Odometry

WeGo & WeCAR

WeGo

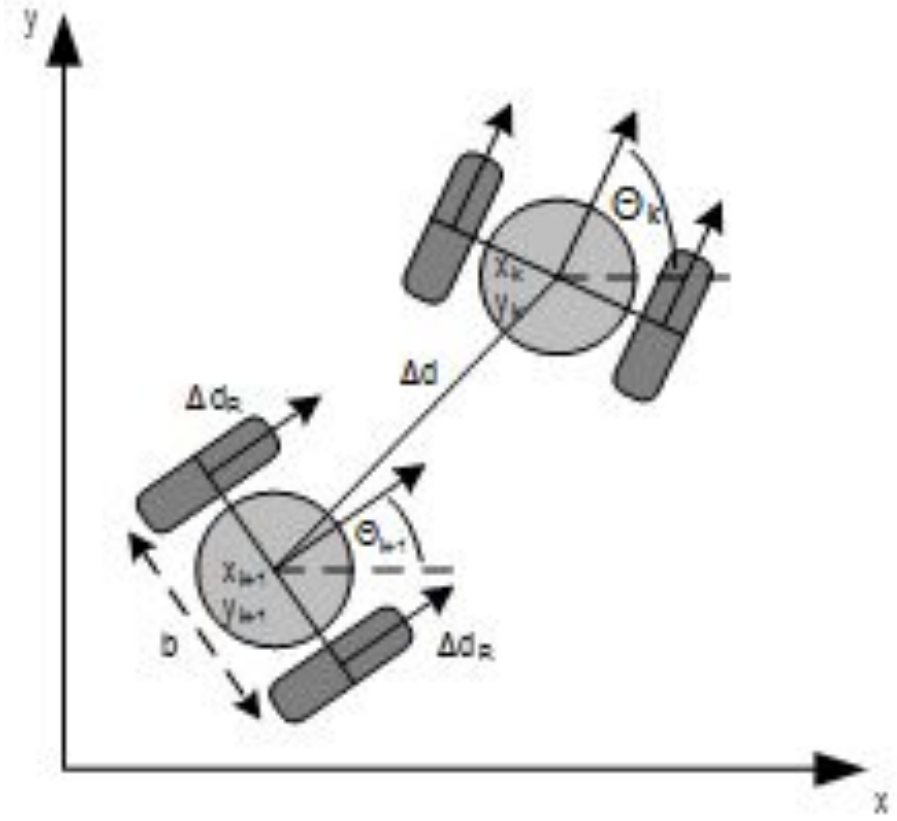
목 차

WeGo

01 Odometry

01 Odometry

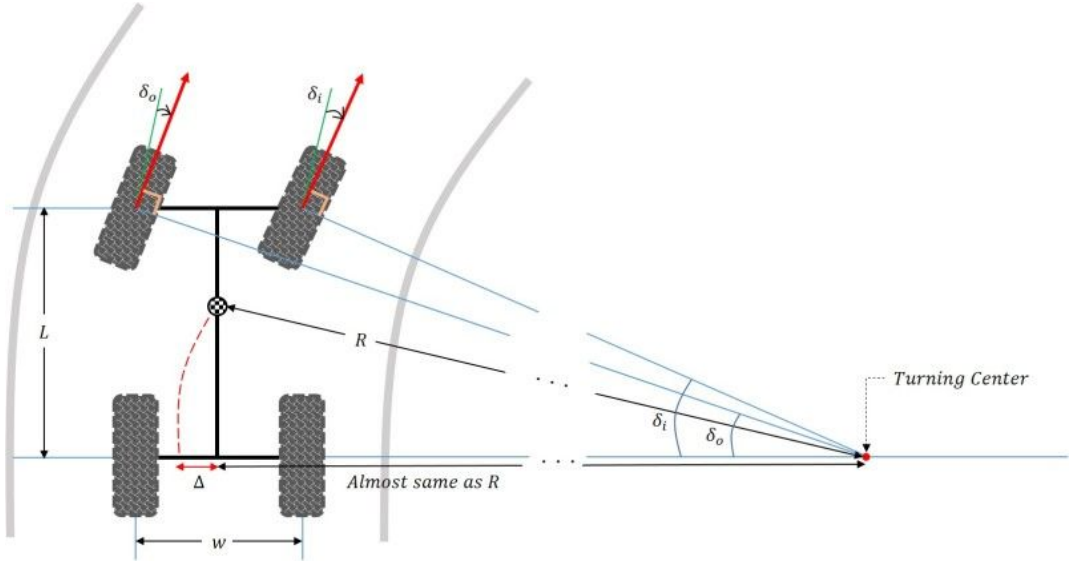
- Odometry는 말 그대로 주행기록계라는 의미
- 바퀴의 회전수(Encoder), IMU(관성 측정 장치) 등을 이용
- 위의 센서를 기반으로 움직이는 물체의 위치를 측정하는 방법



