

MELLÉKLETEK

1.számú Melléklet: Kontroller paraméter fájl

```
Open  ▾  [icon]  *logitech.yaml
~/catkin_ws/src/magni_robot/magni_teleop/param

1 axis_linear: 1 # Left thumb stick vertical
2 scale_linear: 0.5
3 scale_linear_turbo: 1.0
4 axis_angular: 0 # Left thumb stick horizontal
5 scale_angular: 1.0
6
7 enable_button: 4 # Left top trigger
8 enable_turbo_button: 6 # left bottom trigger
```

2.számú Melléklet: A start_sensors.launch fájl

```
Open start_sensors.launch
~/catkin_ws/src/measure_with_magni/launch

1 <!-- -->
2 <launch>
3   <!-- Xsens IMU Node -->
4   <node name="xsens_mti_node" pkg="xsens_mti_driver" type="xsens_mti_node" output="screen">
5     <param name="device" value="/dev/ttyUSB1" />
6     <param name="baudrate" value="921600" />
7     <param name="pub_acceleration" value="True" />
8     <param name="pub_angular_velocity" value="True" />
9     <param name="pub_free_acceleration" value="True" />
10    <param name="pub_imu" value="True" />
11    <param name="pub_mag" value="True" />
12    <param name="pub_quaternion" value="True" />
13    <param name="pub_sampletime" value="True" />
14    <param name="pub_temperature" value="True" />
15    <param name="pub_transform" value="False" />
16    <param name="pub_twist" value="False" />
17    <param name="pub_velocity" value="False" />
18    <param name="pub_gnss" value="False" />
19    <param name="pub_positionLLA" value="False" />
20    <param name="pub_pressure" value="False" />
21    <param name="publisher_queue_size" value="5" />
22  </node>
23
24  <!-- ROS Serial Node for IMU Array -->
25  <node name="rosterial_node" pkg="rosterial_python" type="serial_node.py" output="screen">
26    <param name="port" value="/dev/ttyUSB0" />
27    <param name="baud" value="2000000" />
28  </node>
29
30
31
32  <!-- Ublox GPS Node -->
33  <node name="ublox_gps_node" pkg="ublox_gps" type="ublox_gps" output="screen">
34    <rosparam command="load" file="$(find ublox_gps)/config/c94_m8p_rover.yaml" />
35  </node>
36
37 </launch>
38
```

3.számú Melléklet: A main_imu.py fájl

```
Open  ▾  [ ]  *main_imu.py
~/catkin_ws/src/measure_ws/src/main_imu.py

1#!/usr/bin/env python3
2
3import rospy
4import csv
5import math
6from sensor_msgs.msg import Imu
7
8# Globális változók
9nth_pos = 0 # Inicializáljuk a számlálót
10has_run = False
11
12def header_function():
13    global has_run
14    if not has_run:
15        header = ["IMU", "nth_pos", "acc.x", "acc.y", "acc.z", "gyr.x", "gyr.y", "gyr.z", "mag.x", "mag.y", "mag.z", "mag.norm", "nsecs", "secs", "meas_ON"]
16        for i in range(1, 10):
17            with open(f"/home/inre/Desktop/Imuerek/imu{i}_readings.csv", "a", newline='') as csvfile:
18                writer = csv.writer(csvfile)
19                writer.writerow(header)
20            has_run = True
21
22def meas_callback(msg):
23    global nth_pos
24    imu_meas = 1
25    imu_id = int(msg.orientation.w)
26
27    if imu_id in range(1, 10):
28        mag_norm = math.sqrt(pow(msg.orientation.x, 2) + pow(msg.orientation.y, 2) + pow(msg.orientation.z, 2))
29        imu_data = [imu_id, nth_pos, msg.linear_acceleration.x, msg.linear_acceleration.y, msg.linear_acceleration.z,
30                    msg.angular_velocity.x, msg.angular_velocity.y, msg.angular_velocity.z,
31                    msg.orientation.x, msg.orientation.y, msg.orientation.z, mag_norm,
32                    msg.header.stamp.nsecs, msg.header.stamp.secs, imu_meas]
33        with open(f"/home/inre/Desktop/Imuerek/imu{imu_id}_readings.csv", "a", newline='') as csvfile:
34            writer = csv.writer(csvfile)
35            writer.writerow(imu_data)
36
37    nth_pos += 1 # Számláló növelése
38
39if __name__ == '__main__':
40    rospy.init_node('main_imu_node')
41    header_function() # A header_function hívása, hogy a fejléc létrejöhessen
42    imu_sub = rospy.Subscriber('imu', Imu, meas_callback)
43    rospy.spin()
44
```

