TUGAS UAS ROBOTIKA

Diajukan untuk memenuhi tugas pengganti Ujian Akhir Semester (UAS)

pada mata kuliah Robotika

Disusun oleh :

Wening Alfina Rosunika

1103204017

PROGRAM STUDI TEKNIK KOMPUTER

FAKULTAS TEKNIK ELEKTRO

UNIVERSITAS TELKOM

2023

Buku "A Systematic Approach to Learning Robot Programming with ROS" oleh Wyatt Newman merupakan panduan praktis yang dirancang untuk membantu pembaca mempelajari pemrograman robot menggunakan Robot Operating System (ROS). Buku ini mengikuti pendekatan yang sistematis dalam mengajarkan konsep dan teknik pemrograman yang terkait dengan ROS.

Berikut adalah beberapa topik yang dibahas dalam buku ini:

1. Pengenalan ROS: Buku ini memberikan pemahaman dasar tentang ROS, menjelaskan arsitektur, komponen, dan alat yang terkait dengan ROS. Pembaca akan mempelajari cara menginstal ROS dan memahami struktur dasar sistem.

2. Sistem Koordinat dan Transformasi: Buku ini menjelaskan tentang konsep sistem koordinat dan transformasi dalam konteks robotika. Pembaca akan belajar bagaimana bekerja dengan frame koordinat, melakukan transformasi, dan mengelola kerangka kerja koordinat dalam ROS.

3. Pengendali Gerak: Buku ini membahas pengendalian gerakan robot, termasuk gerakan dasar dan pergerakan yang kompleks. Pembaca akan mempelajari cara membuat pengendali gerakan menggunakan ROS dan bagaimana mengintegrasikannya dengan sensor dan aktuator.

4. Pengolahan Citra: Topik ini menjelaskan pengolahan citra dalam konteks robotika. Buku ini membahas teknik dasar pengolahan citra seperti segmentasi, deteksi objek, dan pencocokan fitur. Pembaca juga akan mempelajari cara menggunakan ROS untuk mengakuisisi dan memproses citra dalam aplikasi robotika.

5. Navigasi Robot: Buku ini menjelaskan tentang navigasi robot menggunakan ROS Navigation Stack. Pembaca akan mempelajari cara membuat peta lingkungan, melakukan pemetaan dan pemantauan lokasi, serta merencanakan dan menjalankan pergerakan robot.

Selain topik-topik di atas, buku ini juga mencakup latihan dan contoh kode yang membantu pembaca dalam mempraktikkan konsep dan teknik yang diajarkan. Buku ini ditujukan untuk membantu pembaca memperoleh pemahaman yang kuat tentang pemrograman robot dengan ROS melalui pendekatan yang terstruktur dan praktis.

[Laporan Teknis]

Judul: Pendekatan Sistematis dalam Pembelajaran Pemrograman Robot dengan ROS

Abstrak:
Laporan teknis ini memberikan gambaran dan analisis dari buku "Pendekatan Sistematis dalam Pembelajaran Pemrograman Robot dengan ROS." Laporan ini merangkum konten setiap bab dan menyediakan contoh kode dari buku tersebut. Buku ini mencakup berbagai topik terkait pemrograman robot menggunakan kerangka kerja Robot Operating System (ROS). Laporan ini bertujuan untuk memberikan pemahaman menyeluruh tentang isi buku dan implikasi praktisnya dalam bidang robotika.

1. Pendahuluan:
Laporan ini dimulai dengan pendahuluan tentang buku dan tujuannya. Hal ini menyoroti pentingnya mempelajari pemrograman robot dengan ROS dan mengatur konteks untuk bab-bab selanjutnya.

2. Bab 1: Pengenalan ROS: Alat dan Node ROS
Ikhtisar Bab: Bab ini memperkenalkan konsep dasar ROS dan memberikan gambaran tentang alat dan node ROS. Ini mencakup topik seperti membuat paket ROS, menulis node ROS, mengompilasi dan menjalankan node ROS, serta menggunakan alat-alat ROS seperti Catkin Simple, roslaunch, rqt console, dan rosbag.