WeGo

01 Odometry

- Odometry를 계산하기 위해서는 바퀴의 회전을 통해 이동을 계산하는 Model이 필요하며, Mobile Robot의 경우, Ackermann Steering Model과 Differential Drive Model의 두 가지가 대표적



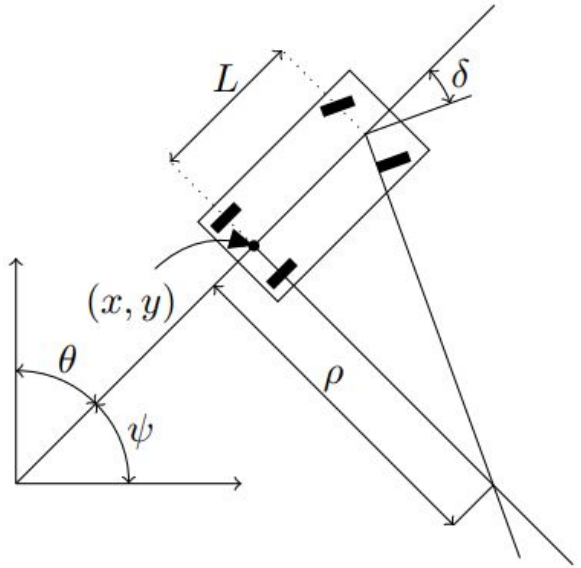
Ackermann Steering Model

WeGo

01 Odometry

- Ackermann Steering Model
- 일반적인 후방 구동, 전방 조향의 차량 모델
- 전방 조향을 통해 원 궤적을 그리는 형태로 주행할 수 있음
- 역으로 계산 시, 원하는 위치에 이동할 때, 어느 정도의 조향각이 필요한지 계산할 수 있음

  (Pure Pursuit)



$$\dot{x} = v\cos(\psi) = v\cos\left(\frac{\pi}{2} - \theta\right)$$

$$\dot{y} = v\sin(\psi) = v\sin\left(\frac{\pi}{2} - \theta\right)$$

$$\dot{\theta} = \frac{v}{L}\tan(\delta).$$

WeGo

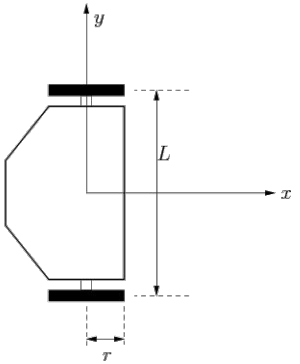
01 Odometry

- Differential Drive Model
- 각 바퀴의 제어가 가능하며, 바퀴 사이의 속도 차이를 이용하여,
  전, 후진 및 회전을 제어할 수 있는 형태

$$\dot{x} = \frac{r}{2}(v_l + v_r)cos(\theta)$$
$$\dot{y} = \frac{r}{2}(v_l + v_r)sin(\theta)$$
$$\dot{\theta} = \frac{r}{L}(v_r - v_l)$$

$$\dot{x} = vcos(\phi)$$
$$\dot{y} = vsin(\phi)$$
$$\dot{\phi} = \omega$$

(a)



