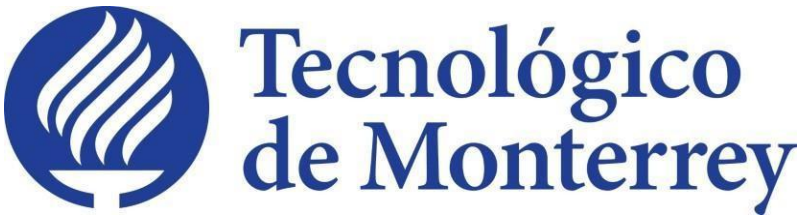


Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Guadalajara

Modelación de sistemas multiagentes con sistemas computacionales (Gpo 102)

Evidencia 2 - Entrega Final

Jose Manuel Martinez Morales-A01734279
Sebastian Gerritsen Ortiz-A01643364
Rodrigo Castellanos Rodríguez-A01643147
Antoine Ganem Nuñez-A01644024
Nathan Sylvain Nicolas Ramard

Profesor

Iván Axel Dounce Nava

Javier Félix Rendón

Fecha de entrega

6 de septiembre de 2024

1.- Descripción Detallada de Funciones de los Agentes

1.1 DronAgent

- **setup()**: Establece la ubicación inicial del dron a (0, 0) y prepara una lista para almacenar puntos de interés. También establece una variable para referenciar al guardia que controlará el dron cuando sea necesario.
- **recibir_puntos_interes(puntos)**: Recibe una lista de puntos de interés (coordenadas) de Unity. Estos puntos representan áreas específicas que el dron deberá monitorear o hacia las cuales debe moverse durante su patrullaje.
- **seleccionar_punto_aleatorio()**: Selecciona un punto de interés aleatoriamente de la lista almacenada. Devuelve el índice del punto seleccionado para que Unity mueva el dron hacia esa ubicación. Ayuda a simular patrones de patrullaje dinámicos y adaptativos.
- **mandar_alerta_guardia()**: Envía una alerta al guardia asignado cuando se detecta una situación que requiere intervención, activando el método recibir_mensaje del guardia para una respuesta coordinada.
- **recibir_mensaje(instrucciones)**:Recibe instrucciones de acción del guardia, las procesa y, si es necesario, activa una respuesta o mantiene la instrucción en espera para ejecución futura.

1.2 GuardiaAgent

- **setup()**: Inicializa el buzón de mensajes, que almacena estados de alertas específicas y referencias a otros agentes como el dron y la cámara.
- **recibir_mensaje(mensaje)**:Procesa mensajes recibidos del dron o la cámara, actualizando el estado del buzón y activando acciones apropiadas como controlar el dron.
- **mandar_instrucciones_dron()**:Envía un conjunto de instrucciones al dron para que realice acciones específicas, como alinearse con un objetivo o manejar una amenaza.
- **limpiar_buzon()**:Restablece el buzón de mensajes a su estado inicial, eliminando todas las alertas procesadas o pendientes.

1.3 CamaraAgent

- **procesar_detecciones_yolo(image)**: Utiliza la red YOLOv4 para procesar una imagen y detectar posibles intrusos o trabajadores. Retorna una clasificación (ladrón o trabajador) basada en las detecciones y la confianza del modelo. Esta función es esencial para la precisión del sistema de vigilancia.
- **recibir_imagen_unity(image)**: Recibe una imagen desde Unity para que la cámara procese el video y realice la detección de personas en el área de vigilancia. La integración de la cámara con Unity permite realizar un análisis en tiempo real y generar alertas de seguridad de manera rápida y precisa.
- **mandar_alerta_seguridad():** Envía una alerta de seguridad al guardia cuando detecta actividad sospechosa, utilizando la infraestructura de comunicación establecida para asegurar que la información llegue al guardia y se actúe de manera rápida y efectiva.

1.4 Interacción Entre Agentes

Las funciones descritas facilitan una red de seguridad coordinada en el sitio de construcción:

- **Interacción Dron-Guardia**: El dron actúa como un explorador móvil que, al detectar algo inusual, puede llamar al guardia para una acción directa. La capacidad de respuesta directa del guardia mediante def mandar_instrucciones_dron(self) asegura que las decisiones críticas son tomadas por personal capacitado.
- **Interacción Cámara-Guardia**: Las cámaras funcionan como puntos de vigilancia fijos que continuamente evalúan su campo de visión para detectar riesgos. Al igual que con el dron, las cámaras pueden solicitar la intervención del guardia, optimizando la respuesta a incidentes específicos detectados visualmente.
- **Cadena de Comando y Control**: Tanto las cámaras como el dron están vinculados al guardia, lo que centraliza la gestión de las alertas y las respuestas, proporcionando un mecanismo de respuesta integrado y eficiente.

