Guía 3: Sensores y percepción

En el mundo del RoboSoccer, los sensores juegan un papel fundamental en la percepción y toma de decisiones de los robots. Estos dispositivos electrónicos permiten a los agentes virtuales capturar información del entorno en tiempo real. De esta misma forma, el servidor de simulación de RoboSoccer equipa a los clientes de tres sensores diferentes, además de un sensor especial. Estos tres sensores trabajan en conjunto para proporcionar al agente una perspectiva completa y precisa del entorno en el que se encuentra.

1.1 Sensor Aural

El sensor aural detecta mensajes enviados por el árbitro, entrenadores y otros jugadores. El sensor aural del agente de RoboCup captura mensajes cuando un cliente o un entrenador emiten un comando "say". Además, se reciben mensajes del árbitro en forma de mensajes aurales. Estos mensajes son procesados de forma inmediata.

El formato de los mensajes captados por el sensor aural del servidor de fútbol sigue la siguiente estructura:

```
(hear Time Sender 'Message')
```

Donde:

- Time : Indica el tiempo actual cuando fue enviado el mensaje.
- Sender: Muestra la dirección relativa al remitente si es otro jugador, de lo contrario puede ser uno de los siguientes:
    - self: Cuando el remitente corresponde al propio jugador.
    - referee: Cuando el remitente es el oficial de arbitraje.
    - online_coach_left o online_coach_right: Cuando el remitente es uno de los entrenadores en línea designados.
- Message: Contenido del mensaje.

El servidor cuenta con parámetros predeterminados que influyen en el comportamiento del sensor aural. Específicamente cada jugador puede recibir solamente un mensaje por ciclo, en decir, un mensaje cada 100 [ms]. Y un mensaje transmitido solo puede ser recibido por un jugador que se encuentra a un máximo de 50 metros, teniendo en consideración que la extención del campo es de 105 metros de largo por 68 metros de ancho.

Por otro lado, el comando ejecutado por un agente para transmitir un mensaje es:

```
(say 'Message')
```

1.2 Sensor Visual

El sensor visual recopila información del entorno de juego, como la posición de la pelota y los jugadores, utilizando un ángulo y distancia específicos de visibilidad. En cada ciclo, el servidor actualiza y distribuye esta información visual de forma simultánea a todos los jugadores. Esto garantiza que todos los jugadores tengan acceso a la misma información al mismo tiempo. La información visual se envía desde el servidor en el siguiente formato básico:

```
1 (see ObjName Distance Direction DistChng DirChng BodyDir HeadDir)
```

Donde:

- ObjName: Representa el nombre del objeto visible en el campo de juego. Puede ser el nombre de un jugador, una pelota u otros objetos relevantes.
- Distance: Indica la distancia en metros entre el jugador y el objeto visible.
- Direction: Es el ángulo en grados que muestra la dirección hacia el objeto visible desde la perspectiva del jugador, medido en sentido horario (0 grados representa la dirección hacia el frente del jugador).
- DistChng: Este parámetro indica el cambio de distancia en metros entre el objeto visible y el jugador en el último ciclo. Refleja cómo la distancia ha variado desde la última actualización visual. Si el valor de DistChng es positivo, la distancia entre el jugador y el objeto ha aumentado desde el último ciclo; si es negativo, la distancia ha disminuido.
- DirChng: Este parámetro representa el cambio en la dirección en grados del objeto visible con respecto al jugador en el último ciclo. Indica cómo ha cambiado la posición angular del objeto desde la última actualización visual. Si el valor de DirChng es positivo, la dirección del objeto ha cambiado en sentido horario respecto a la perspectiva del jugador; si es negativo, indica un cambio en sentido antihorario.
- BodyDir: Es el ángulo en grados que indica la dirección en la que el jugador está orientado. El ángulo se mide en sentido horario, donde 0 grados indica que el jugador está orientado hacia el frente.
- HeadDir: Indica el ángulo en grados de la orientación de la cabeza del jugador. El ángulo se mide en sentido horario, donde 0 grados indica que la cabeza mira hacia el frente.

La Figura 1 muestra el arco de área visual de un cliente dentro del campo de juego. Como es posible apreciar el rango visual de un jugador es de 90°.

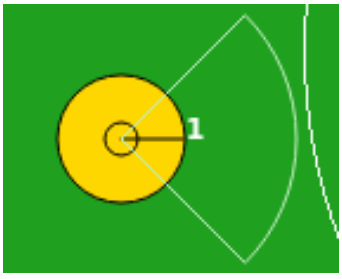

Figura 1: Área visual del jugador.