(b)

WeGo

01 Odometry

- Dead Reckoning
- Odometry와 유사하지만, 실외에서 GPS가 연결이 끊기거나 하는 경우, 차량 Model 및 차량 속도, Heading Angle 등의 정보를 이용하여, 차량의 위치를 추정하는 방법
- 역할 자체는 동일하며, 목적에 따라 용어가 달라짐

WeGo

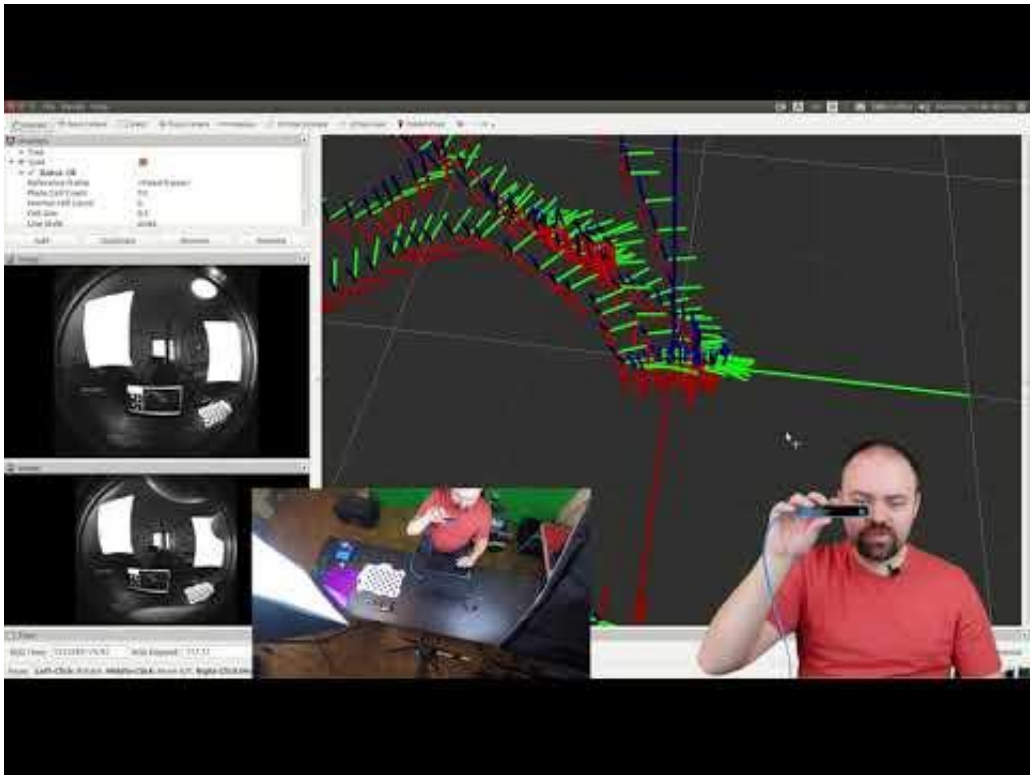
01 Odometry

- Visual Odometry
- Odometry를 Encoder 및 IMU 등 실제 이동에 해당하는 정보를 이용하여 진행하는 것이 아닌, 카메라 영상의 Frame과 Frame 차이를 비교하는 방식을 통해 Odometry를 계산하는 기술
- 카메라를 이용하여 6-DOF의 궤적을 계산
- 이동속도가 너무 빠르거나, Frame Drop 등이 발생할 때, 정확도가 급격히 감소
- Wheel Odometry와 IMU Odometry, Visual Odometry를 모두 퓨전해서 사용하면 정확도 향상에 큰 도움이 됨

WeGo

01 Odometry

- T265 Tracking Camera
  - Intel RealSense 계열이며, 내부적으로 Visual Odometry 기능 및 Visual SLAM 기능이 포함되어 있는 카메라
  - http://www.youtube.com/watch?v=GhHvuAoFC6I