4.számú Melléklet: A convert_struct_to_bag.m fájl

```
% ROS bag fájlok betöltése
robot_bag_path = '../magnimeresek/macskako5_repulogps/robot_topics_20241023_164015.bag';
laptop_bag_path = '../magnimeresek/macskako5_repulogps/laptop_topics.bag';

robot_bag = rosbag(robot_bag_path);
laptop_bag = rosbag(laptop_bag_path);

% mappa genhez név kimentés és ell. hogy van-e ilyen mappa
[~, measurement_name, ~] = fileparts(fileparts(robot_bag_path));

save_folder = fullfile('datas', measurement_name);

if ~exist(save_folder, 'dir')
    mkdir(save_folder);
end

% /cmd_vel topic
cmd_vel_topic = select(robot_bag, 'Topic', '/cmd_vel');
cmd_vel_msgs = readMessages(cmd_vel_topic, 'DataFormat', 'struct');

% /imu/data topic üzeneteinek
imu_data_topic = select(laptop_bag, 'Topic', '/imu/data');
imu_data_msgs = readMessages(imu_data_topic, 'DataFormat', 'struct');

% /imu/mag
imu_mag_topic = select(laptop_bag, 'Topic', '/imu/mag');
imu_mag_msgs = readMessages(imu_mag_topic, 'DataFormat', 'struct');

% /joint_states
joint_states_topic = select(laptop_bag, 'Topic', '/joint_states');
joint_states_msgs = readMessages(joint_states_topic, 'DataFormat', 'struct');

odom_topic = select(robot_bag, 'Topic', '/odom');
odom_msgs = readMessages(odom_topic, 'DataFormat', 'struct');

% /ublox_gps_node/fix
gps_fix_topic = select(laptop_bag, 'Topic', '/ublox_gps_node/fix');
gps_fix_msgs = readMessages(gps_fix_topic, 'DataFormat', 'struct');

% segedvaltozo
num_msgs = numel(gps_fix_msgs);

% a vegleges struktura krealasa
gps_data(num_msgs) = struct('Sec', [], 'Nsec', [], 'Latitude', [], 'Longitude', [], 'Altitude', [], 'PositionCovariance', []);

% es a tablazat kitoltesa
for i = 1:num_msgs
    gps_data(i).Sec = gps_fix_msgs{i}.Header.Stamp.Sec;
    gps_data(i).Nsec = gps_fix_msgs{i}.Header.Stamp.Nsec;
    gps_data(i).Latitude = gps_fix_msgs{i}.Latitude;
    gps_data(i).Longitude = gps_fix_msgs{i}.Longitude;
    gps_data(i).Altitude = gps_fix_msgs{i}.Altitude;
    gps_data(i).PositionCovariance = gps_fix_msgs{i}.PositionCovariance;
end
```

```

% Xsens imu_data topic feldolgozás
num_imu_msgs = numel(imu_data_msgs);

xsens_imu_data(num_imu_msgs) = struct('Sec', [], 'Nsec', [], 'Orientation', [], 'OrientationCovariance', [], ...
    'AngularVelocity', [], 'AngularVelocityCovariance', [], ...
    'LinearAcceleration', [], 'LinearAccelerationCovariance', []);

for i = 1:num_imu_msgs
    xsens_imu_data(i).Sec = imu_data_msgs{i}.Header.Stamp.Sec;
    xsens_imu_data(i).Nsec = imu_data_msgs{i}.Header.Stamp.Nsec;

    % Orientation
    xsens_imu_data(i).Orientation.X = imu_data_msgs{i}.Orientation.X;
    xsens_imu_data(i).Orientation.Y = imu_data_msgs{i}.Orientation.Y;
    xsens_imu_data(i).Orientation.Z = imu_data_msgs{i}.Orientation.Z;
    xsens_imu_data(i).Orientation.W = imu_data_msgs{i}.Orientation.W;
    xsens_imu_data(i).OrientationCovariance = imu_data_msgs{i}.OrientationCovariance;

    % Angular Velocity
    xsens_imu_data(i).AngularVelocity.X = imu_data_msgs{i}.AngularVelocity.X;
    xsens_imu_data(i).AngularVelocity.Y = imu_data_msgs{i}.AngularVelocity.Y;
    xsens_imu_data(i).AngularVelocity.Z = imu_data_msgs{i}.AngularVelocity.Z;
    xsens_imu_data(i).AngularVelocityCovariance = imu_data_msgs{i}.AngularVelocityCovariance;

    % Linear Acceleration
    xsens_imu_data(i).LinearAcceleration.X = imu_data_msgs{i}.LinearAcceleration.X;
    xsens_imu_data(i).LinearAcceleration.Y = imu_data_msgs{i}.LinearAcceleration.Y;
    xsens_imu_data(i).LinearAcceleration.Z = imu_data_msgs{i}.LinearAcceleration.Z;
    xsens_imu_data(i).LinearAccelerationCovariance = imu_data_msgs{i}.LinearAccelerationCovariance;
end

% IMU mag data
num_mag_msgs = numel(imu_mag_msgs);

xsens_imu_mag_data(num_mag_msgs) = struct('Sec', [], 'Nsec', [], 'X', [], 'Y', [], 'Z', []);

for i = 1:num_mag_msgs
    xsens_imu_mag_data(i).Sec = imu_mag_msgs{i}.Header.Stamp.Sec;
    xsens_imu_mag_data(i).Nsec = imu_mag_msgs{i}.Header.Stamp.Nsec;
    xsens_imu_mag_data(i).X = imu_mag_msgs{i}.Vector.X;
    xsens_imu_mag_data(i).Y = imu_mag_msgs{i}.Vector.Y;
    xsens_imu_mag_data(i).Z = imu_mag_msgs{i}.Vector.Z;
end

%odom mentés
num_msgs = numel(odom_msgs);

odom_data(num_msgs) = struct('Sec', [], 'Nsec', [], ...
    'Position', struct('X', [], 'Y', [], 'Z', []), ...
    'Orientation', struct('X', [], 'Y', [], 'Z', [], 'W', []), ...
    'PoseCovariance', []);

for i = 1:num_msgs
    odom_data(i).Sec = odom_msgs{i}.Header.Stamp.Sec;
    odom_data(i).Nsec = odom_msgs{i}.Header.Stamp.Nsec;

    % Position
    odom_data(i).Position.X = odom_msgs{i}.Pose.Pose.Position.X;
    odom_data(i).Position.Y = odom_msgs{i}.Pose.Pose.Position.Y;
    odom_data(i).Position.Z = odom_msgs{i}.Pose.Pose.Position.Z;

    % Orientation
    odom_data(i).Orientation.X = odom_msgs{i}.Pose.Pose.Orientation.X;
    odom_data(i).Orientation.Y = odom_msgs{i}.Pose.Pose.Orientation.Y;
    odom_data(i).Orientation.Z = odom_msgs{i}.Pose.Pose.Orientation.Z;
    odom_data(i).Orientation.W = odom_msgs{i}.Pose.Pose.Orientation.W;

    % Pose Covariance
    odom_data(i).PoseCovariance = odom_msgs{i}.Pose.Covariance;
end