2.- Ontología

1.1 Clases Principales:

1. **Entidad**:

- **Descripción**: Clase base para todos los objetos y agentes dentro del sistema. Esta clase es un punto de partida para definir características y propiedades comunes que pueden ser heredadas por clases más específicas.

2. **Persona**:
   - **Descripción**: Representa a los individuos dentro del entorno. Sirve como clase base para diferenciar entre diferentes tipos de personas que interactúan o están presentes en el sitio de construcción.
   - **Subclases**:
     - **Ladrón**: Individuos que representan una amenaza o riesgo de seguridad. Pueden ser el objetivo de detección por parte de los drones y cámaras.
     - **Trabajador**: Personas autorizadas y legítimas que trabajan en el sitio. Es crucial que el sistema pueda diferenciar entre trabajadores y ladrones para minimizar falsas alarmas y optimizar las respuestas de seguridad.
3. **Dron**:
   - **Descripción**: Un agente volador equipado con cámaras y sensores, utilizado para patrullar y monitorear el área de construcción desde el aire. Capaz de enviar alertas y ser controlado remotamente por personal de seguridad.
4. **Cámara**:
   - **Descripción**: Dispositivos de vigilancia estáticos instalados en puntos estratégicos del sitio. Estas cámaras están diseñadas para monitorear continuamente y alertar sobre actividades sospechosas detectadas.
5. **Guardia**:
   - **Descripción**: Representa al personal de seguridad encargado de responder a incidentes y alertas generadas por drones y cámaras. Los guardias pueden tomar control directo del dron para investigaciones detalladas o intervenciones en tiempo real.

1.2 Propiedades de la Ontología

1. **tiene_ubicacion**:
   - **Dominio**: Dron, Camara, Guardia.
   - **Rango**: Coordenadas o identificadores de ubicación específicos.

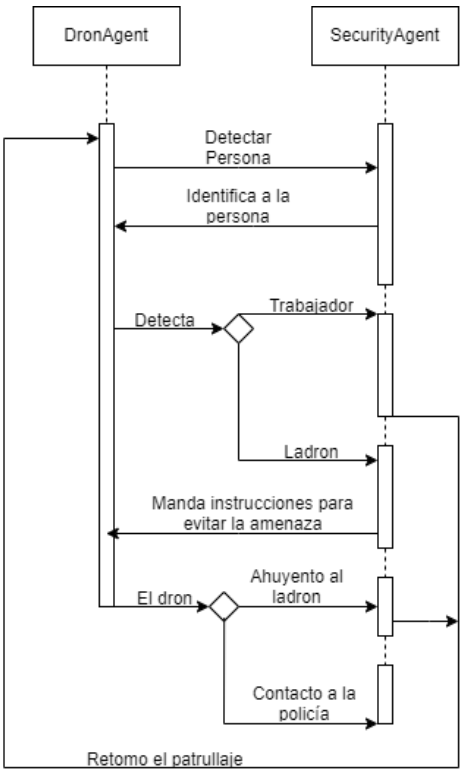
    - **Función**: Permite asociar a cada entidad con una ubicación específica dentro del sitio de construcción, lo que es esencial para la coordinación y respuesta adecuada en situaciones de seguridad.
2. **esta_alertando**:
    - **Dominio**: Dron, Camara.
    - **Rango**: Booleano (verdadero/falso).
    - **Función**: Indica si el dron o la cámara están actualmente en modo de alerta, lo que podría desencadenar una acción por parte del guardia de seguridad.

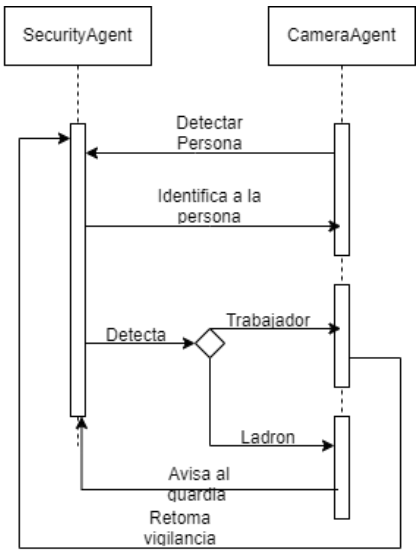
1.3 Uso de la Ontología

- **Gestión de Información**: La ontología permite un manejo estructurado y consistente de la información dentro del sistema, facilitando la interoperabilidad entre diferentes agentes y dispositivos.
- **Interacción y Comunicación**: A través de las propiedades definidas, los agentes pueden comunicar su estado y acciones, permitiendo una respuesta coordinada entre el personal de seguridad, los drones y las cámaras.
- **Adaptabilidad**: La estructura y claridad proporcionadas por la ontología facilitan adaptaciones y expansiones futuras del sistema, como la incorporación de nuevas tecnologías o la expansión del área de vigilancia.