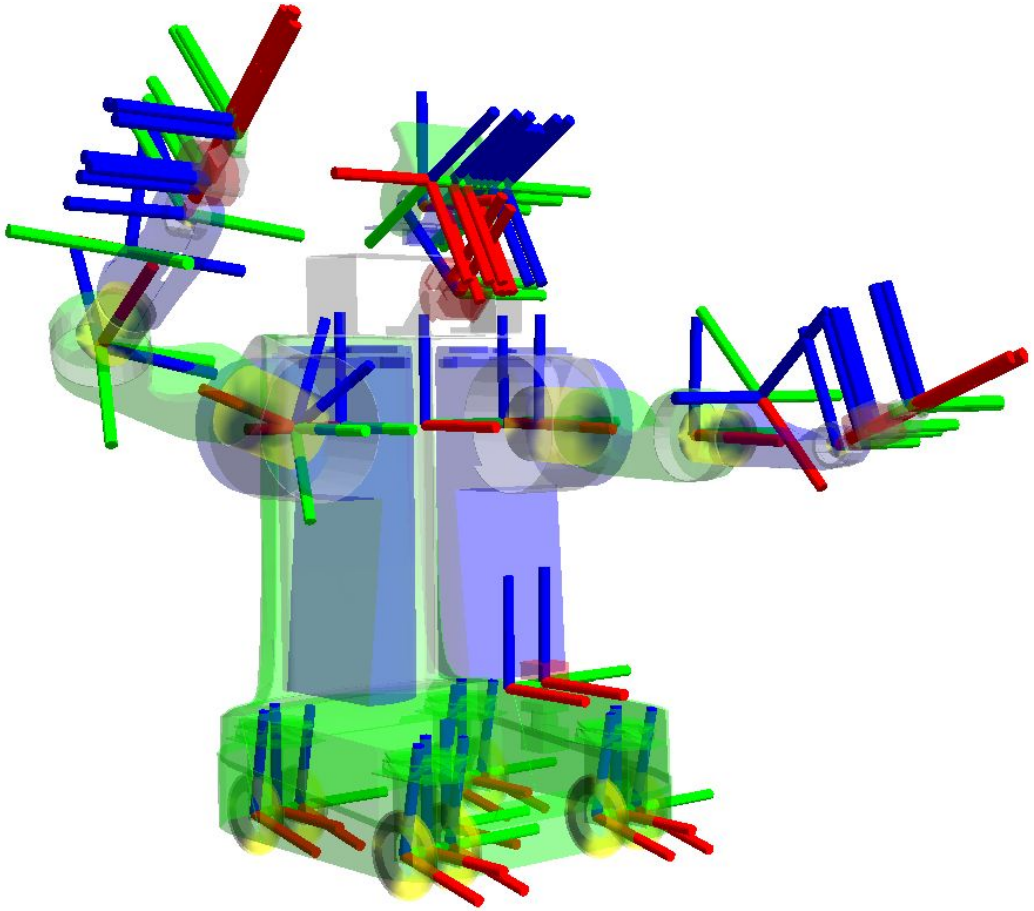
WeGo

02

Transformation

02 Transformation

- Transformation
  - TF는 Frame 사이의 변환 관계를 나타내는 Topic



WeGo

02 Transformation

- Transformation
  - 3D - 3D 사이의 좌표 변환 관계를 나타내므로.

    실제 필요한 값은 6-DOF (x. y. z. roll. pitch. yaw)의 6가지
  - 변환 관계를 모두 정리하면 Tree의 형태로 확인할 수 있다.
  - 일반적으로 특정 Reference Frame을 기준으로 Sensor Data값이 어떻게 들어오는지

    변환하여 사용하거나. Odometry 정보를 나타낼 때 사용한다.