```

```

%cmd_vel mentes
num_msgs = numel(cmd_vel_msgs);

cmd_vel_data(num_msgs) = struct('Sec', [], 'Nsec', [], ...
                                'Linear', struct('X', [], 'Y', [], 'Z', []), ...
                                'Angular', struct('X', [], 'Y', [], 'Z', []));

for i = 1:num_msgs

    time = cmd_vel_topic.MessageList.Time(i); % itt máshogy meg az idő kinyerése
    sec = floor(time);
    nsec = round((time - sec) * 1e9);

    cmd_vel_data(i).Sec = sec;
    cmd_vel_data(i).Nsec = nsec;

    % Linear Velocity
    cmd_vel_data(i).Linear.X = cmd_vel_msgs{i}.Linear.X;
    cmd_vel_data(i).Linear.Y = cmd_vel_msgs{i}.Linear.Y;
    cmd_vel_data(i).Linear.Z = cmd_vel_msgs{i}.Linear.Z;

    % Angular Velocity
    cmd_vel_data(i).Angular.X = cmd_vel_msgs{i}.Angular.X;
    cmd_vel_data(i).Angular.Y = cmd_vel_msgs{i}.Angular.Y;
    cmd_vel_data(i).Angular.Z = cmd_vel_msgs{i}.Angular.Z;
end

% Struktúrák mentése .mat fájlba
save(fullfile(save_folder, 'cmd_vel_data.mat'), 'cmd_vel_data');
save(fullfile(save_folder, 'gps_data.mat'), 'gps_data');

save(fullfile(save_folder, 'xsens_imu_data.mat'), 'xsens_imu_data');
save(fullfile(save_folder, 'xsens_imu_mag.mat'), 'xsens_imu_mag_data');

clear;