Contoh Kode:
Contoh kode yang diberikan dalam bab ini menjelaskan proses membuat publisher dan subscriber ROS minimal, mengompilasi dan menjalankan node, serta menggunakan alat-alat ROS untuk menyederhanakan pengelolaan paket dan perekaman data.

```
#!/usr/bin/env python

import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo("Received message: %s", data.data)

def minimal_subscriber():
    rospy.init_node('minimal_subscriber', anonymous=True)
    rospy.Subscriber('chatter', String, callback)
    rospy.spin()

if __name__ == '__main__':
    minimal_subscriber()
```

3. Bab 2: Pesan, Kelas, dan Server

Ikhtisar Bab: Bab ini berfokus pada definisi pesan kustom, penggunaan layanan ROS, penggunaan kelas C++ dalam ROS, pembuatan modul perpustakaan, dan pengenalan tentang server dan klien aksi. Bab ini juga membahas konsep parameter server dalam ROS.

Contoh Kode:

Contoh kode dalam bab ini mencakup contoh definisi pesan kustom, implementasi node layanan ROS, interaksi manual dengan layanan ROS, pembuatan paket server aksi, definisi pesan server aksi kustom, dan desain klien aksi.

```
#!/usr/bin/env python

import rospy
from std_msgs.msg import Int32
from beginner_tutorials.srv import AddTwoInts, AddTwoIntsRequest

def callback(data):
    rospy.loginfo("Received message: %d", data.data)

def service_client(x, y):
    rospy.wait_for_service('add_two_ints')
    try:
        add_two_ints = rospy.ServiceProxy('add_two_ints', AddTwoInts)
        request = AddTwoIntsRequest()
        request.a = x
        request.b = y
        response = add_two_ints(request)
        return response.sum
    except rospy.ServiceException as e:
        rospy.logerr("Service call failed: %s", str(e))

def minimal_node():
    rospy.init_node('minimal_node', anonymous=True)
    rospy.Subscriber('number', Int32, callback)
    result = service_client(2, 3)
    rospy.loginfo("Result of adding two integers: %d", result)
    rospy.spin()

if __name__ == '__main__':
    minimal_node()
```

4. Bab 3: Simulasi dalam ROS

Ikhtisar Bab: Bab ini menjelajahi teknik simulasi dalam ROS. Mencakup topik seperti simulator robot dua dimensi sederhana, pemodelan untuk simulasi dinamis, format deskripsi robot yang terpadu, pengenalan Gazebo (simulator ROS populer), pengendali persendian minimal, penggunaan plugin Gazebo untuk pengendalian servomotor persendian, pembuatan model robot bergerak, simulasi model robot bergerak, dan menggabungkan model robot.

Contoh Kode:

Contoh kode dalam bab ini mencakup contoh membangun simulator robot dua dimensi sederhana, membuat pengendali persendian minimal, menggunakan plugin Gazebo untuk pengendalian servomotor persendian, dan melakukan simulasi model robot bergerak di Gazebo.

```
#!/usr/bin/env python

import rospy
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan

def callback(data):
    # Mendapatkan jarak terdekat dari data pemindaian laser
    ranges = data.ranges
    min_range = min(ranges)

    # Menghindari rintangan dengan menggerakkan robot secara acak
    if min_range > 1.0:
        move_forward()
    else:
        avoid_obstacle()

def move_forward():
    # Menggerakkan robot maju dengan kecepatas linear 0.2
    velocity_msg = Twist()
    velocity_msg.linear.x = 0.2
    velocity_publisher.publish(velocity_msg)

def avoid_obstacle():
    # Menghindari rintangan dengan berputar ke arah yang berlawanan jarum jam
    velocity_msg = Twist()
    velocity_msg.angular.z = 0.2
    velocity_publisher.publish(velocity_msg)

def simulation_node():
    rospy.init_node('simulation_node', anonymous=True)
    rospy.Subscriber('scan', LaserScan, callback)
    rospy.spin()

if __name__ == '__main__':
    velocity_publisher = rospy.Publisher('cmd_vel', Twist, queue_size=10)
    simulation_node()
```

5. Bab 4: Transformasi Koordinat dalam ROS

Ikhtisar Bab: Bab ini memperkenalkan transformasi koordinat dalam ROS dan mencakup topik seperti transform listener, penggunaan pustaka Eigen, dan transformasi jenis data ROS.