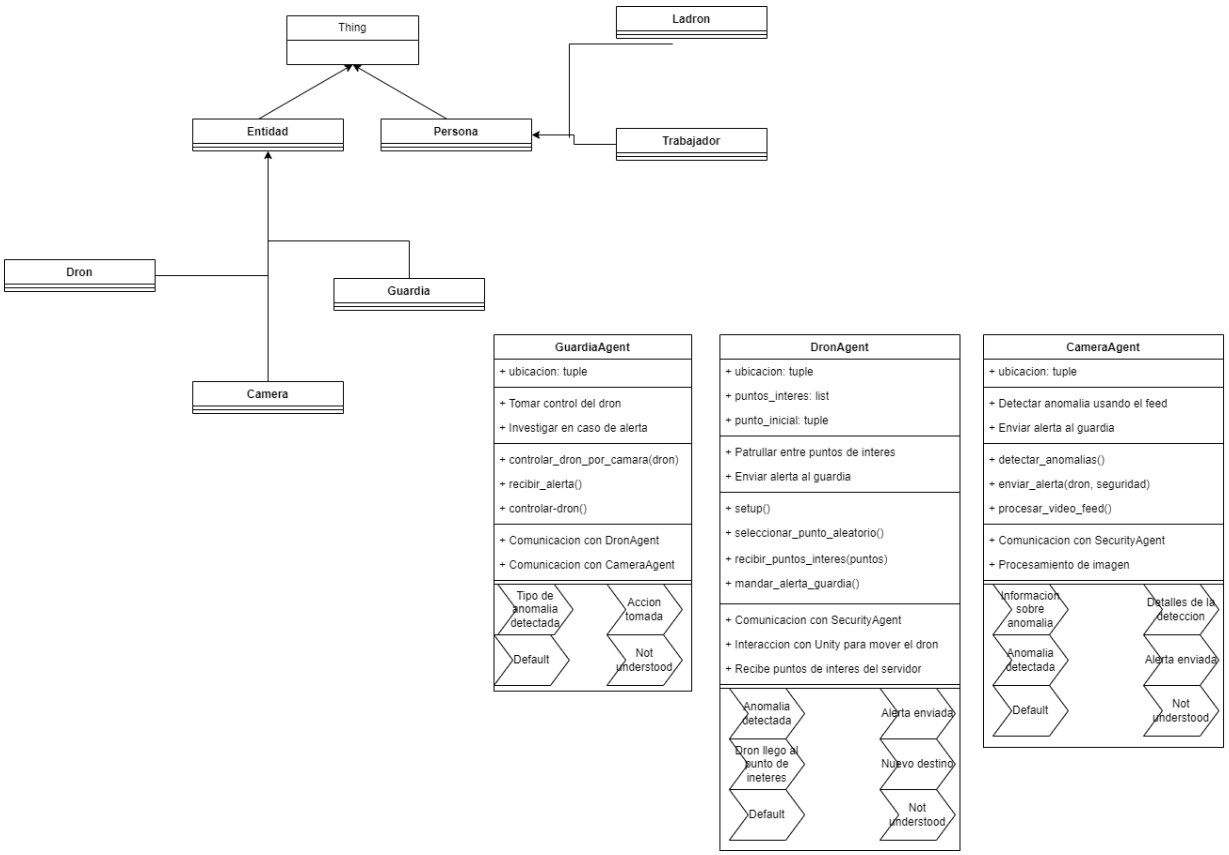
2.- Diagrama de protocolo y clases de agentes

