations=20)
            PGM.addLoopFactor(loop_transform, loop_idx)

            # 2-2/ graph optimization
            PGM.optimizedPoseGraph()
```

1-1. current data in

1-2. add a node to the graph

The overview of PyICP SLAM

```python
# @@@ MAIN @@@: data stream
for for_idx, scan_path in tqdm(enumerate(scan_paths), total=num_frames, mininterval=5.0):

    # get current information
    curr_scan_pts = Ptutils.readScan(scan_path)
    curr_scan_down_pts = Ptutils.random_sampling(curr_scan_pts, num_points=args.num_icp_points)

    # save current node
    PGM.curr_node_idx = for_idx # make start with 0
    SCM.addNode(node_idx=PGM.curr_node_idx, ptcloud=curr_scan_down_pts)
    if(PGM.curr_node_idx == 0):
        PGM.prev_node_idx = PGM.curr_node_idx
        prev_scan_pts = copy.deepcopy(curr_scan_pts)
        icp_initial = np.eye(4)
        continue

    # calc odometry
    prev_scan_down_pts = Ptutils.random_sampling(prev_scan_pts, num_points=args.num_icp_points)
    odom_transform, _, _ = ICP.icp(curr_scan_down_pts, prev_scan_down_pts, init_pose=icp_initial, max_iterations=20)

    # update the current (moved) pose
    PGM.curr_se3 = np.matmul(PGM.curr_se3, odom_transform)
    icp_initial = odom_transform # assumption: constant velocity model (for better next ICP converges)

    # add the odometry factor to the graph
    PGM.addOdometryFactor(odom_transform)

    # renewal the prev information
    PGM.prev_node_idx = PGM.curr_node_idx
    prev_scan_pts = copy.deepcopy(curr_scan_pts)

    # loop detection and optimize the graph
    if(PGM.curr_node_idx > 1 and PGM.curr_node_idx % args.try_gap_loop_detection == 0):
        # 1/ loop detection
        loop_idx, loop_dist, yaw_diff_deg = SCM.detectLoop()
        if(loop_idx == None): # NOT FOUND
            pass
        else:
            print("Loop event detected: ", PGM.curr_node_idx, loop_idx, loop_dist)
            # 2-1/ add the loop factor
            loop_scan_down_pts = SCM.getPtcloud(loop_idx)
            loop_transform, _, _ = ICP.icp(curr_scan_down_pts, loop_scan_down_pts, init_pose=yawdeg2se3(yaw_diff_deg), max_iterations=20)
            PGM.addLoopFactor(loop_transform, loop_idx)

            # 2-2/ graph optimization
            PGM.optimizedPoseGraph()
```

1-1. current data in

1-2. add a node to the graph

2. calculate the odometry and
add the odometry factor to the graph

The overview of PyICP SLAM

```python
# @@@ MAIN @@@: data stream
for for_idx, scan_path in tqdm(enumerate(scan_paths), total=num_frames, mininterval=5.0):

    # get current information
    curr_scan_pts = Ptutils.readScan(scan_path)
    curr_scan_down_pts = Ptutils.random_sampling(curr_scan_pts, num_points=args.num_icp_points)

    # save current node
    PGM.curr_node_idx = for_idx # make start with 0
    SCM.addNode(node_idx=PGM.curr_node_idx, ptcloud=curr_scan_down_pts)
    if(PGM.curr_node_idx == 0):
        PGM.prev_node_idx = PGM.curr_node_idx
        prev_scan_pts = copy.deepcopy(curr_scan_pts)
        icp_initial = np.eye(4)
        continue

    # calc odometry
    prev_scan_down_pts = Ptutils.random_sampling(prev_scan_pts, num_points=args.num_icp_points)
    odom_transform, _, _ = ICP.icp(curr_scan_down_pts, prev_scan_down_pts, init_pose=icp_initial, max_iterations=20)

    # update the current (moved) pose
    PGM.curr_se3 = np.matmul(PGM.curr_se3, odom_transform)
    icp_initial = odom_transform # assumption: constant velocity model (for better next ICP converges)