02 Transformation

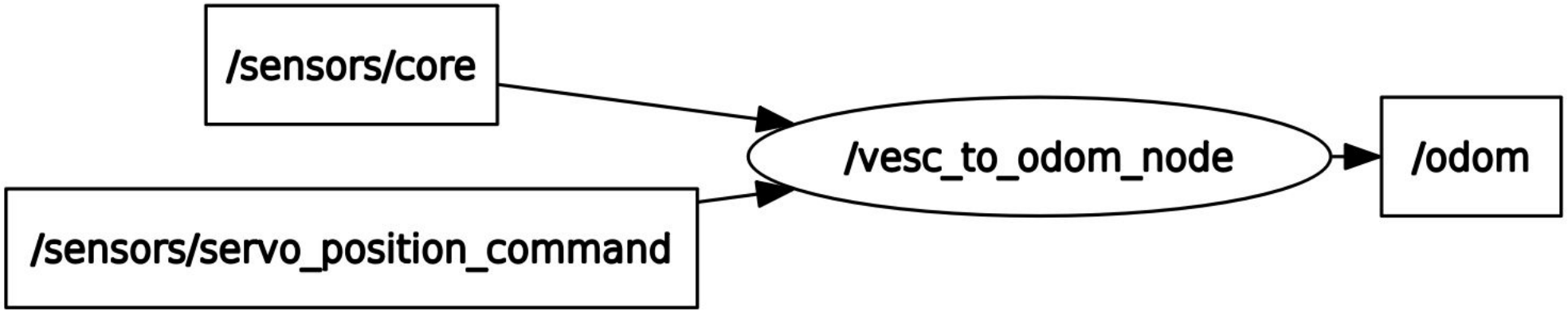
- Transformation
  - TF Demo
    - $ sudo apt install ros-melodic-turtle-tf2 ros-melodic-tf2-tools ros-melodic-tf
    - $ roslaunch turtle_tf2 turtle_tf2_demo.launch
    - http://wiki.ros.org/tf2
    - https://github.com/ros-industrial-consortium/descartes_tutorials/issues/14
    - https://github.com/ros/geometry_tutorials\

03 WeCAR Odometry

WeGo

03 WeCAR Odometry

- vesc_to_odom node
- Subscribed Topics
  - sensors/core(vesc_msgs/VescStateStamped) - Vesc 선속도 확인
  - sensors/servo_position_command(std_msgs/Float64) - Vesc 조향 확인
- Published Topics
  - odom (nav_msgs/Odometry) - 생성된 Odometry 값 (x, y, yaw)
  - tf (tf2_msgs/TFMessage) - 생성된 Transformation 값



03 WeCAR Odometry

- vesc_to_odom node
- Parameters
  - /speed_to_erpm_gain - 속도(m/s)를 ERPM으로 변환할 때 사용하는 Gain값
  - /speed_to_erpm_offset - 속도(m/s)를 ERPM으로 변환할 때 사용하는 Offset값
  - /steering_angle_to_servo_gain - 조향(rad)을 servo motor command(0~1)로 변환할 때 사용하는 Gain값
  - /steering_angle_to_servo_offset - 조향(rad)을 servo motor command(0~1)로 변환할 때 사용하는 Offset값
  - /vesc_to_odom/publish_tf - TF 생성 여부
  - /wheelbase - Ackerman Steering Model State Space Equation 계산 시, 사용

WeGo

03 WeCAR Odometry

```
private_nh.param("odom_frame", odom_frame_, odom_frame_);      Michael Boulet, 5 years ago •
private_nh.param("base_frame", base_frame_, base_frame_);
private_nh.param("use_servo_cmd_to_calc_angular_velocity", use_servo_cmd_, use_servo_cmd_);
if (!getRequiredParam(nh, "speed_to_erpm_gain", speed_to_erpm_gain_))
  return;
if (!getRequiredParam(nh, "speed_to_erpm_offset", speed_to_erpm_offset_))
  return;
if (use_servo_cmd_) {
  if (!getRequiredParam(nh, "steering_angle_to_servo_gain", steering_to_servo_gain_))
    return;
  if (!getRequiredParam(nh, "steering_angle_to_servo_offset", steering_to_servo_offset_))
    return;
  if (!getRequiredParam(nh, "wheelbase", wheelbase_))
    return;
}
private_nh.param("publish_tf", publish_tf_, publish_tf_);
```

```
odom_pub_ = nh.advertise<nav_msgs::Odometry>("odom", 10);

// create tf broadcaster
if (publish_tf_) {
  tf_pub_.reset(new tf::TransformBroadcaster);
}

// subscribe to vesc state and. optionally, servo command
vesc_state_sub_ = nh.subscribe("sensors/core", 10, &VescToOdom::vescStateCallback, this);
if (use_servo_cmd_) {
  servo_sub_ = nh.subscribe("sensors/servo_position_command", 10,
                            &VescToOdom::servoCmdCallback, this);
}
```

