SYMFORCE 2D

AGUSTIO

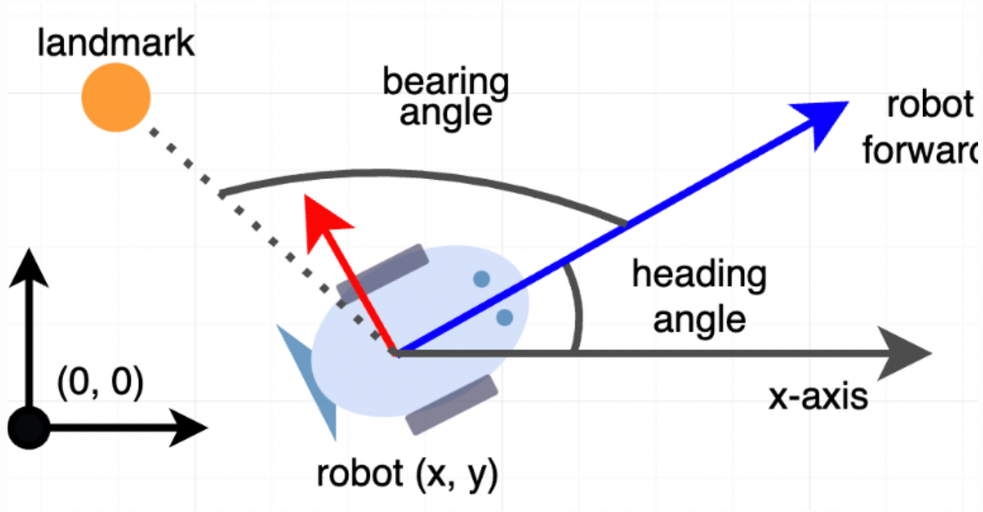
ABOUT SYM

SymForce merupakan sebuah code generation library yang digunakan untuk komputasi simbolik cepat dan sering digunakan untuk pengaplikasian metode robotik. Pada umumnya Symforce akan mengkombinasikan development speed dan fleksibelitas simbolik dalam matematika dengan performa yang autogenerated. Untuk implementasi akan lebih optimal menggunakan bahasa pemrograman C++ atau bahasa pemrograman yang memiliki runtime yang cepat

INSTALL LIBRARY

01 pip install symforce

02 pip install numpy



STUDI CASE

obot bergerak melalui bidang 2D dan memiliki goal untuk memperkirakan pose pada beberapa langkah kedepan dengan sebuah pengukuran kebisingan (noisy measurements).
Pengukuran yang harus dilakukan oleh Robot :

- Jarak yang ditempuh dengan menggunakan sensor odometri
- Sudut relatif menuju landmark.

Berdasarkan kasus di atas, maka robot memiiki sudut dengan tujuan yang berlawanan arah jarum jam dari sumbu x. Oleh karena itu, pengukuran sudut relatif ditentukan dari arah robot akan maju.

IMPORT LIBRARY

```python
import symforce

symforce.set_epsilon_to_symbol()

# ----------------------------------------------------------------------
# Import Library
# ----------------------------------------------------------------------
import numpy as np

from symforce import typing as T
from symforce.values import Values
import symforce.symbolic as sf
```

INISIALISASI NILAI AWAI

```python

def build_initial_values() -> T.Tuple[Values, int, int]:

    num_poses = 3
    num_landmarks = 3

    initial_values = Values(
        poses=[sf.Pose2.identity()] * num_poses,
        landmarks=[sf.V2(-2, 2), sf.V2(1, -3), sf.V2(5, 2)],
        distances=[1.7, 1.4],
        angles=np.deg2rad([[55, 245, -35], [95, 220, -20], [125, 220, -20]]).tolist(),
        epsilon=sf.numeric_epsilon,
    )

    return initial_values, num_poses, num_landmarks
```

```python
def bearing_residual(
    pose: sf.Pose2, landmark: sf.V2, angle: sf.Scalar, epsilon: sf.Scalar
) -> sf.V1:

    t_body = pose.inverse() * landmark
    predicted_angle = sf.atan2(t_body[1], t_body[0], epsilon=epsilon)
    return sf.V1(sf.wrap_angle(predicted_angle - angle))
```

RESIDU 2D

```python
def odometry_residual(
    pose_a: sf.Pose2, pose_b: sf.Pose2, dist: sf.Scalar, epsilon: sf.Scalar
) -> sf.V1:

    return sf.V1((pose_b.t - pose_a.t).norm(epsilon=epsilon) - dist)


from symforce.opt.factor import Factor
```

JARAK RESIDU 2D

STUDI CASE MASALAH

```python
def build_factors(num_poses: int, num_landmarks: int) -> T.Iterator[Factor]:

    for i in range(num_poses - 1):
        yield Factor(
            residual=odometry_residual,
            keys=[f"poses[{i}]", f"poses[{i + 1}]", f"distances[{i}]", "epsilon"],
        )

    for i in range(num_poses):
        for j in range(num_landmarks):
            yield Factor(
                residual=bearing_residual,
                keys=[f"poses[{i}]", f"landmarks[{j}]", f"angles[{i}][{j}]", "epsilon"],
            )

```

MEMBUAT INISIAL VALUE DARI MASALAH, DAN MEMBUAT FACTOR DALAM FUNGSI MAIN YANG AKAN DIJALANKAN

```python
def main() -> None:
    initial_values, num_poses, num_landmarks = build_initial_values()

    factors = build_factors(num_poses=num_poses, num_landmarks=num_landmarks)

    optimized_keys = [f"poses[{i}]" for i in range(num_poses)]

    optimizer = Optimizer(
        factors=factors,
        optimized_keys=optimized_keys,
        debug_stats=True,  # Return problem stats for every iteration
        params=Optimizer.Params(verbose=True),  # Customize optimizer behavior
    )

    result = optimizer.optimize(initial_values)
```

MENAMPILKAN HASIL DARI TOTAL ITERASI YANG DILAKUKAN

```python
print(f"Num iterations: {len(result.iteration_stats) - 1}")
print(f"Final error: {result.error():.6f}")

for i, pose in enumerate(result.optimized_values["poses"]):
    print(f"Pose {i}: t = {pose.position()}, heading = {pose.rotation().to_tangent()[0]}")

from symforce.examples.robot_2d_localization.plotting import plot_solution

plot_solution(optimizer, result)
```

OUTPUT TAMPILAN ROBOT BERGERAK MELALUI BIDANG 2D

```
if __name__ == "__main__":
    main()
```

```
Num iterations: 8
Final error: 0.000220
Pose 0: t = [[-0.58303818]
 [-0.82449079]], heading = [1.073486]
Pose 1: t = [[ 1.01671023]
 [-0.23835618]], heading = [0.85760621]
Pose 2: t = [[1.79784992]
 [0.92055145]], heading = [0.67637098]
```