UNIVERSIDAD DE EXTREMADURA

ESCUELA DE INGENIERÍAS INDUSTRIALES

SIMULACIÓN PARELELA DE LA EMISIÓN DE WIFI CON TRAZADO DE RAYOS

TRABAJO PRESENTADO PARA OPTAR AL TITULO DE

MÁSTER UNIVERSITARIO EN SIMULACIÓN EN CIENCIAS E INGENIERÍA

POR

DANIEL ALFONSO DURÁN GARCÍA

BADAJOZ, JUNIO DE 2021

TRABAJO FIN DE GRADO/ TRABAJO FIN DE MÁSTER/ PROYECTO FIN DE CARRERA

Autor:

DANIEL ALFONSO DURÁN GARCÍA

Director o Directores:

CARLOS JAVIER GARCÍA ORELLANA

Tribunal de evaluación:

MIGUEL ÁNGEL JARAMILLO MORÁN

JUAN JESÚS RUIZ LORENZO

JOSE LUIS AUSÍN SÁNCHEZ

Índice

1. Introducción y objetivos

Las redes inalámbricas de WiFi se han popularizado en la última década hasta el punto de ser un protocolo barato de usar más a allá de su uso original, con hardware compatible en multitud de dispositivos.

La potencia de su señal, crítica en su uso principal y en otros como el posicionamiento local, es difícil de predecir en escenarios con obstáculos, de forma que aplicaciones que no los tengan en cuenta no podrán garantizar un desempeño correcto.

Para poder conocer esta potencia en las zonas de interés surge la idea de usar la aproximación de las ondas a un rayo que se deplaza con el frente de onda, de modo que es posible conocer su recorrido teniendo en cuenta los posibles rebotes que se puedan producir.

Esta técnica ha sido tradicionalmente usada en el desarrollo y posicionamiento de antenas de telefonía, principalmente para su uso en ciudades. En estos escenarios, los edificios harán las veces de obstáculos de modo que en ciertas zonas la señal puede ser más débil.

La simulaciones con trazado de rayos buscan encontrar este tipo de zonas antes de su implantación, de modo que sea posible corregir las posibles deficiencias y conseguir un posicionamiento óptimo de las antenas usadas.

En este trabajo se plantea su uso en un escenario local. Como se comentaba al inicio, la señal de WiFi encontrará en los escenarios donde se usa obstáculos en forma de paredes o grandes objetos que impidan su avance.

Esto generará efectos de propagación multicamino, donde los rebotes de la señal en las paredes y obstáculos que se encuentren en su camino harán que en ciertos puntos lleguen, además de la señal original, una cierta potencia adicional o bien, en escenarios sin visión directa, la señal recibida será exclusivamente la de estos rebotes.

De forma general, la modelización de estos efectos es imposible basándose únicamente en argumentos geométricos o estadísticos, por lo que será necesario simular la propagación de la señal con un mapa virtual del entorno a estudiar.

2. Fundamento teórico

El uso del trazado de rayos como aproximación a ondas electromagnéticas no es una idea reciente, aunque su alto coste computacional ha hecho que su popularidad haya crecido de forma notable en los últimos años al haber sido posible su uso de forma general.

La principal disciplina en busca de implantaciones de esta técnica es la de gráficos generados computacionalmente, en la que se aplican estas aproximaciones a la luz de modo que se puedan conseguir resultados realistas de una forma más sencilla comparado con la rasterización habitual en la industria.

Encontramos en la bibliografía de este ámbito numerosos capítulos dedicados a la implantación del trazado de rayos que han sido recuperados en este trabajo. Aunque las bases geométricas son similares, las características electromagnéticas de los rayos de luz simplifican parte de los cálculos que debemos corregir en nuestro caso.

2.1. Geometría

2.1.1. Rayos

Comenzamos con la geometría del problema caracterizando los rayos a evaluar. Su composición es muy simple: constan de un punto de origen y una dirección. En su transcurso se encontrarán con las paredes del mapa, contra las que rebotarán para seguir su recorrido en la zona de interés.

Para poder modelizar este comportamiento se considera el rayo como una recta. En este caso el origen O será un punto de esta directa, con su dirección siendo el vector director $\mathbf{d}$ de la misma de modo que sigue la ecuación

$$O + t\mathbf{d} \tag{1}$$

con $t \in \mathbb{R}$.

2.1.2. Paredes

Las paredes serán planos definidos con cuatro puntos como extremos, a partir de los cuáles se calcula su vector normal. Así, los puntos de intersección de las rectas con alguno de estos planos serán los puntos donde los rayos golpearán las paredes, que servirán de origen para los rayos trasmitidos y reflejados que se generen.

Para determinar cuál es este punto partimos de la consideración de que el vector normal del plano –denominado aquí $\hat{\mathbf{n}}$– será perpendicular a cualquier vector contenido en dicho plano, en este caso el definido como diferencia entre el punto de intersección X y el punto donde está definida la normal[1] P de tal forma que su producto escalar es nulo

$$(X - P) \cdot \hat{\mathbf{n}} = 0 \tag{2}$$

X es un punto de la recta, por lo que debe cumplir, para un cierto t_i

$$X = O + t_i\mathbf{d} \tag{3}$$

que es posible incluir en (2) de forma que

$$((O + t_i\mathbf{d}) - P) \cdot \hat{\mathbf{n}} = 0 \tag{4}$$

[1]Esta consideración proviene del caso de una superficie general; en este caso la normal será la misma independientemente de dónde se defina, pudiendo elegir cualquier punto del plano.

que se puede manipular para obtener t_i

$$\begin{aligned} ((O + t_i\mathbf{d}) - P) \cdot \hat{\mathbf{n}} &= 0 \\ (O - P) \cdot \hat{\mathbf{n}} + t_i\mathbf{d} \cdot \hat{\mathbf{n}} &= 0 \\ t_i\mathbf{d} \cdot \hat{\mathbf{n}} &= -(O - P) \cdot \hat{\mathbf{n}} \\ t_i &= -\frac{(O - P) \cdot \hat{\mathbf{n}}}{\mathbf{d} \cdot \hat{\mathbf{n}}} \\ t_i &= \frac{(P - O) \cdot \hat{\mathbf{n}}}{\mathbf{d} \cdot \hat{\mathbf{n}}} \end{aligned} \tag{5}$$

Con este valor es posible ahora usar la Ec. (3) para obtener las coordenadas del punto de impacto, pero no en cualquier caso.