2.1- Diagrama de protocolo de agentes





2.2 Diagrama de clases de agentes y Ontología

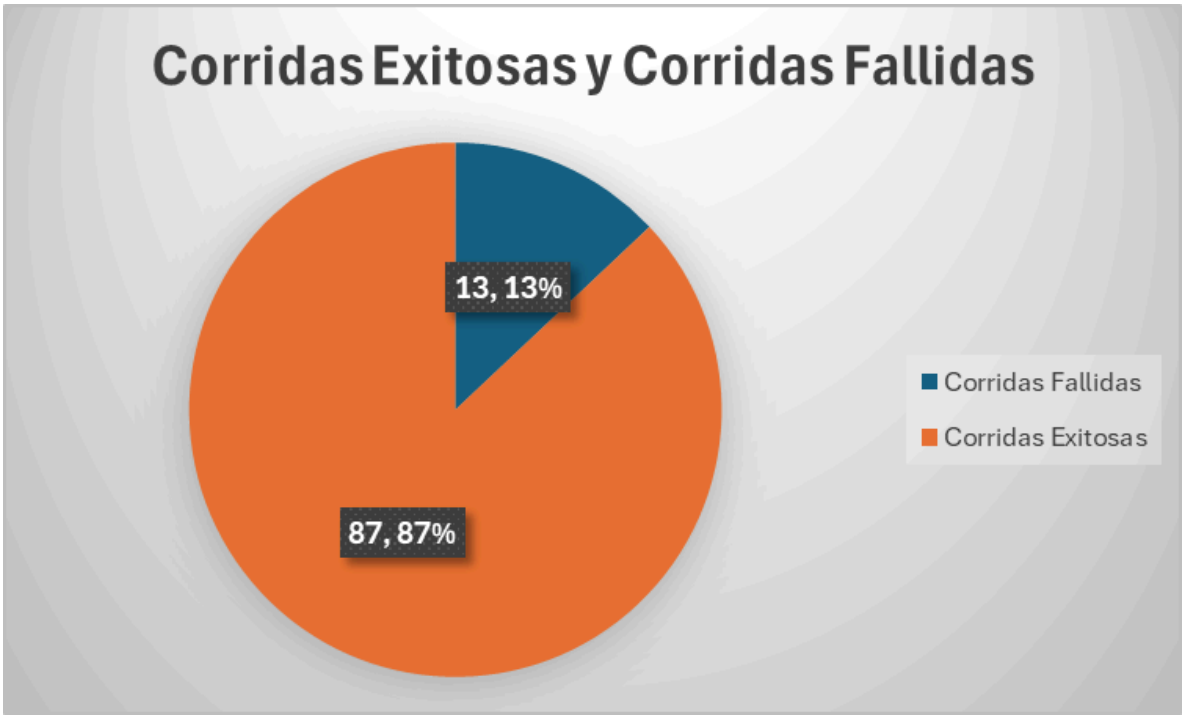


3.- Métrica de utilidad y éxito de cada agente

Primera Métrica: Éxito en Corridas Simuladas

La primera manera en que evaluamos el éxito de nuestros agentes fue mediante la simulación de 100 corridas del sistema. En cada corrida, nuestros agentes (dron, cámara y personal de seguridad) tenían la tarea de detectar y responder a cualquier anomalía en el entorno, como intrusiones o comportamientos sospechosos. Consideramos una corrida exitosa cuando todos los agentes lograron identificar y detener correctamente las anomalías sin dejar pasar ningún evento crítico.

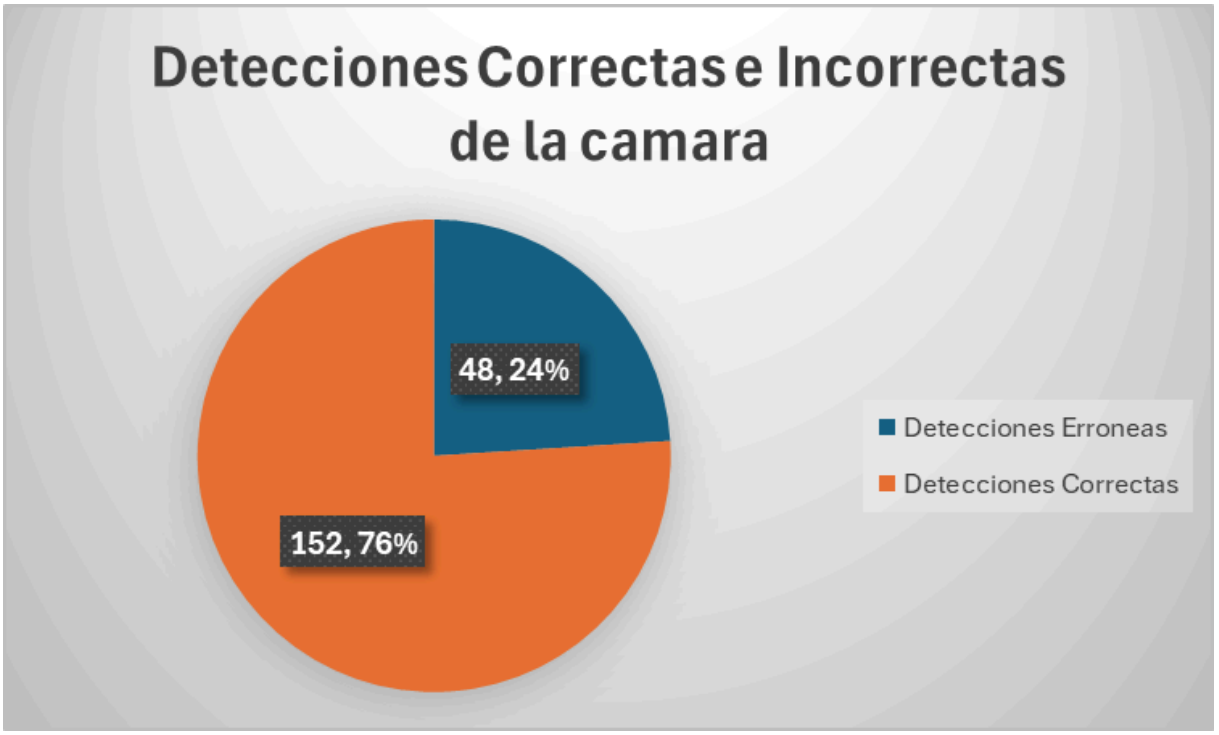
De las 100 corridas que realizamos, nuestros agentes lograron un éxito en 87 de ellas, lo que representa una tasa de éxito del 87%. Este enfoque nos permitió evaluar la capacidad del sistema para funcionar de manera eficiente en diferentes escenarios y asegurarnos de que los agentes fueran capaces de proteger el sitio bajo diversas condiciones.