    # add the odometry factor to the graph
    PGM.addOdometryFactor(odom_transform)

    # renewal the prev information
    PGM.prev_node_idx = PGM.curr_node_idx
    prev_scan_pts = copy.deepcopy(curr_scan_pts)

    # loop detection and optimize the graph
    if(PGM.curr_node_idx > 1 and PGM.curr_node_idx % args.try_gap_loop_detection == 0):
        # 1/ loop detection
        loop_idx, loop_dist, yaw_diff_deg = SCM.detectLoop()
        if(loop_idx == None): # not found
            pass
        else:
            print("Loop event detected: ", PGM.curr_node_idx, loop_idx, loop_dist)
            # 2-1/ add the loop factor
            loop_scan_down_pts = SCM.getPtcloud(loop_idx)
            loop_transform, _, _ = ICP.icp(curr_scan_down_pts, loop_scan_down_pts, init_pose=yawdeg2se3(yaw_diff_deg), max_iterations=20)
            PGM.addLoopFactor(loop_transform, loop_idx)

            # 2-2/ graph optimization
            PGM.optimizedPoseGraph()
```

1-1. current data in

1-2. add a node to the graph

2. calculate the odometry and
add the odometry factor to the graph

3-1. try loop detection

The overview of PyICP SLAM

```python
# @@@ MAIN @@@: data stream
for for_idx, scan_path in tqdm(enumerate(scan_paths), total=num_frames, mininterval=5.0):

    # get current information
    curr_scan_pts = Ptutils.readScan(scan_path)
    curr_scan_down_pts = Ptutils.random_sampling(curr_scan_pts, num_points=args.num_icp_points)

    # save current node
    PGM.curr_node_idx = for_idx # make start with 0
    SCM.addNode(node_idx=PGM.curr_node_idx, ptcloud=curr_scan_down_pts)
    if(PGM.curr_node_idx == 0):
        PGM.prev_node_idx = PGM.curr_node_idx
        prev_scan_pts = copy.deepcopy(curr_scan_pts)
        icp_initial = np.eye(4)
        continue

    # calc odometry
    prev_scan_down_pts = Ptutils.random_sampling(prev_scan_pts, num_points=args.num_icp_points)
    odom_transform, _, _ = ICP.icp(curr_scan_down_pts, prev_scan_down_pts, init_pose=icp_initial, max_iterations=20)

    # update the current (moved) pose
    PGM.curr_se3 = np.matmul(PGM.curr_se3, odom_transform)
    icp_initial = odom_transform # assumption: constant velocity model (for better next ICP converges)

    # add the odometry factor to the graph
    PGM.addOdometryFactor(odom_transform)

    # renewal the prev information
    PGM.prev_node_idx = PGM.curr_node_idx
    prev_scan_pts = copy.deepcopy(curr_scan_pts)

    # loop detection and optimize the graph
    if(PGM.curr_node_idx > 1 and PGM.curr_node_idx % args.try_gap_loop_detection == 0):
        # 1/ loop detection
        loop_idx, loop_dist, yaw_diff_deg = SCM.detectLoop()
        if(loop_idx == None): # not found
            pass
        else:
            print("Loop event detected: ", PGM.curr_node_idx, loop_idx, loop_dist)
            # 2-1/ add the loop factor
            loop_scan_down_pts = SCM.getPtcloud(loop_idx)
            loop_transform, _, _ = ICP.icp(curr_scan_down_pts, loop_scan_down_pts, init_pose=yawdeg2se3(yaw_diff_deg), max_iterations=20)
            PGM.addLoopFactor(loop_transform, loop_idx)

            # 2-2/ graph optimization
            PGM.optimizedPoseGraph()
```

1-1. current data in

1-2. add a node to the graph

2. calculate the odometry and
add the odometry factor to the graph

3-1. try loop detection

3-2. if a loop is found,
- calculate the loop factor
- add to the graph
- optimized the graph