Es necesario tener varias consideraciones a la hora de determinar t_i. La primera de ellas es que es posible obtener un valor negativo: al modelizar el rayo como una recta se abre la posibilidad de encontrar un punto de intersección en la dirección contraria al vector director, por lo que consideraremos que no hay intersección si $t_i < 0$.

Otra de las posibilidades es que la recta sea paralela al plano, de tal forma que no exista un punto de intersección. Si esto ocurre, el producto escalar del denominador de la Ec. (5) tomará un valor nulo, por lo que es necesario evitar la operación.

Es más, es posible optimizar este caso en un grado algo mayor al tener en cuenta que ángulos bajos de la normal del vector y la dirección de la recta también indicarán que la intersección se producirá a una distancia muy lejana, por lo que a efectos prácticos no se producirá –habrá otra pared más cerca–. Así, en el caso de que $\mathbf{d} \cdot \hat{\mathbf{n}} < 10^{-4}$ se interpretará que no hay un punto de intersección con la pared a evaluar.

La última de las consideraciones tiene que ver con los errores de redondeo. A la hora de calcular t_i o las coordenadas del punto de impacto es posible que el punto de intersección obtenido no se encuentre en el plano de incidencia, lo que provoca futuros errores con los rayos reflejados y transmitidos. Para evitarlo, solo se considerará la intersección con los planos si $t_i > 10^{-3}$.

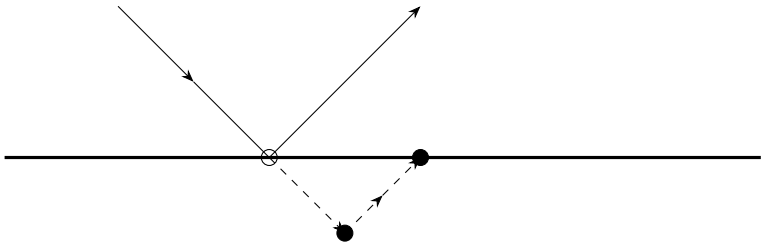


Figura 1: Puntos de intersección en presencia de errores de redondeo: al estar fuera del plano, la evaluación del rayo reflejado encuentra otro punto de intersección en la misma pared. Los círculos huecos y líneas sólidas indican los puntos y rayos correctos; los círculos rellenos y líneas punteadas representan los puntos y rayos calculados erróneamente.

Esta condición puede parecer confusa pero su razón se puede ver en la Figura 1. Debido a errores de redondeo, los rayos reflejados y transmitidos no tienen su origen en el plano de incidencia, por lo que al evaluarlos, encontraremos que la pared con la que impactaría sería, de nuevo, el mismo plano.

Con la condición introducida, no habrá intersecciones estos planos tan cercanos, por lo que el rayo será libre de obviarlo y buscar un rebote en alguna otra pared, como debería haber hecho sin los errores de redondeo.

Aunque en este ejemplo solo se han representado los rayos reflejados, en el caso de que el punto de intersección se encuentre –de nuevo, siguiendo la simetría del ejemplo– delante del plano tendríamos la misma situación al evaluar el rayo transmitido.

Estos nuevos puntos no solo son erróneos –no se deberían producir–, sino que además pueden llegar a producir valores totalmente distorsionados de la potencia de la señal como se explicará en secciones sucesivas.

Una vez determinado el punto de impacto, es necesaria una última comprobación. Al igual que al hablar de las rectas se ponía su manifiesto su extensión infinita, es posible tener la misma consideración con los planos, es decir, será posible encontrar puntos de intersección en cualquier punto del espacio.

Para imponer la presencia de los puntos de intersección entre los límites de la pared es necesario recurrir a sus esquinas –los puntos sobre los que se definen–. Tras determinar el punto de intersección, se comprueba, para cada eje, que su coordenada se encuentra entre los valores máximos y mínimos de las coordenadas de cada eje.

Será necesario añadir un cierto épsilon para, de nuevo, evitar los errores de redondeo, y así evitar que el plano no «sea invisible» al rayo. Por tanto, la comprobación a realizar será, que para cada eje i, las coordenadas estén contenidas en $[\text{mín}_i(c_{ij}) - \varepsilon, \text{máx}_i(c_{ij}) + \varepsilon]$ siendo c_{ij} la coordenada i de la esquina j.

Geometría tras los impactos

Una vez obtenido el punto de impacto del rayo en la pared, queda evaluar los rayos obtenidos a partir de él.

En cada uno de estos impactos se generá un rayo reflejado y un rayo transmitido. El rayo reflejado seguirá una reflexión especular, de modo que su dirección seguirá

$$\mathbf{d}_s = 2(\hat{\mathbf{n}} \cdot \mathbf{d}_i)\hat{\mathbf{n}} - \mathbf{d}_i \tag{6}$$

donde $\mathbf{d}_i$ indica la dirección del rayo incidente y $\mathbf{d}_s$ la del rayo reflejado, ambos vectores normalizados.

La Ec. (6) supone la disposición de las direcciones tal y como se indica en la Figura 2, de tal forma que ambas se indican partiendo desde la pared. Desde la perspectiva del rayo incidente, esta dirección será la inversa a su dirección de incidencia.