1.3 Sensor corporal

El sensor corporal detecta el estado físico actual del jugador, incluyendo su resistencia, velocidad y ángulo del cuello. La información se envía automáticamente al jugador en intervalos regulares, con una frecuencia determinada por el parámetro "sense_body_step". En la configuración predeterminada, esta información se actualiza cada 100 milisegundos. Sin embargo, este valor puede ser modificado al ajustar la magnitud de dicho parámetro en el archivo “server.conf”.

Este sensor entrega las datos corporales de cada cliente según el siguiente formato:

```
(sense_body Time
(view_mode ViewQuality ViewWidth)
(stamina Stamina Effort Capacity)
(speed AmountOfSpeed DirectionOfSpeed)
(head_angle HeadAngle)
(kick KickCount)
(dash DashCount)
(turn TurnCount)
(say SayCount)
(turn_neck TurnNeckCount)
(catch CatchCount)
(move MoveCount)
(change_view ChangeViewCount)
(arm (movable MovableCycles) (expires ExpireCycles) (count PointtoCount))
(focus (target {none|{l|r} Unum}) (count AttentiontoCount))
(tackle (expires ExpireCycles) (count TackleCount))
(collision {none|[(ball)] [(player)] [(post)]})
(foul (charged FoulCycles) (card {red|yellow|none})))

```

Donde:

- Time: Representa el tiempo transcurrido desde el inicio del juego.
- ViewMode: Indica el modo de visualización del jugador, que puede ser de alta o baja calidad.
- ViewQuality: Representa la calidad de la vista del jugador.
- ViewWidth: Indica el ancho del campo de visión del jugador.
- Stamina: Representa la cantidad de energía o resistencia del jugador.
- Effort: Indica el esfuerzo actual que el jugador está aplicando.
- Capacity: Representa la capacidad máxima de energía del jugador.
- Speed: Indica la magnitud de velocidad del jugador.
- AmountOfSpeed: Representa la cantidad específica de velocidad del jugador.
- DirectionOfSpeed: Indica la dirección en la que el jugador se está moviendo.
- HeadAngle: Representa el ángulo de orientación de la cabeza del jugador.
- Kick: Indica la cantidad de veces que el jugador ha realizado un kick.
- Dash: Representa la cantidad de veces que el jugador ha realizado un dash.
- Turn: Indica la cantidad de veces que el jugador ha realizado un turn.
- Say: Representa la cantidad de veces que el jugador ha emitido un mensaje.
- TurnNeck: Indica la cantidad de veces que el jugador ha girado el cuello.
- Catch: Representa la cantidad de veces que el jugador ha realizado una atrapada.
- Move: Indica la cantidad de veces que el jugador ha realizado un movimiento.

- **ChangeView:** Representa la cantidad de veces que el jugador ha cambiado la vista.
- **Arm:** Representa el estado del brazo del jugador, incluyendo si es móvil, cuántos ciclos le quedan para expirar y la cantidad de puntos.
- **Focus:** Indica el enfoque del jugador, incluyendo si tiene un objetivo (izquierdo, derecho o ninguno) y la cantidad de atención que ha dedicado.
- **Tackle:** Representa el estado del tackle del jugador, incluyendo cuántos ciclos le quedan para expirar y la cantidad de tackles.
- **Collision:** Indica si el jugador está en colisión con la pelota, otro jugador o un poste.
- **Foul:** Representa el estado de falta del jugador, incluyendo cuántos ciclos de falta ha acumulado y si tiene una tarjeta (roja, amarilla o ninguna).

Existen una multitud de factores que afectan el estado de un jugador. La Figura 2 muestra un cliente cuya estamina se encuentra en un estado crítico.

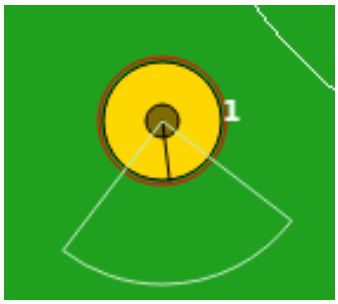

Figura 2: Jugador sin suficiente estamina.

2 Actividad 1:

Duración aproximada: 40 minutos.

La finalidad de esta actividad consiste en desarrollar un algoritmo que permita al jugador de RoboSoccer procesar la información visual y corporal entregada por los sensores. El objetivo es que el jugador pueda detectar la posición del balón y navegar hacia él para luego patearlo hacia la portería. A través de esta actividad, se busca mejorar las habilidades de toma de decisiones y el rendimiento estratégico del jugador.

