1. worlds/marker_world.sdf

The environment was modified to shift from a circular layout to a linear coordinate system.

- **Coordinate Re-alignment**: All aruco_box models were moved from their circular positions (e.g., $x=-3, y=2$) to a straight line along the Y-axis at $y=1.0$.
- **Linear Distribution**: Markers were placed at incremental X-distances: $x=1.0, 3.0, 5.0, 7.0,$ and 9.0.
- **Face Orientation**: The yaw rotation for all markers was set to 3.14159 radians (180 degrees). This ensures every marker face is oriented directly toward the robot's starting position at the origin.

2. src/aruco_marker_processor.cpp

The C++ logic was updated to transition from rotational scanning to translational exploration and mapping.

- **Discovery Mechanism**: The SCANNING state now uses linear.x velocity to move the robot along the line instead of spinning in place.
- **Odometry Integration**: A subscription to the /odom topic was added to track the robot's X position during the scan.
- **Spatial Mapping**: The detected_marker_ids set was replaced with a std::map<int, double>. This stores each Marker ID along with the specific X-coordinate where it was first detected.
- **Targeted Navigation**: In the SEARCHING state, the robot no longer rotates blindly; it calculates whether the target marker is ahead or behind its current X position and moves linearly toward that recorded coordinate.

3. launch/aruco_world.launch.py & diff_aruco_world.launch.py

The launch configurations were updated to ensure the simulation starts in a context that matches the linear requirement.

- **Fixed Spawn Point**: The spawnModelNodeGazebo arguments were strictly set to $x=0.0, y=0.0, z=0.1$. This ensures the robot begins at the "start" of the line rather than the center of a circle.
- **Localization Dependency**: The ekf_node was highlighted as critical, as it provides the fused odometry necessary for the robot to move linearly and return to specific coordinates accurately.