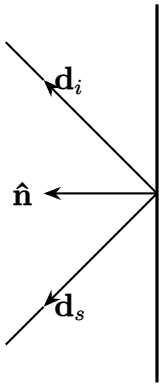


Figura 2: Reflexión especular.

Determinación del ángulo de impacto

Como se explica en siguientes secciones, será necesario evaluar el ángulo de impacto del rayo en la pared para obtener la potencia de los rayos generados.

Este valor se obtiene sin problema teniendo en cuenta que su coseno será el producto escalar de la normal y el vector de la dirección incidente –más concretamente de su inverso, como se puede observar de nuevo en la Figura 2–, pero es necesario hacer una puntualización.

El cálculo de la normal del plano se define a partir de los tres primeros puntos introducidos para cada pared. Por ello, dependiendo del orden en el que se escriban será posible que la normal tome un sentido o su opuesto.

En las ecuaciones (5) y (6) ese cambio de signo es irrelevante, pero no en este caso. Si la normal está definida en la dirección opuesta a la de llegada del rayo, este ángulo será el complementario, como se puede observar en la Figura 3.

(a) Normal en la dirección de llegada del rayo.

(b) Normal en la dirección opuesta a la llegada del rayo.

Figura 3: Dependiendo de la definición de la normal, el ángulo de incidencia puede variar.

Para corregir este comportamiento bastará con tomar el valor absoluto del producto escalar mencionado anteriormente, de modo que su arco coseno esté limitado entre 0 y $\pi/2$ y se obtenga el ángulo correcto.

2.1.3. Receptores

Una vez definidos los rayos y el entorno solo falta la modelización de los receptores de señal.

La interpretación anterior de rayos y paredes hace que solo se tengan en cuenta los puntos de partida e impacto de las rectas, pero no su camino entre ellos. Es este camino el objetivo del problema, pues son los puntos en los que la señal de WiFi es útil.

Para solventarlo se colocan a lo largo del entorno generado una serie de esferas que harán las veces de antenas receptoras. Estas esferas registrarán la potencia de los rayos que impacten contra ellas, de modo que sea posible obtener en los puntos en los que se han colocado la potencia total de señal que una antena colocada en el mismo lugar.

Así, se necesario buscar la intersección de las rectas de las que se compone cada rayo con cada una de estas esferas. Teniendo en cuenta que la ecuación de una esfera de radio r centrada en C es

$$\|X - C\|^2 = r^2 \tag{7}$$

y recuperando la Ec. (3) es posible combinarlas de modo que tengan una intersección para algún t_i

$$\|O + t_i\mathbf{d} - C\|^2 = r^2 \tag{8}$$

Tras manipularla, llegamos a poder obtener t_i a partir de una ecuación cuadrática de modo que

$$t_i = \frac{-2\hat{\mathbf{n}} \cdot (O - C) \pm \sqrt{(2\hat{\mathbf{n}} \cdot (O - C))^2 - 4\|\hat{\mathbf{n}}\|^2(\|O - C\|^2 - r^2)}}{2\|\hat{\mathbf{n}}\|^2} \equiv \hat{\mathbf{n}} \cdot (O - C) \pm \sqrt{\Delta} \tag{9}$$

con

$$\Delta = (2\hat{\mathbf{n}} \cdot (O - C))^2 - 4\|\hat{\mathbf{n}}\|^2(\|O - C\|^2 - r^2) \tag{10}$$

El discriminante Δ de esta solución indicará las características del punto de intersección.

Si toma un valor negativo la recta y la esfera no se encontrarán; si es nulo, ambas soluciones serán idénticas, es decir, la recta toca a la esfera de forma tangencial. Por último,

valores mayores que cero indicarán que el rayo penetra en la esfera, con lo que hay dos puntos de intersección: uno de entrada y otro de salida.

Es únicamente este último caso el que se busca en la simulación, ya que un contacto tangencial no tiene relevancia física para el problema. Por ello, solo se considerará la existencia de un impacto en el caso de que el discriminante de la Ec. (9) sea mayor que cero.

Es posible, por las mismas razones que se explicaban en el caso de los planos, que las soluciones tomen valores negativos. De nuevo, no se considerarán estos casos como impactos de los rayos al no encontrarse en su dirección de propagación.

Se obtienen dos soluciones para t_i. Si ambas son positivas, está claro que la solución en el caso de la suma en la Ec. (9) será siempre mayor que la de la resta, por lo que es en este último caso en el que la distancia de impacto es menor.

Se tomará ese valor –el de la resta– como solución final para determinar el punto de intersección de rayo y esfera, recurriendo de nuevo a la Ec. (3) para obtener sus coordenadas.

2.2. Electromagnetismo

Con las bases de la geometría del problema establecidas, resta abordar la física del problema: la interacción electromagnética de los rayos con el medio.

Como punto de partida es necesario abordar el comportamiento de estos rayos en su camino anterior a cualquier impacto contra alguno de los obstáculos, para lo que es habitual el uso de la ecuación de transmisión de Friis, definida como[**?**]

$$P_t = P_i \left(\frac{\lambda}{4\pi r} \right)^2 \tag{11}$$

donde P_t indica la potencia de una señal de longitud de onda λ a una distancia r, siendo emitida en su origen por una potencia P_i.

Esta ecuación es una aproximación que no siempre es aplicable. Su uso se limita a la modelización de la señal emitida y captada por antenas, en las que se omite cualquier efecto de reflexión entre ellas.

Para poder hacer esta consideración, se establece el límite

$$r > \frac{2D^2}{\lambda} \tag{12}$$

como rangos válidos, con D siendo el tamaño característico de las antenas –en el caso de que sean distintos, se considera el mayor de ellos–.

Teniendo en cuenta que en este caso se usarán ondas de WiFi de 2,4GHz, su longitud de onda será $\lambda \approx 0{,}12$m. Las antenas a usar tienen un tamaño de entre 10 y 20 cm, así que la Ec. (11) será valida a distancias superiores a unos 70cm.