Segunda Métrica: Precisión en Detección con YOLO

El segundo método que utilizamos para medir la efectividad fue específicamente con la cámara, evaluando su capacidad para distinguir entre trabajadores y posibles intrusos utilizando la red de detección YOLO. Presentamos a la cámara un conjunto de 200 personas en diferentes situaciones, y la tarea de la cámara fue identificar correctamente si cada individuo era un trabajador o un ladrón.

De esas 200 detecciones, la cámara identificó correctamente a 152 personas, lo que da una precisión del 76%. Este análisis nos permitió medir cuán efectiva es la cámara en la identificación precisa de amenazas, lo que es esencial para evitar falsos positivos o fallos en la detección de intrusos reales en el sitio de construcción.



(En la gráfica de arriba debería decir 52%) Ambas métricas nos proporcionan una visión clara de la efectividad de los agentes y nos ayudan a identificar áreas de mejora en la simulación y la toma de decisiones en tiempo real.

4.- Análisis y Reflexión Individual

Jose Manuel Martinez:

Selección del Modelo Multiagentes

Elegimos el modelo multiagentes por su eficiencia en coordinar interacciones complejas entre drones, cámaras y personal de seguridad en un sitio de construcción. Este modelo permite adaptarse rápidamente a situaciones cambiantes y manejar incidentes eficazmente, mejorando los tiempos de respuesta y la seguridad general.

Diseño Gráfico Presentado

El diseño en Unity ofrece una visualización intuitiva del entorno, facilitando la operación y respuesta rápida. Sin embargo, requiere recursos significativos, lo que podría ser limitante en entornos con infraestructura tecnológica menos desarrollada.

Ventajas y Desventajas de la Solución

La solución proporciona monitoreo constante y adaptabilidad a escenarios de seguridad, pero depende fuertemente de la conectividad y tecnología avanzada. Para mitigar estas desventajas, consideramos aumentar la redundancia del sistema y mejorar la gestión local de datos.

Reflexión Sobre el Proceso de Aprendizaje

Este proyecto me enseñó la importancia de la planificación detallada y la adaptabilidad en el diseño de sistemas tecnológicos, reforzando mi habilidad para aplicar teorías de sistemas multiagentes en contextos prácticos y mejorar la eficiencia operacional en entornos complejos.

Sebastian Gerritsen Ortiz:

Selección del Modelo Multiagentes

Se seleccionó un modelo multiagentes debido a la naturaleza de este reto, que nos mostraba la necesidad de una respuesta coordinada y eficiente entre diferentes agentes (el dron, cámaras y el guardia) dentro de nuestro entorno de un lugar de construcción. Este modelo nos permitió que cada agente cumpliera con un rol específico cómo lo es la vigilancia móvil por parte del dron y la vigilancia fija por parte de las cámaras, así como la toma de decisiones centralizada en el guardia. Este enfoque nos permitió descentralizar las tareas de vigilancia optimizando la detección de anomalías.

Las principales variables tomadas en cuenta fueron la eficiencia de detección, la autonomía de los agentes y la necesidad de interacción humana mínima. Estas variables nos permitieron distribuir la responsabilidad entres agentes autónomos mejorando la eficiencia en la vigilancia.La interacción de las variables en la simulación se refleja tanto en cómo los agentes colaboran para mejorar la eficiencia de detección. Esto nos ayudó a reducir la necesidad de intervención humana constante y optimizó el uso de recursos.

Diseño Gráfico Presentado

El diseño gráfico seleccionado para esta SP fue Unity, que nos ofrece una visualización clara entre los agentes y sus tareas, así como nos permite identificar su ubicación en el sitio de

construcción. El único inconveniente podría ser la cantidad de recursos que requiere sobre todo al trabajar con proyectos más elaborados cómo este.

