XLIV Jornadas de Automática 2023

https://jautomatica.es

CollisionGP: comprobación de colisiones probabilística con procesos gaussianos

Muñoz, J.[a,*], Moreno, L.[a]

[a]*RoboticsLab, Departamento de Ingeniería de Sistemas y Automática, Universidad Carlos III de Madrid, Avenida de la Universidad, nº30, 28911, Leganés*

To cite this article: Muñoz, J., Moreno, L. 2023. CollisionGP: probabilistic collision checking with Gaussian processes. XLIV Jornadas de Automática, 691-696. https://doi.org/10.17979/spudc.9788497498609.691

Resumen

La comprobación de colisiones es la operación primitiva de la planificación de movimientos que más tiempo consume. Se ha demostrado que los algoritmos de aprendizaje automático aceleran la comprobación de colisiones. Presentamos CollisionGP, un algoritmo basado en procesos gaussianos para modelar el espacio de configuraciones de un robot y comprobar colisiones. CollisionGP introduce una variable auxiliar Pòlya-Gamma para cada punto de datos en el conjunto de entrenamiento para permitir que la inferencia de clasificación se realice exactamente con una expresión de forma cerrada. Los procesos gaussianos proporcionan una distribución como salida, obteniendo una media y una varianza para la comprobación de colisión. La varianza obtenida se procesa para reducir los falsos negativos (FN). Demostramos que CollisionGP puede utilizar la aceleración de la GPU para procesar comprobaciones de colisiones para miles de configuraciones mucho más rápido que las librerías tradicionales de detección de colisiones. Además, obtenemos mejores resultados de precisión, ratio de verdaderos positivos (TPR) y verdaderos positivos (TNR) que los algoritmos del estado del arte basados en aprendizaje utilizando menos puntos de soporte, lo que hace que nuestro método sea más ligero.

Palabras clave: evasión de colisiones, procesos gaussianos, aprendizaje automático, planificación de movimientos

CollisionGP: probabilistic collision checking with Gaussian processes

Abstract

Collision checking is the primitive operation of motion planning that consumes most time. Machine learning algorithms have proven to accelerate collision checking. We propose CollisionGP, a Gaussian process-based algorithm for modeling a robot's configuration space and query collision checks. CollisionGP introduces a Pòlya-Gamma auxiliary variable for each data point in the training set to allow classification inference to be done exactly with a closed-form expression. Gaussian processes provide a distribution as the output, obtaining a mean and variance for the collision check. The obtained variance is processed to reduce false negatives (FN). We demonstrate that CollisionGP can use GPU acceleration to process collision checks for thousands of configurations much faster than traditional collision detection libraries. Furthermore, we obtain better accuracy, true positive rate (TPR) and true negative rate (TNR) results than state-of-the-art learning-based algorithms using less support points, thus making our proposed method more sparse.

Keywords: collision avoidance, Gaussian processes, machine learning, motion planning

1. Introducción

La planificación de trayectorias es la tarea de crear una trayectoria para mover un robot desde una configuración inicial hasta una configuración de destino, evitando al mismo tiempo las autocolisiones y las colisiones con el entorno. La mayoría de las soluciones de planificación de trayectorias se calculan en el espacio de configuraciones (C-space) del robot y no en el espacio cartesiano del entorno. En el C-space, cada dimensión representa un grado de libertad (DoF) del robot (Choset (2005)). A medida que aumenta el número de DoFs del robot, la dimensionalidad del C-space crece, aumentando así la complejidad

*Autor para correspondencia: jamunozm@ing.uc3m.es

Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

del problema de planificación de trayectorias. Para planificar trayectorias libres de colisiones, los algoritmos de planificación de trayectorias se basan en comprobadores de colisiones. Estos algoritmos determinan si una configuración específica pertenece al espacio de configuración libre C_{libre} o al espacio de configuración bloqueado por obstáculos C_{obs}. Las consultas al comprobador de colisiones son la operación primitiva más costosa desde el punto de vista computacional de los algoritmos de planificación de trayectorias, ya que suponen alrededor del 90 % del tiempo de cálculo (Elbanhawi and Simic (2014)).