Tras su recorrido libre en el aire, a su llegada al punto de impacto contra alguna de las paredes u obstáculos el rayo seguirá las leyes de Fresnel para la frontera entre dos medios.

En este caso son solo de interés las expresiones de los coeficientes de reflexión y transmisión de la potencia, que describen la proporción de energía de los rayos reflejados y transmitidos respectivamente.

Estas ecuaciones tendrán expresiones distintas dependiendo de si tratamos ondas s-polarizadas –con el campo eléctrico perpendicular al plano de incidencia– o p-polarizadas

–con el campo eléctrico paralelo al plano de incidencia–.

$$\begin{aligned} R_{\mathrm{s}} &= \left| \frac{Z_2 \cos(\theta_i) - Z_1 \cos(\theta_t)}{Z_2 \cos(\theta_i) + Z_1 \cos(\theta_t)} \right|^2 \\ R_{\mathrm{p}} &= \left| \frac{Z_2 \cos(\theta_t) - Z_1 \cos(\theta_i)}{Z_2 \cos(\theta_t) + Z_1 \cos(\theta_i)} \right|^2 \end{aligned} \tag{13}$$

donde Z_i hace referencia a la impedancia de la onda en cada medio, dada por

$$Z = \sqrt{\frac{j\omega\mu}{\sigma + j\omega\varepsilon}} \tag{14}$$

con μ indicando la permeabilidad magnética del medio, ε la permivitidad dieléctrica y ω la frecuencia de la onda en cuestión.

Asumiendo que ambos medios son no magnéticos –es decir, $\mu = \mu_0$, como va a ser el caso en las simulaciones a realizar– las expresiones de la Ec. (15) pasan a depender de los índices de refracción de modo que

$$\begin{aligned} R_{\mathrm{s}} &= \left| \frac{n_1 \cos(\theta_i) - n_2 \cos(\theta_t)}{n_1 \cos(\theta_i) + n_2 \cos(\theta_t)} \right|^2 \\ R_{\mathrm{p}} &= \left| \frac{n_1 \cos(\theta_t) - n_2 \cos(\theta_i)}{n_1 \cos(\theta_t) + n_2 \cos(\theta_i)} \right|^2 \end{aligned} \tag{15}$$

Es posible eliminar el término dependiente del ángulo transmitido usando la ley de Snell de la refracción, de modo que las Ecs.(15) pasan a ser

$$\begin{aligned} R_{\mathrm{s}} &= \left| \frac{n_1 \cos(\theta_i) - n_2 \sqrt{1 - \left[\frac{n_1}{n_2} \sin(\theta_i)\right]^2}}{n_1 \cos(\theta_i) + n_2 \sqrt{1 - \left[\frac{n_1}{n_2} \sin(\theta_i)\right]^2}} \right|^2 \\ R_{\mathrm{p}} &= \left| \frac{n_1 \sqrt{1 - \left[\frac{n_1}{n_2} \sin(\theta_i)\right]^2} - n_2 \cos(\theta_i)}{n_1 \sqrt{1 - \left[\frac{n_1}{n_2} \sin(\theta_i)\right]^2} + n_2 \cos(\theta_i)} \right|^2 \end{aligned} \tag{16}$$

Asumiendo que el medio 1 es el aire, con índice de refracción igual a 1, se obtiene la expresión final usada en la simulación

$$\begin{aligned} R_{\mathrm{s}} &= \left| \frac{\cos(\theta_i) - \sqrt{\varepsilon_r - \sin^2(\theta_i)}}{\cos(\theta_i) + \sqrt{\varepsilon_r - \sin^2(\theta_i)}} \right|^2 \\ R_{\mathrm{p}} &= \left| \frac{\varepsilon_r \cos(\theta_i) - \sqrt{\varepsilon_r - \sin^2(\theta_i)}}{\varepsilon_r \cos(\theta_i) + \sqrt{\varepsilon_r - \sin^2(\theta_i)}} \right|^2 \end{aligned} \tag{17}$$

donde se ha utilizado la relación $n = \sqrt{\varepsilon}$, dejando las expresiones dependiendo solo de la permitividad dieléctrica del medio y el ángulo de incidencia.

Para cada uno de los dos casos, la potencia transmitida será

$$T_i = 1 - R_i \tag{18}$$

por consevación de la energía.

Queda por aclarar la polarización de la ondas emitidas. La aproximación tomada es asumir ondas no polarizadas, o lo que es lo mismo, con polarización circular, de modo que las energías de los campos eléctrico y magnético son las mismas y es posible tomar un coeficiente de reflexión de

$$R = \frac{1}{2}\left[R_{\mathrm{s}} + R_{\mathrm{p}}\right] \tag{19}$$

Por último, también existirá en los rebotes un fenómeno de difración en el que se generan rayos de forma isotrópica. Aunque es un comportamieno que se incluye en algunos modelos de propagación en exteriores –donde las antenas direccionales tienen una potencia muy concentrada–, su efecto en interiores es despreciable, donde las potencias de emisión son mucho menores.

Por ello, no se implementarán en este modelo. La potencia de cálculo extra requerida no compensa su contribución a los resultados, de modo que, al igual que en otros modelos de propagación interior, se considerará que no se produce.

2.3. Antenas

Una vez establecidas las bases para la evaluación de los rayos en el área de interés es necesario abordar su origen y su recepción.

La emisión y recepción de estos rayos se hará mediante antenas que tendrán cierta direccionalidad fruto de su simetría que será necesario considerar. Aquí aparece le concepto de direccionalidad, donde se parametriza la intensidad de la emisión o recepción dependiendo del ángulo de emisión/incidencia del ángulo.[**?**]

Esta directividad está definida como

$$D = \frac{4\pi U}{P_{\mathrm{rad}}} \tag{20}$$

donde U indica la potencia por unidad de ángulo sólido y P_{rad} la potencia total emitida.