Ventajas y Desventajas de la Solución

Entre las ventajas podemos mencionar que logramos una coordinación eficiente entre agentes autónomos; eso nos ayudó a reducir la intervención humana y nos da alta adaptabilidad del sistema a posibles diferentes condiciones de vigilancia de ser necesario. También la escalabilidad es una ventaja porque para futuros desarrollos cómo la incorporación de más tecnologías de monitoreo o una ubicación más grande sí sería posible.

En cuánto a las desventajas podemos mencionar que el sistema se basa en el correcto funcionamiento de las cámaras, por lo que si las cámaras no son tan eficientes identificando anomalías esto nos podría representar un gran problema tanto en la identificación de anomalías cómo en la posible detección errónea de alguna.

Reflexión Sobre el Proceso de Aprendizaje

El de esta Situación Problema me permitió comprender a profundidad el uso de sistemas multiagentes en entornos dónde estos deben colaborar, también me enseñó la importancia de una buena coordinación entre agentes autónomos y el usuario, y cómo una buena ontología y protocolos de interacción pueden ayudarnos a mejorar la eficiencia de nuestro sistema significativamente. Finalmente me quedo sobre todo con el aprendizaje de la aplicación y uso práctico en la vida real de los agentes y cómo están tan presentes en nuestro día a día.

Rodrigo Castellanos Rodríguez:

¿Por qué seleccionaron el modelo multiagentes utilizado?

Seleccionamos el modelo porque fue el que nos ayudó a implementar el flujo de comunicación entre todos los agentes del sistema de una manera simple. Es capaz de manejar las interacciones que se llevan a cabo cuando entra un intruso al sitio de construcción.

¿Cuáles fueron las variables que se tomaron al momento de tomar la decisión?

El rol específico que debía cumplir cada tipo de agente en nuestro sistema, que escalabilidad iba a tener el proyecto, con control y coordinación como hacer reacciones que simulen situaciones reales, la efectividad necesaria para cumplir con los requerimientos de seguridad.

¿Cuál es la interacción de esas variables con respecto al resultado de la simulación?

Estas variables impactan de manera directa en el resultado de la simulación. Por ejemplo, si las cámaras no detectan correctamente un intruso o si el dron no es capaz de patrullar bien, el sistema puede fallar. Si hay demoras en la transmisión de datos o errores en la interpretación de las alertas, la reacción puede ser más lenta o errónea. También, la capacidad del sistema para adaptarse a nuevas situaciones o modificar su comportamiento en tiempo real es crucial.

¿Por qué seleccionaron el diseño gráfico presentado?

Optamos por usar Unity por ya estar familiarizados con él. Por ejemplo, sería contraproducente usar Unreal Engine porque jamás lo hemos usado. Unity permite hacer visualizaciones que simulen la vida real, con varios tipos de iluminación, texturas, materiales, físicas, etc. Todo de manera muy sencilla. El único problema es que consume mucho RAM y son proyectos muy pesados.

¿Cuáles son las ventajas que encuentras en la solución final presentada?

La ventaja principal es que es un sistema totalmente autónomo. No se necesita de inputs de parte del usuario para que funcione, aunque eso es una funcionalidad extra que agregamos. Existe coordinación entre agentes que permite que se adapten a las situaciones que se presentan.

En cuánto a las desventajas podemos mencionar que el sistema se basa en el correcto funcionamiento de las cámaras, por lo que si las cámaras no son tan eficientes identificando anomalías esto nos podría representar un gran problema tanto en la identificación de anomalías cómo en la posible detección errónea de alguna.

¿Cuáles son las desventajas que existen en la solución presentada?

Pienso que es muy práctico o cuadrado por ponerlo de alguna manera. Me hubiera gustado añadir más pensamiento a los agentes. Y algunos errores en detección que se dan porque no se hace cada frame sino en un intervalo de tiempo puede hacer que algunos ladrones pasen desapercibidos entonces si hay grietas de seguridad en algunos casos.

¿Qué modificaciones podrías hacer para reducir o eliminar las desventajas mencionadas?

Yo quise hacer por ejemplo, que el dron en vez de elegir una zona random para patrullar que llevará un conteo de cuantos ladrones ha capturado en cada zona para tomar una decisión. Si le conviene ir a donde más ha atrapado o a donde menos. Sería agregar esa lógica al agente. De parte de detección usamos YOLO pero no estoy seguro si los otros modelos como OpenAI o SAM se puedan correr de tal manera que no se reduzcan tanto los frames del juego. Lo del intervalo de tiempo en cada acción fue más que nada por eso.