Los objetivos de esta actividad son:

- Aplicar los conocimientos adquiridos sobre sensores de percepción del simulador.
- Desarrollar un algoritmo de toma de decisiones que aplique la información entregada por los sensores.

En primer lugar es necesario que los participantes se familiaricen con el formato de la información entregada por el servidor. Para ello se recomienda utilizar el cliente de demostración aplicado durante la Guía 1 de este taller, utilizando específicamente la opción de comando, ./rcssclient > info-sensores.txt ya que esta opción enviara toda la información proporcionada por los sensores a un archivo de texto llamado info-sensores.txt. Luego, se debe procesar la información visual y corporal diseñando un algoritmo que

permita al jugador moverse de forma eficiente hacia el balón. Posteriormente los participantes deben alterar el algoritmo creado para que el jugador sea capaz de patear el balón hacia la portería hasta ser capaz de anotar un gol.

Al completar esta actividad, se espera haber logrado los siguientes resultados:

- El algoritmo de cliente detecta la posición del balón y navega hacia él en menos de un minuto.
- El jugador patea el balón hacia la portería.
- De una cantidad de 5 intentos el jugador logra anotar en por lo menos 2 ocasiones.

La siguiente figura muestra los resultados esperados.

Figura 3: Agente jugador justo antes de lograr una anotación.

Esta actividad nos brinda la oportunidad de explorar la importancia de la percepción y toma de decisiones. Al desarrollar un algoritmo efectivo para detectar y navegar hacia el balón, así como para patearlo hacia la portería, los participantes obtienen las habilidades básicas para crear algoritmos capaces de interactuar con su entorno.

3 Actividad 2:

Duración Aproximada: 20 minutos

Esta actividad permite a los participantes adquirir habilidades clave en la programación y coordinación de equipos de RoboSoccer. En ella se fomentan las estrategias en conjunto, mejorando la comunicación y colaboración al aplicar el trabajo en equipo.

Los participantes del taller deberán formar parejas con el fin de crear un algoritmo capaz de coordinar el movimiento de dos robots, para que en conjunto sean capaces de marcar una anotación.

Los objetivos de esta actividad son:

- Formar los conocimientos básicos para la creación de un equipo.
- Mejorar la comunicación y colaboración entre los participantes al trabajar en parejas.
- Potenciar el trabajo en equipo y la toma de decisiones conjuntas para alcanzar un objetivo común.

Los integrantes del taller se dividirán en parejas. Luego de formar los equipos, cada pareja deberá discutir la manera de fusionar los algoritmos de movimiento creados en las actividades anteriores. Finalmente, usando los comandos de comunicación del servidor y la información entregada por el sensor Aural, los participantes deben modificar sus algoritmos para conseguir que sus jugadores sean capaces de enviar y recibir mensajes, con el fin de coordinar sus movimientos y marcar una anotación.

Al completar esta actividad, se espera haber logrado los siguientes resultados:

- Coordinación efectiva entre los dos robots logrando un movimiento fluido hasta la portería.
- Marcar una anotación utilizando a ambos jugadores.

Las siguientes imágenes ilustran los resultados esperados:

Figura 4: Dos jugadores ofensivos iniciados en el campo de juego.

Figura 5: Jugadores utilizando movimientos coordinados con el fin de marcar una anotación.

La actividad de coordinación de movimiento de dos robots en el juego de RoboSoccer ofrece a los participantes la oportunidad de aplicar sus habilidades de trabajo en equipo para lograr un objetivo común. A través de esta actividad, se promueve el desarrollo de la coordinación, la comunicación y la toma de decisiones en un entorno desafiante.

Referencias

[1] Akiyama, H. (s.f.). *HELIOS Base: A base team for the RoboCup Soccer Simulation*. Recuperado el 28 de Junio de 2023, de https://github.com/helios-base

[2] Akiyama, H., & Nakashima, T. (2014). HELIOS Base: An Open Source Package for the RoboCup Soccer 2D Simulation. *RoboCup2013: Robot World XVII, Lecture Notes in Artificial Intelligence.* Berling: In Sven Behnke, Manuela Veloso, Arnoud Visser, and Rong Xiong editors.

[3] RoboCup Federation. (s.f.). *RoboCup Federation official website*. (RoboCup Federation) Recuperado el 2 de Julio de 2023, de https://robocup.org

[4] The RoboCup Soccer Simulator Maintenance Committee. (s.f.). *The robocup soccer simulator users manual*. Recuperado el 02 de Julio de 2023, de https://rcsoccersim.readthedocs.io