En el caso de la antena receptora se obtiene la misma expresión, teniendo en el denominador la pontencia recibida. Este valor de recepción es a veces llamado «ganancia», aunque su origen y concepto es el mismo que la direccionalidad.

En el caso de que una antena emita isotrópicamente la direccionalidad será la misma para todos los ángulos, de modo que es posible definir la direccionalidad como el ratio entre la potencia de emisión entre la potencia de emisión si tuviese una emisión isotrópica.

3. Implementación para la simulación

Partiendo de las bases descritas en la sección anterior la simulación consistirá, de forma general, en lanzar rayos y acumular los resultados de los receptores.

Aprovechando las ventajas de la programación orientada a objetos, se considerarán los siguientes clases:

- **Rayo** Esta clase contendrá el punto de partida y el vector de dirección del rayo, así como la potencia en el origen.
- **Pared** En esta clase se incluirán los cuatro puntos de los extremos de la pared, a partir de los que se calculará su normal, también almacenada. Además, contendrá el valor de permeabilidad dieléctrica de su material.
- **Receptor** Esta clase incluye el origen de la esfera y su radio.

Las clases de las paredes y los receptores implementarán métodos que tengan de entrada un cierto rayo y que determinarán si habrá algún impacto entre ellos.

A la hora de almacenar los datos de los receptores no es posible acumular los valores de todos los rayos impactados ya que daría valores muy altos totalmente irreales. El hecho de que muchos de los rayos impacten contra los receptores es fruto de la modelización como esfera, dando más volumen del que realmente tiene la esfera.

Por ello, se dividirá el reconocimiento de impactos dependiendo de los rebotes que haya recibido. De esta forma se podrán separar los rayos que se reciben de forma directa de los rebotados.

A la hora de registrar la potencia recibida, para cada uno de los rebotes que ha sufrido el rayo, se elegirá aquel con una potencia mayor, al ser el que impacta de forma más directa con la esfera receptora, comportamiento buscado.

Estos valores se registrarán en una matriz en la que se considera a cada fila como cada receptor, y cada columna como estos valores de rebotes. Así, tras finalizar la evaluación de rayos se acumularán todos los valores en cada fila, siendo el resultado final para cada receptor.

Algoritmo 1 Bucle Principal

```
1: Leer parámetros.
2: Cargar mapa.
3: Colocar receptores.
4: for azimut ∈ [0, 360) do  ▷ Se evaluan los ángulos sumando un cierto paso fijado en los parámetros.
5:     for elev ∈ [90, −90] do
6:         eval_ray(azimut, elev, pot, data)
7:     end for
8: end for
9: for i ← 0, rows(data) do
10:    for all j ← 1, cols(data) do
11:        data[i][0] ← data[i][0] + data[i][j]
12:    end for
13: end for
14: write_results(file, data)
```

Evolución de cada rayo

El lanzamiento y evolución de cada rayo tiene varias fases a discutir.

En primer lugar, una vez se ha determinado la dirección de lanzamiento se debe calcular la potencia a emitir. Habiendo importado la direccionalidad de la antena de emisión, se realiza una interpolación bidimiensional[2] para determinar el valor exacto.

Como se comentaba en la sección del fundamento teórico, en cada impacto se generarán dos rayos. Así, si este impacto llega a producirse, será necesario almacenar uno de los rayos producidos para ser evaluado más tarde.

Este almacenamiento se hará en forma de pila, de tal forma que los rayos añadidos más recientemente serán los primeros en ser evaluados. Esta elección es totalmente arbitraria: todos los rayos son independientes entre sí.

Para optimizar el uso de memoria de la pila se reservará toda la memoria que se podría usar antes de iniciar el bucle y así no necesitar movimientos posteriores. El tamaño necesario vendrá determinado por el número de rebotes que se quieran registrar.

En la Figura **??** se indica el árbol generado por un rayo, donde en cada impacto nacen dos rayos. Considerando que siempre se evalúa el rayo reflejado –en esta representación, el camino de la izquierda– podría parecer que se deberá reservar memoria para todos los rayos restantes, pero es posible un uso menor.

Los rayos guardados en la pila no son evaluados de forma inmediata, así que los rayos que se generan no se conocerán hasta que se llegue a su nodo. Es decir, el número de rayos en la pila no llegará nunca a ser tan alto.

3.1. Implementación en paralelo

El hecho de que los rayos lanzados sean independientes entre sí convierte en el bucle principal en el escenario perfecto para ser ejecutado de forma paralela, ya que cada iteración del bucle no se verá interrumpida por las demás.

Para esta paralelización se ha hecho uso de tarjetas gráficas, que cuentan con un gran número de núcleos de cómputo pero para las que hay que tener en cuenta su estructura de memoria.

En general, a la hora de paralelizar un algoritmo es necesario hacer duplicaciones de datos para evitar condiciones de carrera que proporcionen resultados erróneos. En los casos de muy alta paralelización como este es posible encontrarse con ciertas limitaciones.

La tarjeta gráfica usada en este caso ha sido una Nvidia GTX1070, sobre la que es posible el uso de la librería de Nvidia CUDA. Esta tarjeta cuenta con 1920 núcleos agrupados en 15 multiprocesadores.

La estrategia de paralelización consistirá en asignar un rayo a cada hilo, que contará con su matriz de datos correspondiente donde registrará los impactos para luego ser acumulada en los datos finales.

Las tarjetas gráficas agrupan sus hilos en bloques. Los hilos que se encuentren en un mismo bloque pueden acceder a una cierta cantidad de memoria compartida de un acceso más rápido que la memoria global común.

Para reducir la necesidad de memoria y aprovechar esta característica la acumulación de los datos se producirá en dos fases: el cálculo de cada rayo –asignado a cada hilo– volcará

[2]En este caso esta interpolación ha sido bilineal. Aunque es posible obtener unos valores más precisos con métodos de mayor orden, partiendo de puntos separados solo un grado en ambos ángulos esta elección proporciona valores aceptables sin tener que abordar las condiciones de contorno en los extremos.