Contoh Kode:

Contoh kode dalam bab ini mencakup implementasi transform listener untuk mendapatkan transformasi koordinat, penggunaan pustaka Eigen untuk operasi transformasi, dan contoh transformasi data ROS.

```
#!/usr/bin/env python

import rospy
import tf2_ros
import geometry_msgs.msg

if __name__ == '__main__':
    rospy.init_node('coordinate_transform_node')

    tf_buffer = tf2_ros.Buffer()
    tf_listener = tf2_ros.TransformListener(tf_buffer)

    rate = rospy.Rate(10.0)
    while not rospy.is_shutdown():
        try:
            # Mendapatkan transformasi dari 'frame1' ke 'frame2'
            transform = tf_buffer.lookup_transform('frame2', 'frame1', rospy.Time())

            # Membuat titik dengan koordinat relatif terhadap 'frame1'
            point = geometry_msgs.msg.PointStamped()
            point.header.frame_id = 'frame1'
            point.point.x = 1.0
            point.point.y = 0.0
            point.point.z = 0.0

            # Melakukan transformasi titik ke 'frame2'
            transformed_point = tf_buffer.transform(point, 'frame2')

            rospy.loginfo('Transformed Point: (%f, %f, %f) in frame2',
                    transformed_point.point.x, transformed_point.point.y, transformed_point.point.z)


except(tf2_ros.LookupException,tf2_ros.ConnectivityException,tf2_ros.ExtrapolationException):
            rospy.logwarn('Failed to perform coordinate transform')

     rate.sleep()
```

6. Bab 5: Sensor dan Visualisasi dalam ROS

Ikhtisar Bab: Bab ini membahas penggunaan marker dan interactive marker dalam RViz, tampilan nilai sensor dalam RViz, simulasi dan tampilan data LIDAR, kamera warna, dan kamera kedalaman dalam RViz, serta seleksi titik dalam RViz.

Contoh Kode:

Contoh kode dalam bab ini mencakup contoh penggunaan marker dan interactive marker dalam RViz, simulasi dan tampilan data sensor seperti LIDAR, kamera warna, dan kamera kedalaman dalam RViz, serta contoh seleksi titik dalam RViz.

```
#!/usr/bin/env python

import rospy
from visualization_msgs.msg import Marker
from interactive_markers.interactive_marker_server import *
from geometry_msgs.msg import Point

def create_marker():
    marker = Marker()
    marker.header.frame_id = "base_link"
    marker.header.stamp = rospy.Time.now()
    marker.ns = "my_markers"
    marker.id = 0
    marker.type = Marker.CUBE
    marker.action = Marker.ADD
    marker.pose.position = Point(1.0, 2.0, 0.5)
    marker.scale.x = 0.2
    marker.scale.y = 0.2
    marker.scale.z = 0.2
    marker.color.r = 1.0
    marker.color.g = 0.0
    marker.color.b = 0.0
    marker.color.a = 1.0
    return marker

def create_interactive_marker():
    server = InteractiveMarkerServer("my_markers")
    int_marker = InteractiveMarker()
    int_marker.header.frame_id = "base_link"
    int_marker.name = "my_marker"
    int_marker.pose.position = Point(2.0, -1.0, 0.5)
    int_marker.scale = 1.0

    box_marker = Marker()
    box_marker.type = Marker.CUBE
    box_marker.pose.position = Point(0, 0, 0)
    box_marker.scale.x = 0.2
    box_marker.scale.y = 0.2
    box_marker.scale.z = 0.2
    box_marker.color.r = 0.0
    box_marker.color.g = 1.0
    box_marker.color.b = 0.0
    box_marker.color.a = 1.0
```

```
    int_marker.controls.append(MarkerControl())
    int_marker.controls[0].interaction_mode = InteractiveMarkerControl.MOVE_ROTATE
    int_marker.controls[0].markers.append(box_marker)
    server.insert(int_marker, None)
    server.applyChanges()

if __name__ == '__main__':
    rospy.init_node('sensing_visualization_node')

    marker_publisher = rospy.Publisher('visualization_marker', Marker, queue_size=10)

    rate = rospy.Rate(1)
    while not rospy.is_shutdown():
        marker = create_marker()
        marker_publisher.publish(marker)
        rospy.loginfo('Published marker')

        create_interactive_marker()
        rospy.loginfo('Created interactive marker')

        rate.sleep()
```

7. Bab 6: Menggunakan Kamera dalam ROS

Ikhtisar Bab: Bab ini membahas transformasi proyektif ke koordinat kamera, kalibrasi intrinsik kamera, kalibrasi intrinsik kamera stereo, dan penggunaan OpenCV dengan ROS.