WeGo

03 WeCAR Odometry

```
ros::Duration dt = state->header.stamp - last_state_->header.stamp;

/** @todo could probably do better propigating odometry, e.g. trapezoidal integration */

// propigate odometry
double x_dot = current_speed * cos(yaw_);
double y_dot = current_speed * sin(yaw_);
x_ += x_dot * dt.toSec();
y_ += y_dot * dt.toSec();
if (use_servo_cmd_)
  yaw_ += current_angular_velocity * dt.toSec();

// save state for next time
last_state_ = state;
```

WeGo

03 WeCAR Odometry

```
nav_msgs::Odometry::Ptr odom(new nav_msgs::Odometry);
odom->header.frame_id = odom_frame_;
odom->header.stamp = state->header.stamp;
odom->child_frame_id = base_frame_;

// Position      Michael Boulet, 5 years ago • add vesc_ackermann package for translating betwee…
odom->pose.pose.position.x = x_;
odom->pose.pose.position.y = y_;
odom->pose.pose.orientation.x = 0.0;
odom->pose.pose.orientation.y = 0.0;
odom->pose.pose.orientation.z = sin(yaw_/2.0);
odom->pose.pose.orientation.w = cos(yaw_/2.0);

// Position uncertainty
/** @todo Think about position uncertainty, perhaps get from parameters? */
odom->pose.covariance[0]  = 0.2; ///< x
odom->pose.covariance[7]  = 0.2; ///< y
odom->pose.covariance[35] = 0.4; ///< yaw

// Velocity ("in the coordinate frame given by the child_frame_id")
odom->twist.twist.linear.x = current_speed;
odom->twist.twist.linear.y = 0.0;
odom->twist.twist.angular.z = current_angular_velocity;
```

WeGo

03 WeCAR Odometry

```
if (publish_tf_) {
  geometry_msgs::TransformStamped tf;
  tf.header.frame_id = odom_frame_;
  tf.child_frame_id = base_frame_;
  tf.header.stamp = ros::Time::now();
  tf.transform.translation.x = x_;
  tf.transform.translation.y = y_;
  tf.transform.translation.z = 0.0;
  tf.transform.rotation = odom->pose.pose.orientation;
  if (ros::ok()) {
    tf_pub_->sendTransform(tf);
  }
}

if (ros::ok()) {
  odom_pub_.publish(odom);
}
```

WeGo

03 WeCAR Odometry

- wecar_odom.py
- Subscribed Topics
  - /imu(sensor_msgs/Imu) - imu 값을 확인할 수 있는 Topic
  - sensors/core(vesc_msgs/VescStateStamped) - Vesc 선속도 확인
- Published Topics
  - odom (nav_msgs/Odometry) - 생성된 Odometry 값 (x, y, yaw)
  - tf (tf2_msgs/TFMessage) - 생성된 Transformation 값
- Parameters
  - /speed_to_erpm_gain - 속도(m/s)를 ERPM으로 변환할 때 사용하는 Gain값
- wecar_odom.py의 경우, 선속도는 VESC의 값을 받으며,

  회전은 IMU의 회전 속도를 이용하여, Odometry를 생성
- $ rosrun wecar wecar_odom.py

WeGo

WeGo Robotics

Tel. 031 – 229 – 3553

Fax. 031 – 229 – 3554

제품 문의: go.sales@wego-robotics.com

기술 문의: go.support@wego-robotics.com