Algoritmo 2 Bucle que evalúa cada rayo

```
1: while Tamaño pila >0 do
2:     reb ← segundo valor de la pareja {rayo, rebote}.
3:     for all Paredes do
4:         hit_dist ← INFINITY
5:         wall_hit ← -1
6:         if Pared golpeada then
7:             if dist <hit_dist then
8:                 hit_dist ← dist
9:                 wall_hit ← pared
10:            end if
11:        end if
12:    end for
13:    for all Receptores do
14:        hit_dist ← INFINITY
15:        power ← 0
16:        if Receptor golpeado then
17:            if hit_power >power and dist <hit_dist then
18:                hit_power ← power
19:            end if
20:        end if
21:    end for
22:    if power >CUTOFF_POWER AND reb <MAX_REBOUND then
23:        {rayo, rebote} ← { rayo_reflejado, reb+1}
24:        Añadir { rayo_transmitido, reb+1} a la pila.
25:    else
26:        {rayo, rebote} ← último valor de la pila.
27:    end if
28: end while
```

sus datos en la memoria compartida del bloque. Una vez acaban todo los hilos del bloque, se acumulan los datos en la memoria global, donde se ha reservado espacio para cada bloque.

Debido a la herencia del uso de este tipo de dispositivos para trabajos de vídeo, es posible acceder a los hilos y bloques con dos índices, de forma similar a una matriz. Esta característica será útil en este caso, en el que los rayos se lanzarán en torno a los dos ángulos de las coordenadas esféricas.

Así, se asociará la dirección x al ángulo azimutal y la dirección y a la elevación.

Queda determinar el número de hilos en cada bloque. En general, la memoria disponible para su uso compartido es una cantidad baja. En el caso de la tarjeta gráfica usada, es de 48 KB[**?**].

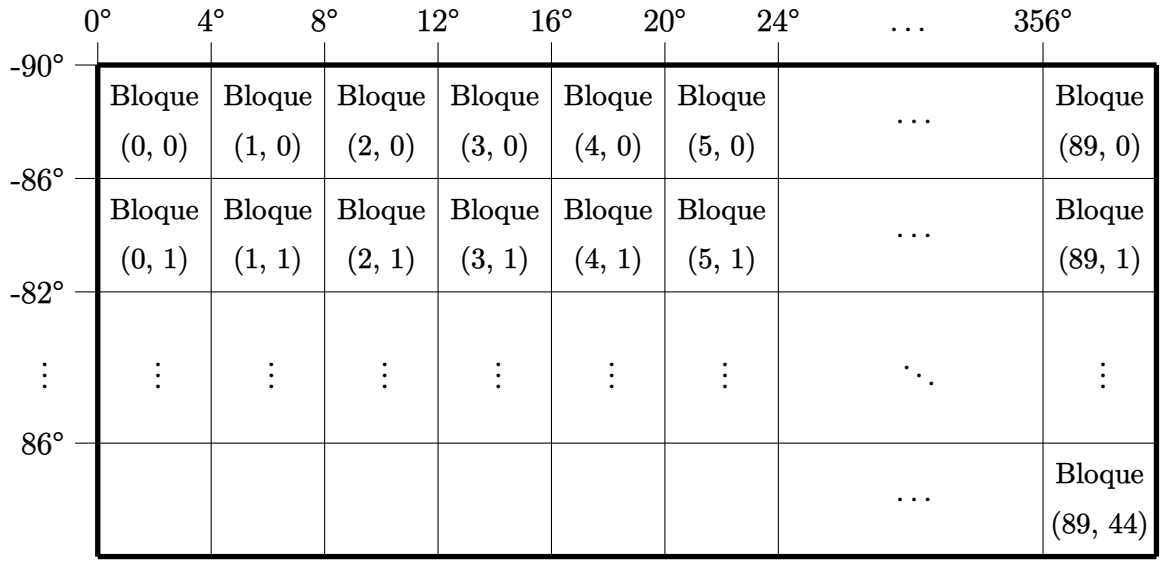


Figura 1: Esquema con la asignación de ángulos a cada bloque de hilos.

Esta cantidad no permite lanzar bloques con un número grande de hilos, aunque dependerá del número de receptores y los rebotes que se quieran registrar. A modo de ejemplo, al colocar unos 40 receptores y registrar los rayos hasta 5 rebotes, con valores de precisión simple –es decir, 4 bytes–, podremos lanzar bloques con 60 hilos como máximo.

Para evitar futuras incompatibilidades en caso de elegir valores mayores en alguno de estos dos parámetros, se han elegido el mínimo valor de hilos posible. La elección es la siguiente: solo se lanzarán 16 hilos por bloque, 4 en cada dirección.

Además de ser un valor bajo compatible con los límites establecidos, el número de ángulos a evaluar en ambas direcciones es divisible por 4, hecho aprovechable para la implementación.

La elección de asignación de valores angulares a los bloques será de forma fija: cada bloque solo evaluará 4 grados en cada dirección, independientemente del número de rayos que tenga asignado ejecutar.

Está claro que si la distancia entre los ángulos de un grado, cada hilo ejecutará el rayo con los ángulos que le corresponde sin más. En el caso –habitual– que la distancia sea menor, cada hilo deberá evaluar más de un rayo.

Para facilitar el recorrido y garantizar una ejecución lo más homogénea posible[3], solo se considerarán saltos en los valores angulares como potencias negativas de dos –es decir, solo

[3]Hay que tener en cuenta que la ejecución de los programas en tarjetas gráficas se realizan ejecutando

se usarán saltos de valor 0,5, 0,25, 0,125,...– a fin de poder utilizar la siguiente estrategia.