Contoh Kode:

Contoh kode dalam bab ini mencakup contoh transformasi proyektif ke koordinat kamera, kalibrasi intrinsik kamera, kalibrasi intrinsik kamera stereo, serta contoh penggunaan OpenCV untuk pemrosesan citra berdasarkan warna dan tepi.

```
#!/usr/bin/env python

import rospy
import cv2
from sensor_msgs.msg import Image
from cv_bridge import CvBridge

def image_callback(msg):
    bridge = CvBridge()
    try:
        # Konversi pesan gambar ROS menjadi citra OpenCV
        cv_image = bridge.imgmsg_to_cv2(msg, "bgr8")
    except Exception as e:
        rospy.logerr(e)
        return

    # Proses citra
    # Misalnya, menampilkan citra dan menunggu tombol 'Esc' untuk keluar
    cv2.imshow("Image", cv_image)
    key = cv2.waitKey(1) & 0xFF
    if key == 27:  # Tombol 'Esc'
        cv2.destroyAllWindows()

if __name__ == '__main__':
    rospy.init_node('camera_node')

    # Menggunakan topik 'camera/image' untuk menerima citra dari kamera
    image_subscriber = rospy.Subscriber('camera/image', Image, image_callback)

    rospy.spin()
```

8. Bab 7: Pemetaan Kedalaman dan Awan Titik
Ikhtisar Bab: Bab ini membahas pemetaan kedalaman dari LIDAR pemindaian, pemetaan kedalaman dari kamera stereo, dan pengenalan kamera kedalaman.

Contoh Kode:
Contoh kode dalam bab ini mencakup contoh pemetaan kedalaman dari LIDAR pemindaian, pemetaan kedalaman dari kamera stereo, serta contoh penggunaan kamera kedalaman.

```
#!/usr/bin/env python

import rospy
from sensor_msgs.msg import PointCloud2

def point_cloud_callback(msg):
    # Proses data point cloud di sini
    # Misalnya, dapatkan koordinat XYZ dari titik-titik dalam point cloud
    for point in pc2.read_points(msg, field_names=("x", "y", "z"), skip_nans=True):
        x, y, z = point
        # Lakukan operasi lainnya dengan koordinat XYZ

if __name__ == '__main__':
    rospy.init_node('point_cloud_node')

    # Menggunakan topik 'point_cloud' untuk menerima data point cloud
    point_cloud_subscriber = rospy.Subscriber('point_cloud', PointCloud2, point_cloud_callback)

    rospy.spin()
```

9. Bab 8: Pemrosesan Awan Titik

Ikhtisar Bab: Bab ini membahas pemrosesan awan titik dalam ROS, termasuk visualisasi awan titik, membaca dan menampilkan gambar awan titik dari disk, menyimpan gambar awan titik yang dipublikasikan ke disk, interpretasi gambar awan titik dengan menggunakan metode PCL, dan pencarian objek.

Contoh Kode:
Contoh kode dalam bab ini mencakup contoh visualisasi awan titik, membaca dan menampilkan gambar awan titik dari disk, menyimpan gambar awan titik ke disk, dan menggunakan metode PCL untuk interpretasi gambar awan titik dan pencarian objek.

```
#!/usr/bin/env python

import rospy
from sensor_msgs.msg import PointCloud2
import pcl

def point_cloud_callback(msg):
    # Mengonversi pesan PointCloud2 ke objek pcl.PointCloud
    cloud = pcl.PointCloud()
    cloud.fromROSMsg(msg)

    # Proses pemrosesan pada titik-titik dalam point cloud
    # Misalnya, melakukan segmentasi atau filtering

    # Mengonversi kembali objek pcl.PointCloud ke pesan PointCloud2
    filtered_cloud_msg = cloud.toROSMsg()

    # Mengirimkan point cloud yang telah diproses ke topik lain
    filtered_cloud_publisher.publish(filtered_cloud_msg)

if __name__ == '__main__':
    rospy.init_node('point_cloud_processing_node')

    # Menggunakan topik 'point_cloud' untuk menerima data point cloud
    point_cloud_subscriber = rospy.Subscriber('point_cloud', PointCloud2, point_cloud_callback)

    # Menggunakan topik 'filtered_cloud' untuk mengirimkan point cloud yang telah diproses
    filtered_cloud_publisher = rospy.Publisher('filtered_cloud', PointCloud2, queue_size=10)

    rospy.spin()
```

Kesimpulan:
Laporan ini menyimpulkan dengan merangkum temuan utama dari setiap bab dan menekankan pentingnya memahami konsep dan teknik yang dibahas dalam buku ini untuk mengembangkan aplikasi robot yang canggih.