Reflexión Sobre el Proceso de Aprendizaje

Al inicio del curso, no estaba seguro de qué contenidos veríamos, y debo admitir que me tomó un poco por sorpresa. Nunca antes había trabajado con agentes, y descubrí que tienen una complejidad mayor de lo que imaginaba. Sin embargo, me resultó interesante llevar estos conceptos a la práctica. Programar un agente se siente como tratar de replicar el comportamiento de una persona, lo cual representa un gran desafío, ya que hay muchos factores a considerar. En cuanto a Unity, pude reforzar conocimientos que ya tenía, lo cual fue bastante satisfactorio. Por otro lado, nunca había trabajado con visión por computadora, y la primera implementación para la actividad integradora me resultó muy complicada. A pesar de eso, una vez superada esa primera dificultad, logré avanzar sin tantos problemas.

Antoine Ganem Núñez:

¿Por qué seleccionaron el modelo multiagentes utilizado ?

Elegimos un modelo de agentes deductivo , ya que nos pareció lo más práctico y sencillo de implementar. Los agentes deductivo interactúan con otros agentes con reglas que ya están determinadas en sus comportamientos, lo que nos facilitó el tiempo de implementación. Además de que este enfoque es una solución certera para la naturaleza del problema.

¿Cuáles fueron las variables que se tomaron al momento de tomar la decisión?

Las variables que se tomaron en cuenta fueron la cantidad de agentes, los protocolos de comunicación, los modelos para la simulación y los parámetros de la visión computacional.

¿Cuál es la interacción de esas variables con respecto al resultado de la simulación?

El resultado de la simulación está determinado por la interacción entre las diferentes variables, ya que estas dependen unas de otras. La visión computacional se encarga de detectar a los intrusos y de enviar esa información a los agentes. Estos, a su vez, dependen de los modelos de simulación para ser entrenados, y necesitan los datos proporcionados por la visión computacional para identificar a los ladrones y cumplir con sus objetivos.

¿Por qué seleccionaron el diseño gráfico presentado?

Seleccionamos este diseño gráfico para simular un entorno realista, como el de una zona de construcción. Decidimos considerar todas las variables visuales necesarias para recrear la simulación, con el objetivo de demostrar su utilidad en un entorno real.

¿Cuáles son las ventajas que encuentras en la solución final presentada?

- Modelar situaciones complejas
- Automatización de procesos de vigilancia
- Aplicación con el mundo real
- Modelación de entornos reales
- Flexibilidad en la simulación

¿Cuáles son las desventajas que existen en la solución presentada?

- Problemas de precisión en el modelo de visión por el poder computacional
- La ruta de los agentes es muy fácil de evadir
- Demasiado poder computacional para correr la simulación

¿Qué modificaciones podrías hacer para reducir o eliminar las desventajas mencionadas?

Algunas modificaciones que podrías implementar sería la optimización de código y de assets dentro de unity para reducir el consumo de recursos computacionales y optimizar las rutas de los agentes para aumentar la efectividad de detección de enemigos.

Reflexión Sobre el Proceso de Aprendizaje

Al inicio del curso, no sabía qué esperar de la materia, ya que no estaba familiarizado con los sistemas multiagentes. Sin embargo, al finalizar el curso, comprendí cómo llevar conocimientos abstractos del mundo real a un ambiente de multiagentes y realizar simulaciones para modelar el entorno que nos rodea. Aprendí que estos sistemas pueden ser una herramienta poderosa para resolver problemas complejos en nuestra sociedad. Esta experiencia me ha dado una

nueva perspectiva sobre cómo las tecnologías emergentes pueden influir en el futuro y cómo puedo aplicarlas para generar soluciones innovadoras.

Nathan Sylvain Nicolas Ramard:

¿Por qué seleccionaron el modelo multiagentes utilizado?

Se eligió el modelo multiagentes debido a su capacidad para simular interacciones complejas entre múltiples agentes que representan diferentes entidades dentro del sistema. Este enfoque permite una mejor representación de comportamientos emergentes y dinámicas del sistema que no se podrían captar con un modelo tradicional.

¿Cuáles fueron las variables que se tomaron al momento de tomar la decisión?

Las variables clave incluyeron la cantidad de agentes, sus reglas de comportamiento, la topología de la red de interacciones, y los parámetros que afectan sus decisiones, como la percepción del entorno y las recompensas. Estas variables son cruciales para definir cómo los agentes interactúan y cómo se comportan colectivamente.

¿Cuál es la interacción de esas variables con respecto al resultado de la simulación?

Las interacciones entre las variables afectan directamente el rendimiento del sistema. Por ejemplo, un aumento en el número de agentes puede resultar en una mayor complejidad de la simulación, mientras que las reglas de comportamiento influyen en la eficiencia y efectividad de las estrategias adoptadas por los agentes. La topología de la red puede afectar la propagación de información y las decisiones colectivas.

¿Por qué seleccionaron el diseño gráfico presentado?

El diseño gráfico fue seleccionado para facilitar la visualización clara de las interacciones entre los agentes y el entorno. Se optó por gráficos intuitivos que permiten identificar rápidamente los patrones emergentes y las dinámicas de comportamiento, asegurando que la información se comunique de manera efectiva.