Se considerará una matriz de «minibloques» dentro del bloque, todas ellas con los 16 hilos del bloque. La cantidad de estos minibloques dependerá del valor del salto entre ángulos: si es $0{,}5 \equiv 1/2$ habrá 4 minibloques, dos en cada dirección; si es $0{,}25 \equiv 1/4$ habrá 16 minibloques, etc.

Dentro de cada uno de los minibloques solo habrá 16 pares de direcciones a evaluar, uno para cada hilo. Una vez todos estos hilos finalicen su bucle, se avanzará al siguiente minibloque en dirección vertical –es decir, avanzando en el eje y– hasta acabar con todos los ángulos que se deben evaluar.

una misma instrucción en varios hilos a la vez. Evitar expresiones condicionales que creen distintas ramas en el desarrollo del programa maximizará el rendimiento del dispositivo.

4. Resultados

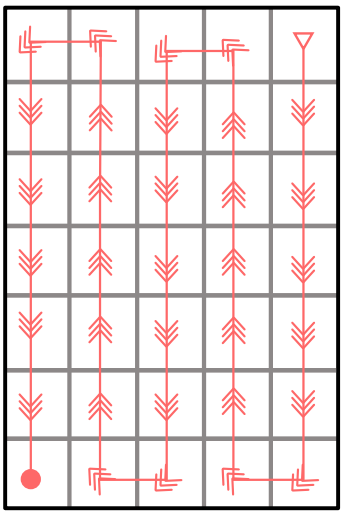
4.1. Entorno de pruebas

Para comprobar cómo se ajusta el modelo al comportamiento real de la emisión se tomaron medidas en el Laboratorio de Robótica 0L3 del Instituto de Computación Científica Avanzada de la Universidad de Extremadura (ICCAEx), situado en los Institutos Universitarios de Investigación de la Universidad de Extremadura en Badajoz.

En este laboratorio se dispone de una superficie amplia en la que fue posible usar un robot para automatizar la toma de medidas en divesos puntos, que más tarde fueron usados en la simulación colocando los receptores en dichos puntos.

31	32	33	34	35
26	27	28	29	30
21	22	23	24	25
16	17	18	19	20
11	12	13	14	15
6	7	8	9	10
1	2	3	4	5

(a) Puntos a evaluar



(b) Trayectoria en vertical

Figura 1: Puntos de medida por el robot en el laboratorio y la trayectoria tomada.

La Figura 1 recoge la disposición de los puntos tomados y la ruta del robot para ello. En total se compone de 35 puntos en los el robot se para y toma medidas cada 5 segundos.

Esta trayectoria se repitió unas 20 veces de modo que fue posible hacer estimaciones sobre todos los puntos con un mayor volumen de datos que pudiera eliminar los efectos de otras redes del edificio.

4.2. Robot

Con el fin de automatizar la toma de medidas para hacerla lo más eficiente posible se ha decidido usar un robot móvil capaz de desplazarse mediante navegación autónoma en un entorno conocido.

El elegido en este caso fue el robot TurtleBot 2, un robot con fines educativos y de investigación capaz de desplazarse y orientarse con total libertad en superficies llanas, como eran los en el que se desarrolló este trabajo. En la Figura 2 se puede ver el robot durante una de las tomas de medidas.

Figura 2: Turtlebot en una de las tomas de medidas.

El TurtleBot funciona con ROS –*Robot Operating System*, en inglés–, un entorno de trabajo enfocado a los sistemas robóticos encargado del soporte para todos los dispositivos hardware del robot como motores, encoders o cámaras. Cuenta con diversas librerías para distintos casos de uso, entre las que se encuentra la navegación en un entorno controlado como la que se da en el caso de este trabajo.

Su funcionamiento se basa en una arquitectura de grafos, donde se definen *nodos*, tareas en las que se realiza el procesamiento de sensores, control, actuadores o cualquier otra función. Los nodos se comunican entre ellos con mensajes llamados *topic*, y es posible su desarrollo con los lenguajes C++ y Python.

Dentro de los paquetes disponibles, el usado en el desarrollo del trabajo es el encargado de la navegación del robot, que permite desplazar al robot a cualquier punto del entorno de trabajo y proporcionar de forma constante su posición en el mapa.

Es posible acoplar al robot un sistema de visión llamado Kinect que, mediante luz, permite obtener un mapa de profundidad del entorno que lo rodea. Fue desarrollado en primera instancia para el uso en videojuegos pero su uso también se ha extendido a labores de investigación al permitir el posicionamiento de objetos y paredes, de tal manera que permite al robot evitar obstáculos dinámicos en el caso de que interpongan en su camino.

La funcionalidad de navegación aprovecha estos sensores realizando una fusión de sus resultados con los datos de movimiento de los motores que impulsan al robot, de tal forma que es posible corregir cualquier error en casos donde el empuje de las ruedas no se traslade directamente en un desplazamiento del robot, como puede ocurrir al rotar sobre sí mismo.

Para conseguir el posicionamiento en el mapa el paquete de navegación utiliza un *planner* global sobre el que es posible determinar las rutas a seguir por el robot para llegar a un punto dado sorteando obstáculos y paredes. Junto a él trabaja un *planner* local, en el que gracias a los sensores incorporados se evalúan de forma continua los alrededores del robot. Así, es posible conseguir de forma continua una evaluación de los posibles obstáculos que no se encuentran en el mapa del planner global, para lo que se construye un mapa de costes con el

fin de abortar el movimiento en el caso de que sea imposible alcanzar el punto objetivo [**?**].

Con la información actualizada del *planner* local, el *planner* global es capaz de modificar los trayectos para que el robot pueda continuar su desplazamiento por el mapa. Es posible observar un esquema del funcionamiento de estos sistemas en la Figura 3.