```

5.számú Melléklet: A mozgásmodell implementációja

```
function [x_pred, P_pred] = motion_model(x, V, omega, dt, P, Q)
    theta = x(3);

    if abs(omega) > 1e-5 % Ha omega nem nulla, akkor köríven halad
        r = V / omega; %

        ICC_x = x(1) - r * sin(theta);
        ICC_y = x(2) + r * cos(theta);

        % Predikált állapot
        x_pred = [
            ICC_x + r * sin(theta + omega * dt);
            ICC_y - r * cos(theta + omega * dt);
            theta + omega * dt
        ];

        % Állapotátmeneti mátrix (F) köríves mozgás esetén
        F = [
            1, 0, r * (cos(theta + omega * dt) - cos(theta));
            0, 1, r * (sin(theta + omega * dt) - sin(theta));
            0, 0, 1
        ];

    else % Ha omega nulla, akkor egyenes mozgás
        % Predikált állapot
        x_pred = [
            x(1) + V * cos(theta) * dt;
            x(2) + V * sin(theta) * dt;
            theta
        ];

        % Állapotátmeneti mátrix (F) egyenes mozgás esetén
        F = [
            1, 0, -V * dt * sin(theta);
            0, 1, V * dt * cos(theta);
            0, 0, 1
        ];

    end

    % Kovariancia előrejelzés
    P_pred = F * P * F' + Q;
end
```

6.számú Melléklet: A megfigyelésmodell implementációja

```
function [x_upd, P_upd] = kalman_update(x_pred, P_pred, z, R)
    H = [1, 0, 0;
         0, 1, 0];
    S = H * P_pred * H' + R;
    K = P_pred * H' / S;
    y = z - H * x_pred;
    x_upd = x_pred + K * y;
    P_upd = (eye(length(x_pred)) - K * H) * P_pred;
end
```

7.számú Melléklet: A Kalman-szűrő implementációja

```
clear;
close all;
clc;
format compact;

%% Adatok betöltése
gps_data_loaded = load('datas\tanya2_1\gps_data.mat');
cmd_vel_data_loaded = load('datas\tanya2_1\cmd_vel_data.mat');
gps_data = gps_data_loaded.gps_data;
cmd_vel_data = cmd_vel_data_loaded.cmd_vel_data;

%% menteshez kell a mppa
data_path = 'datas\varga_foci_2\gps_data.mat';
 [~, folder_name, ~] = fileparts(fileparts(data_path));

%% CMD_VEL adatok kibontása
num_cmd_vel_msgs = numel(cmd_vel_data);
cmd_vel_times_sec = zeros(1, num_cmd_vel_msgs);
linear_velocity = struct('X', cell(1, num_cmd_vel_msgs), 'Y', cell(1, num_cmd_vel_msgs), 'Z', cell(1, num_cmd_vel_msgs));
angular_velocity = struct('X', cell(1, num_cmd_vel_msgs), 'Y', cell(1, num_cmd_vel_msgs), 'Z', cell(1, num_cmd_vel_msgs));

for i = 1:num_cmd_vel_msgs
    cmd_vel_times_sec(i) = double(cmd_vel_data(i).Sec) + double(cmd_vel_data(i).Nsec) * 1e-9;
    linear_velocity(i).X = cmd_vel_data(i).Linear.X;
    angular_velocity(i).Z = cmd_vel_data(i).Angular.Z;
end

%% GPS adatok kibontása (eredeti koordináták)
num_gps_msgs = numel(gps_data);
gps_times_sec = zeros(1, num_gps_msgs);
gps_latitudes = zeros(1, num_gps_msgs);
gps_longitudes = zeros(1, num_gps_msgs);

for i = 1:num_gps_msgs
    gps_times_sec(i) = double(gps_data(i).Sec) + double(gps_data(i).Nsec) * 1e-9;
    gps_latitudes(i) = gps_data(i).Latitude;
    gps_longitudes(i) = gps_data(i).Longitude;
end

%% Ground_truth generálása a GPS adatokból (interpolálva a cmd_vel időpontokra)
ref_latitude = gps_latitudes(1);
ref_longitude = gps_longitudes(1);
meters_per_deg_lat = 111320;
meters_per_deg_lon = meters_per_deg_lat * cosd(ref_latitude);

gps_x = (gps_longitudes - ref_longitude) * meters_per_deg_lon;
gps_y = (gps_latitudes - ref_latitude) * meters_per_deg_lat;

% Az eredeti GPS adatok hossza
num_gps_points = length(gps_latitudes);

% A cmd_vel adatok hossza, ahány adatpontot rá interpolálunk
num_cmd_vel_points = length(cmd_vel_times_sec);

% Interpolációs indexek létrehozása az eredeti GPS indexek és a cmd_vel hossza alapján
gps_indices = linspace(1, num_gps_points, num_cmd_vel_points);

% Interpoláció a GPS adatokból a cmd_vel hosszára
ground_truth_gps_lat = interp1(1:num_gps_points, gps_latitudes, gps_indices, 'linear');
ground_truth_gps_lon = interp1(1:num_gps_points, gps_longitudes, gps_indices, 'linear');
```

```

ref_latitude = gps_latitudes(1);
ref_longitude = gps_longitudes(1);
meters_per_deg_lat = 111320;
meters_per_deg_lon = meters_per_deg_lat * cosd(ref_latitude);
gps_x = (gps_longitudes - ref_longitude) * meters_per_deg_lon;
gps_y = (gps_latitudes - ref_latitude) * meters_per_deg_lat;

num_gps_points = length(gps_latitudes);
num_cmd_vel_points = length(cmd_vel_times_sec);

gps_indices = linspace(1, num_gps_points, num_cmd_vel_points);

ground_truth_gps_lat = interp1(1:num_gps_points, gps_latitudes, gps_indices, 'linear');
ground_truth_gps_lon = interp1(1:num_gps_points, gps_longitudes, gps_indices, 'linear');