¿Cuáles son las ventajas que encuentras en la solución final presentada?

Las ventajas incluyen:

- Capacidad de modelar interacciones complejas.

- Visualización clara y comprensible de los resultados.
- Flexibilidad para ajustar parámetros y observar diferentes escenarios.
- Potencial para identificar patrones emergentes que podrían no ser evidentes en un enfoque tradicional.

¿Cuáles son las desventajas que existen en la solución presentada?

Las desventajas incluyen:

- Complejidad en la calibración de parámetros.
- Posible consumo elevado de recursos computacionales con un gran número de agentes.
- Dificultad en la interpretación de resultados en sistemas altamente dinámicos.

¿Qué modificaciones podrías hacer para reducir o eliminar las desventajas mencionadas?

Para mitigar las desventajas, se podrían implementar las siguientes modificaciones:

- Optimizar el código y los algoritmos para mejorar la eficiencia computacional.
- Utilizar técnicas de reducción de dimensiones para simplificar la visualización y la interpretación de datos.
- Realizar una calibración más rigurosa de los parámetros para reducir la complejidad en la configuración inicial.

Reflexión Sobre el Proceso de Aprendizaje

A lo largo de este proceso, he aprendido a valorar la importancia de la modelización en sistemas complejos y cómo el enfoque multiagentes puede ofrecer una perspectiva única sobre las dinámicas interactivas. He desarrollado habilidades para identificar y ajustar variables críticas, lo que me ha permitido comprender mejor los resultados de las simulaciones. Además, la experiencia en el diseño gráfico me ha ayudado a comunicar hallazgos de manera más efectiva. Este aprendizaje ha enriquecido mi comprensión de cómo los sistemas complejos operan en el mundo real y cómo se pueden modelar para mejorar la toma de decisiones.

Links:

- Github: https://github.com/Rodrigocr04/ConstruccionUnity
- Presentación Canvas: https://www.canva.com/design/DAGQC_NsDPE/T2Gk78vwwh88QfNoKxyntA/edit?utm_content=DAGQC_NsDPE&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton
- Video funcionamiento: https://youtu.be/FohCQ_peNsw?si=43tZk-TkjhwCZPIW

Proceso de instalación:

En el Github se encuentran 2 carpetas. “SinAssets.zip” y “SoloAssets.zip”. El archivo del proyecto era demasiado pesado así que lo tuvimos que subir en 2 carpetas. “SinAssets.zip” contiene todos los archivos y configuraciones de Unity. “SoloAssets.zip” contiene la carpeta de Assets del proyecto de Unity que incluye, scripts, modelos, texturas, etc. Descarguen ambos zip y extraigan los archivos. La carpeta de Assets en “SoloAssets.zip” muevanla a la carpeta de “SinAssets.zip” para tener todo el proyecto junto.

Para correr los códigos es necesario iniciar primero el código de “deteccion_camara.py” y esperar a que imprima el mensaje que está esperando conexión. Después se corre el archivo “agents.py” y se debe esperar a que imprima el mensaje que está esperando conexión. Este código de “agents.py” lo tienes que correr cada que cierres el juego de unity porque al cerrar el juego también se cierra el código. El de “deteccion_camara.py” no se detiene aunque se cierre el juego. Cada código se debe correr en su propia terminal. Para los códigos de Unity se deben asignar varias referencias en el inspector. En el código dron agent.cs los puntos de interés son las esferas en las puertas, los spawn guardia son las esferas de la caja fuerte, y las zonas son esferas se encuentran debajo del piso en las puertas. Para la el código dron detección se debe asignar las cámaras. En algunos códigos en lugar de referencias del inspector se utiliza referencias por tags que hay que agregar. Hay que verificar que los códigos de Unity y Python tengan los mismos servers y ip para los de detección y para los de agentes. El código de “prueba.py” no tiene funcionalidad puede ser ignorado. Camara_stream.cs y captura_imagenes.cs no más se habían usado para tomar SS del juego para poder entrenar el modelo YOLO.

Para la parte de visión computacional en una ventana externa de python se debe correr el código “4_yolo_opencv_detector”, verificar que el nombre de la ventana del editor unity sea

el mismo que el del código. También, comprobar que sean correctos los paths a los archivos .WEIGHTS y .cfg. Hay 1 de cada uno en la carpeta yolo opencv detector. Y hay 1 de cada uno en la misma carpeta de Scripts que el código “4_yolo_opencv_detector”. Esos archivos son lo mismo solo son copias, cualquiera de estos se pueden usar para correr el modelo. Vienen más scripts de jupyter que sirven para entrenar tu propio modelo Yolo en caso de ser necesario. El link de guía para esa parte es el siguiente: https://www.youtube.com/watch?v=RSXgyDf2ALo