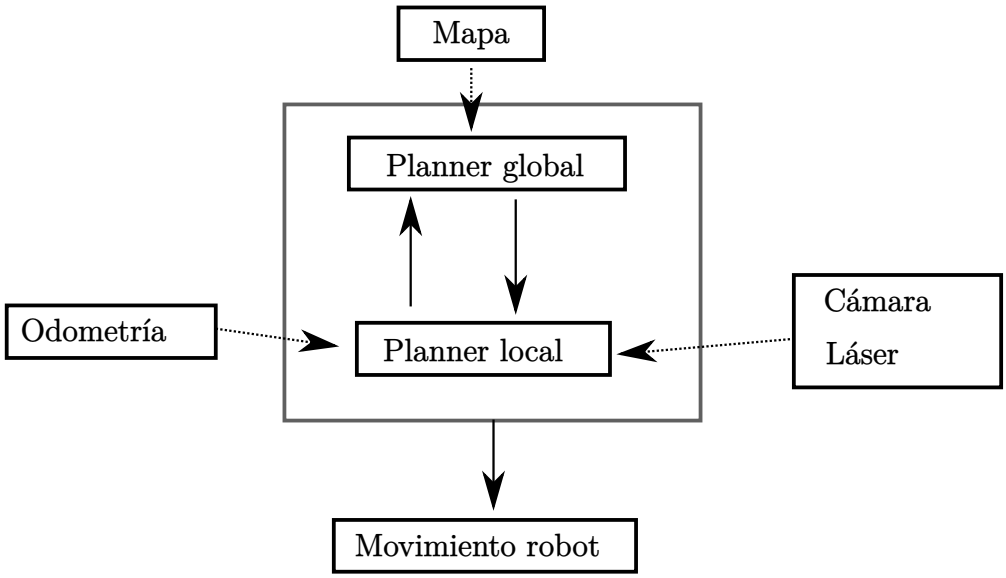


Figura 3: Esquema del funcionamiento del paquete de navegación de ROS.

Con esta herramienta la librería de navegación permite un mapeado autónomo tomando como referencia los datos de odometría que proporcionan los motores propulsores del robot para determinar las dimensiones del entorno.

En el caso de disponer de antemano de dichas dimensiones es posible proporcionar un mapa al sistema de navegación y evitar el paso de reconocimiento del entorno. Esto no solo ahorra tiempo, sino que además minimiza las posibles discrepancias entre los datos de odometría y los desplazamientos reales del robot al realizar el mapeado de forma autónoma.

La opción de realizar un mapa previo fue la elegida en este caso, ya que el robot cuenta con rutinas para el reposicionamiento en el mapa en dicho caso. Así, a partir de los límites establecidos y comprobados de forma manual, las posibles discrepancias en la odometría del robot se ven continuamente compensadas y corregidas.

A partir de estos datos corregidos es posible conocer en cualquier momento la posición del robot en el mapa, expuesta a través de ROS en uno de los topic disponibles.

Para facilitar el uso de ROS existe la posibilidad de usar el simulador Stage, capaz de crear un mundo virtual a partir de un mapa en dos dimensiones en el que colocar el Turtlebot y simular su funcionamiento de forma total sin tener acceso al robot de forma física. Es posible observar su interfaz en la Figura 4.

Además, ROS también permite el uso de una herramienta de visualización de la posición del robot en el mapa y de todos los sensores que incorpora llamada *rviz*. Aunque es compatible con Stage, sus funcionalidades brillan al usar el robot en entornos reales, donde es posible comprobar de forma continua que su posicionamiento es correcto y que sus sensores funcionan como es debido.

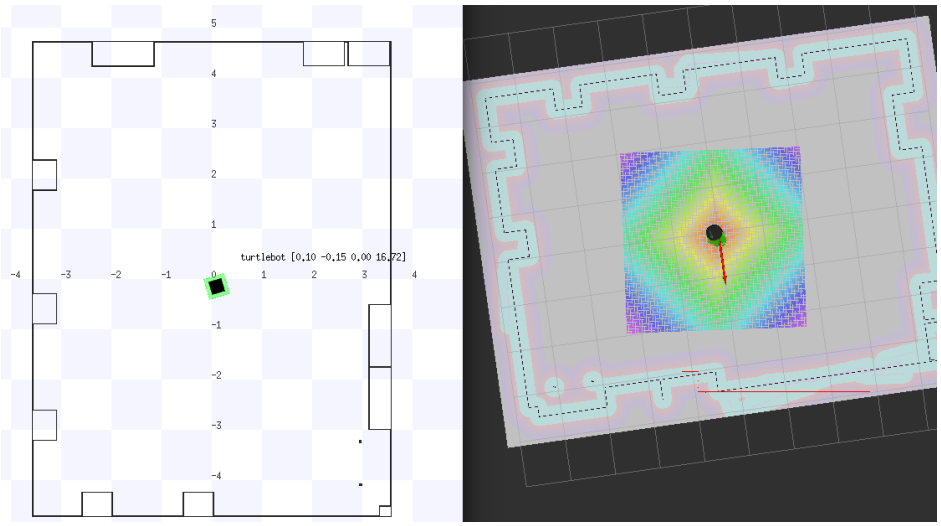


Figura 4: Captura del simulador Stage (a la izquierda) y la herramienta de visualización *rviz* (a la derecha).

5. Conclusiones

En este trabajo se ha desarrollado un modelo de emisión electromagnética con la aproximación de trazado de rayos para su modelización en propagación en interiores.

6. Bibliografía

[1] C. A. Balanis, *Antenna Theory: Analysis and Design.* Wiley, 2016.

[2] Nvidia Corporation, "CUDA C Programmig Guide: Features and Technical Specifications [en línea]." Dirección URL: <https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.htmlfeatures-and-technical-specifications>. [Consulta: 25 de junio de 2021].

[3] ROS Documentation, "move_base package [en línea]." Dirección URL: <http://wiki.ros.org/move_base>. [Consulta: 14 de julio de 2021].