```

%% Idők normalizálása

```

first_cmd_vel_time = cmd_vel_times_sec(1);
first_gps_time = gps_times_sec(1);
cmd_vel_times_sec = cmd_vel_times_sec - first_cmd_vel_time;
gps_times_sec = gps_times_sec - first_gps_time;

```

%% Kalman-szűrő init

```

x_est = [0; 0; 0];
P = diag([1, 1, 0.1]);
Q = diag([0.1, 0.1, 0.01]);
R = diag([5, 5]);

x_kalman = zeros(1, num_cmd_vel_msgs);
y_kalman = zeros(1, num_cmd_vel_msgs);
theta_kalman = zeros(1, num_cmd_vel_msgs);

```

%% Kalman-szűrő futtatása

```

gps_idx = 1;
for i = 1:num_cmd_vel_msgs
    if i == 1
        dt = cmd_vel_times_sec(i);
    else
        dt = cmd_vel_times_sec(i) - cmd_vel_times_sec(i-1);
    end

    % priori becs
    [x_pred, P_pred] = motion_model(x_est, linear_velocity(i).X, angular_velocity(i).Z, dt, P, Q);

    % Legközelebbi GPS mérés keresése
    while gps_idx <= num_gps_msgs && gps_times_sec(gps_idx) < cmd_vel_times_sec(i)
        gps_idx = gps_idx + 1;
    end

    % közeli gps adat keresése
    if gps_idx > 1 && gps_idx <= num_gps_msgs
        time_diff = abs(gps_times_sec(gps_idx) - cmd_vel_times_sec(i));
        if time_diff < 0.1
            z = [gps_x(gps_idx); gps_y(gps_idx)];
            [x_est, P] = kalman_update(x_pred, P_pred, z, R);
        else
            x_est = x_pred;
            P = P_pred;
        end
    else
        x_est = x_pred;
        P = P_pred;
    end

    x_kalman(i) = x_est(1);
    y_kalman(i) = x_est(2);
    theta_kalman(i) = x_est(3);
end

```

```
%% kalman szuro kimenet gpshez igazitasa
kalman_lat = y_kalman / meters_per_deg_lat + ref_latitude;
kalman_lon = x_kalman / meters_per_deg_lon + ref_longitude;
```

```
%% CMD_VEL poziciók inicializálás a GPS alapján
```

```
cmd_vel_x(1) = 0;
cmd_vel_y(1) = 0;
theta = atan2(gps_y(10) - gps_y(1), gps_x(10) - gps_x(1)); % Kezdeti orientáció
cmd_vel_est = [cmd_vel_x(1); cmd_vel_y(1); theta]; % Kezdeti állapot

for i = 2:num_cmd_vel_msgs
    dt = cmd_vel_times_sec(i) - cmd_vel_times_sec(i-1);
    [cmd_vel_est, ~] = motion_model(cmd_vel_est, linear_velocity(i).X, angular_velocity(i).Z, dt, P, Q);
    cmd_vel_x(i) = cmd_vel_est(1);
    cmd_vel_y(i) = cmd_vel_est(2);
end

cmd_vel_lat = cmd_vel_y / meters_per_deg_lat + ref_latitude;
cmd_vel_lon = cmd_vel_x / meters_per_deg_lon + ref_longitude;
```

```
%% Adatok mentése
```

```
results_path = fullfile('results', folder_name, 'results.mat');
if ~exist(fullfile('results', folder_name), 'dir')
    mkdir(fullfile('results', folder_name));
end
```

```
%% Vizualizáció műholdképen
```

```
ground_truth_loaded = load('results\tanya2_1\ground_truth_gps.mat');
gps_data_loaded = load('results\tanya2_1\results.mat');
```

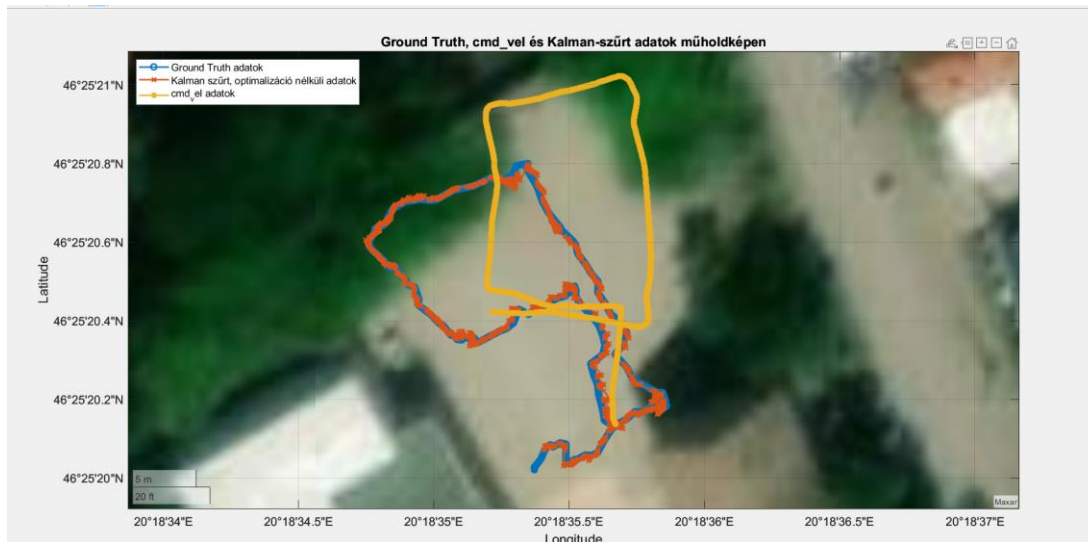
```
% Extract variables from the loaded data
```

```
ground_truth_gps_lat = ground_truth_loaded.ground_truth_gps_lat;
ground_truth_gps_lon = ground_truth_loaded.ground_truth_gps_lon;
kalman_lat = gps_data_loaded.kalman_lat;
kalman_lon = gps_data_loaded.kalman_lon;
```

```
figure;
geoplot(ground_truth_gps_lat, ground_truth_gps_lon, '-o', 'DisplayName', 'Ground Truth adatok', 'LineWidth', 1.5, 'MarkerSize', 6);
hold on;
geoplot(kalman_lat, kalman_lon, '-x', 'DisplayName', 'Kalman szűrt, optimalizáció nélküli adatok', 'LineWidth', 1.5, 'MarkerSize', 6);
geoplot(cmd_vel_lat, cmd_vel_lon, '-*', 'DisplayName', 'cmd_vel adatok', 'LineWidth', 1.5, 'MarkerSize', 6);
geobasemap satellite;
title('Ground Truth, cmd_vel és Kalman-szűrt adatok műholdképen', 'FontSize', 14, 'FontWeight', 'bold', 'Interpreter', 'none');
legend('Location', 'northwest', 'FontSize', 10);
set(gca, 'FontSize', 12);
grid on;
hold off;
```

8.számú Melléklet: A optimalizálatlan Kalman-szűrő eredménye különböző trajektóriákra.

Aszfaltos focipálya:



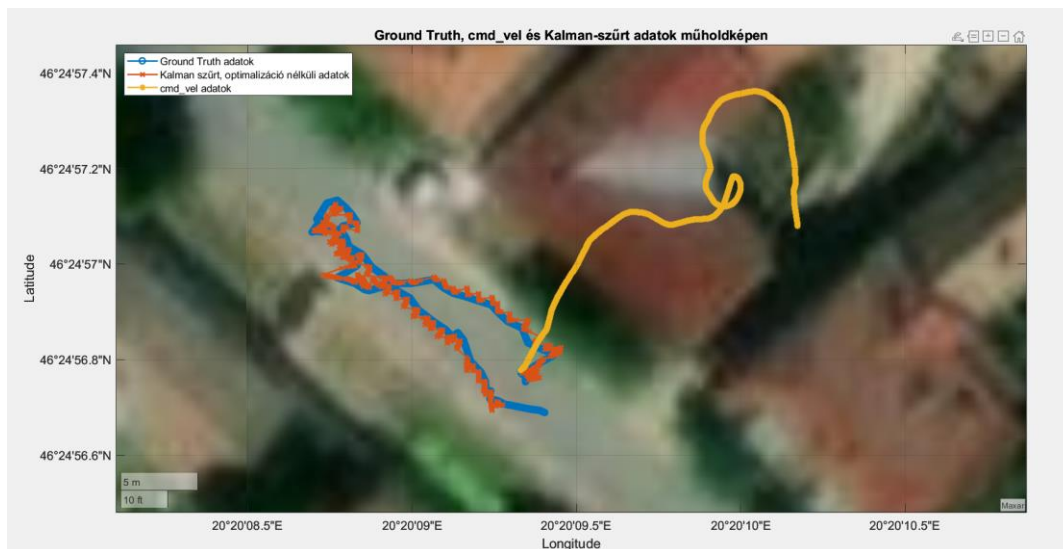
Füves talaj:



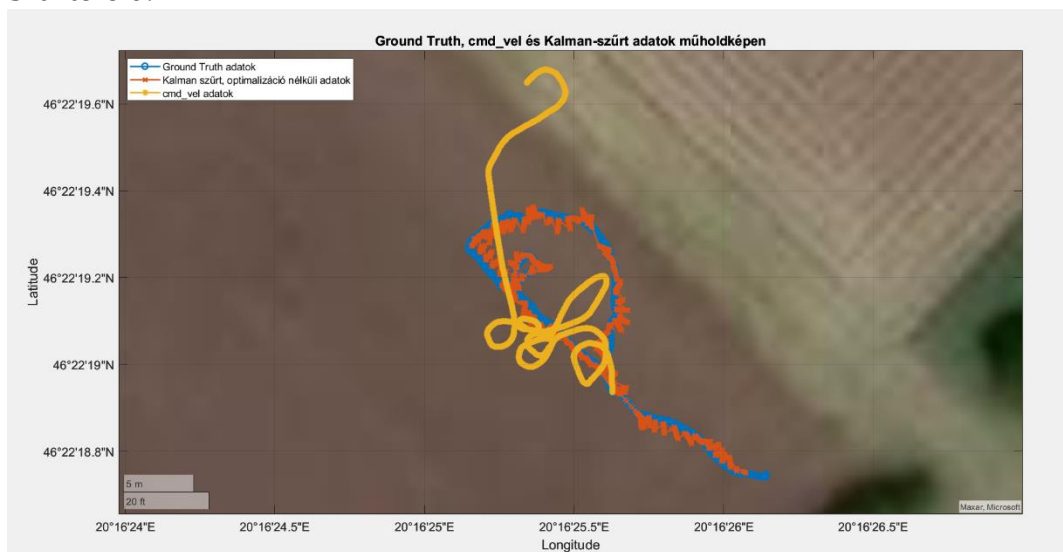
Homokos talaj:



Macskaköves út:



Szántóföld:



9.számú Melléklet: Az optimalizálás forráskódja:

```
function error = kalman_pso_objective(params, gps_data, cmd_vel_data, ...
    interp_lat, interp_lon, ref_latitude, ref_longitude)
    R = diag([params(1), params(2)]);
    Q = diag([params(3), params(4), params(5)]);
    [kalman_lat, kalman_lon] = run_kalman_filter(gps_data, cmd_vel_data, ...
        R, Q, ref_latitude, ref_longitude);
    lat_to_meters = 111320;
    lon_to_meters = 111320 * cosd(ref_latitude);
    error = sqrt(mean(((kalman_lat - interp_lat) * lat_to_meters).^2 ...
        + ((kalman_lon - interp_lon) * lon_to_meters).^2));
end

clear;
clc;
format compact;

% Az alapadatok mappája
data_folder = 'datas';
folders = dir(data_folder);
folders = folders([folders.isdir]); % Csak a mappák kiválasztása

for folder_idx = 1:length(folders)
    folder_name = folders(folder_idx).name;

    % Skip '.' and '..'
    if strcmp(folder_name, '.') || strcmp(folder_name, '..')
        continue;
    end

    % Az aktuális mappa elérési útja
    current_folder = fullfile(data_folder, folder_name);

    % Eredmények mappa és fájlok ellenőrzése
    results_folder = fullfile('results', folder_name);
    optimal_params_file = fullfile(results_folder, 'optimal_params.mat');
    error_improvements_file = fullfile(results_folder, 'error_improvements.mat');
    error_plots_file = fullfile(results_folder, 'error_plots.mat');

    % Ha az összes eredményfájl már létezik, ugorjon a következő mappára
    if isfile(optimal_params_file) && isfile(error_improvements_file) && isfile(error_plots_file)
        disp([folder_name, ' mappán már végrehajtott optimalizálás van, és van hiba file.']);
        continue;
    end
end
```

```

gps_file = fullfile(current_folder, 'gps_data.mat');
cmd_vel_file = fullfile(current_folder, 'cmd_vel_data.mat');
ground_truth_file = fullfile('results', folder_name, 'ground_truth_gps.mat');
gps_tofilter_file = fullfile(current_folder, 'gps_tofilter.mat');

if isfile(gps_file) && isfile(cmd_vel_file) && isfile(ground_truth_file) && isfile(gps_tofilter_file)
    gps_data_loaded = load(gps_file);
    cmd_vel_data_loaded = load(cmd_vel_file);
    ground_truth_data = load(ground_truth_file);
    gps_tofilter = load(gps_tofilter_file);

    % Extract relevant data
    gps_data = gps_data_loaded.gps_data;
    cmd_vel_data = cmd_vel_data_loaded.cmd_vel_data;
    interp_lat = ground_truth_data.ground_truth_gps_lat;
    interp_lon = ground_truth_data.ground_truth_gps_lon;

    ref_latitude = gps_data(1).Latitude;
    ref_longitude = gps_data(1).Longitude;

    % Update GPS data with filtered values
    for i = 1:length(gps_data)
        gps_data(i).Latitude = gps_tofilter.gps_coordinates(i).Latitude;
        gps_data(i).Longitude = gps_tofilter.gps_coordinates(i).Longitude;
    end

    % Original R and Q matrices
    R_original = diag([5, 5]); % Lat, Lon
    Q_original = diag([0.1, 0.1, 0.01]); % X Y Theta

    % PSO Parameters
    num_particles = 50;
    num_iterations = 100;
    lb = [0.01, 0.01, 0.0001, 0.0001, 0.00001];
    ub = [20, 20, 10, 10, 1];

    options = optimoptions('particleswarm', ...
        'SwarmSize', num_particles, ...
        'MaxIterations', num_iterations, ...
        'Display', 'iter', ...
        'InertiaRange', [0.6, 0.9], ...
        'SelfAdjustmentWeight', 1.2, ...
        'SocialAdjustmentWeight', 1.2);

    optimal_params = particleswarm(@(params) kalman_pso_objective(params, gps_data, cmd_vel_data, ...
        interp_lat, interp_lon, ref_latitude, ref_longitude), 5, lb, ub, options);

    % Extract optimized R and Q matrices
    optimal_R = diag([optimal_params(1), optimal_params(2)]);
    optimal_Q = diag([optimal_params(3), optimal_params(4), optimal_params(5)]);

    % Run original Kalman filter
    [orig_kalman_lat, orig_kalman_lon] = run_kalman_filter(gps_data, cmd_vel_data, R_original, ...
        Q_original, ref_latitude, ref_longitude);

    % Run optimized Kalman filter
    [opt_kalman_lat, opt_kalman_lon] = run_kalman_filter(gps_data, cmd_vel_data, optimal_R, optimal_Q, ...
        ref_latitude, ref_longitude);

    lat_to_meters = 111320;
    lon_to_meters = 111320 * cosd(ref_latitude);

    error_orig = sqrt(((orig_kalman_lat - interp_lat) * lat_to_meters).^2 + ((orig_kalman_lon - interp_lon) ...
        * lon_to_meters).^2);
    error_opt = sqrt(((opt_kalman_lat - interp_lat) * lat_to_meters).^2 + ((opt_kalman_lon - interp_lon) ...
        * lon_to_meters).^2);

```

```

% Calculate error metrics
ame_original = mean(error_orig);
rmse_original = sqrt(mean(error_orig.^ 2));
ame_optimized = mean(error_opt);
rmse_optimized = sqrt(mean(error_opt.^ 2));

% Calculate improvements
improvement_ame = 100 * (ame_original - ame_optimized) / ame_original;
improvement_rmse = 100 * (rmse_original - rmse_optimized) / rmse_original;

error_orig_x = (orig_kalman_lat - interp_lat) * lat_to_meters;
error_orig_y = (orig_kalman_lon - interp_lon) * lon_to_meters;
error_opt_x = (opt_kalman_lat - interp_lat) * lat_to_meters;
error_opt_y = (opt_kalman_lon - interp_lon) * lon_to_meters;

if ~exist(results_folder, 'dir')
    mkdir(results_folder);
end

save(optimal_params_file, 'optimal_params', 'R_original', 'Q_original', 'optimal_R', 'optimal_Q', ...
    'orig_kalman_lat', 'orig_kalman_lon', 'opt_kalman_lat', 'opt_kalman_lon');

|
save(error_improvements_file, 'ame_original', 'rmse_original', 'ame_optimized', 'rmse_optimized', ...
    'improvement_ame', 'improvement_rmse');

save(error_plots_file, 'error_orig_x', 'error_orig_y', 'error_opt_x', 'error_opt_y');

fprintf('Results saved for folder: %s\n', folder_name);
else
    disp(['Skipping ', folder_name, ' as it does not contain all required files.']);
end
end
end

```