El espacio de configuraciones puede modelarse mediante técnicas estándar de aprendizaje automático, lo que permite al modelo recibir consultas del algoritmo de planificación de trayectorias. Sin embargo, este modelo pierde su validez cuando cambia cualquier aspecto del entorno. Los algoritmos no pueden actualizar con la suficiente rapidez modelos con millones de pesos, como las redes neuronales artificiales (ANN), para que resulten útiles en entornos cambiantes. En este artículo, presentamos un nuevo enfoque para la detección de colisiones mediante procesos gaussianos (GP), con el objetivo de construir un modelo más ligero y menos costoso computacionalmente que reduzca el tiempo de cálculo empleado en la comprobación de colisiones durante las aplicaciones de planificación de trayectorias.

Proponemos CollisionGP, un modelo GP ligero basado en variables auxiliares Pòlya-Gamma para clasificación binaria (Polson et al. (2013)). Esta es la primera aplicación de GPs al problema de comprobación de colisiones en el espacio de configuración de un robot, hasta donde sabemos. Otros métodos, sin embargo, utilizan GPs para la planificación de movimientos (Dong et al. (2016); Mukadam et al. (2016); Chou et al. (2022)). Las principales ventajas de los GPs es que son fáciles y rápidos de entrenar ya que su número de hiperparámetros es significativamente menor que otros métodos como las ANNs, y proporcionan una distribución probabilística completa como salida del modelo. Esto nos permite no sólo determinar si un punto de consulta pertenece a C_{libre} o C_{obs}, sino medir la incertidumbre del modelo sobre esa predicción. Usamos esta medida de incertidumbre para ponderar el límite de decisión hacia la eliminación de falsos negativos, que llevarían a una colisión del robot con un obstáculo o consigo mismo. Esto es extremadamente importante en aplicaciones críticas como la manipulación de residuos químicos o la robótica quirúrgica.

Elegimos el kernel cuadrático racional (RQ) con determinación automática de relevancia (ARD) como el kernel del GP, que mejora la precisión del modelo al descartar los DoF que no influyen directamente en las colisiones. Para obtener un modelo ligero, utilizamos un conjunto de puntos de inducción para realizar las predicciones sobre las consultas del algoritmo de planificación de trayectorias. Demostramos que se necesitan pocos puntos de inducción para obtener mapas del C-space fiables y precisos.

2. Estado del arte

Los métodos de aprendizaje automático se han utilizado previamente para modelar el C-space de un robot y acelerar la comprobación de colisiones en la planificación de trayectorias.

Pan and Manocha (2015) utilizan máquinas de soporte vectorial (SVM) para generar una frontera de decisión precisa entre C_{libre} y C_{obs} para dos objetos, y presentan una estrategia de aprendizaje activo para mejorar esta frontera de forma iterativa. En nuestro caso, los cálculos de CollisionGP pueden procesarse en línea y el algoritmo no necesita un nuevo modelo para cada par de objetos.

Arslan and Tsiotras (2015) utilizan un clasificador bayesiano para calcular dos funciones de densidad de probabilidad aproximadas para determinar C_{libre} y C_{obs} basándose en los datos disponibles y, a continuación, utilizan la regla bayesiana para predecir si un punto de consulta pertenece a C_{libre} o C_{obs}. CollisionGP tiene la ventaja sobre los clasificadores bayesianos ingenuos de poder proporcionar una varianza predictiva para los resultados del modelo.

Danielczuk et al. (2021) utilizan una red neuronal que aprende las colisiones a partir de escenas, representadas como nubes de puntos, y es capaz de predecir colisiones para poses de objetos de 6 DoF dentro de la escena. Sin embargo, necesitan 1 millón de pares de nubes de puntos escena/objeto y más de 2.000 millones de consultas de colisión para entrenar la red, y necesitan una cámara con percepción de profundidad para realizar comprobaciones de colisión. Demuestran la aplicabilidad del método con un robot que realiza tareas de reordenación sobre una mesa.

Chase Kew et al. (2020) presentan ClearanceNet y CN-RRT, una heurística de comprobación de colisiones basada en una red neuronal y un algoritmo de planificación, respectivamente. CN-RRT utiliza la capacidad de ClearanceNet de procesar lotes de miles de comprobaciones de colisiones para planificar rutas de forma eficiente y rápida.

Qureshi et al. (2019) utilizan una red neuronal que codifica el espacio de trabajo a partir de una nube de puntos y calcula trayectorias sin colisiones para el robot. El entrenamiento se realiza mediante demostraciones de expertos, utilizando 4000 demostraciones para 100 espacios de trabajo diferentes. La red neuronal es capaz de generar múltiples trayectorias libres de colisión para una única combinación de configuraciones de inicio y objetivo en un tiempo finito debido a su comportamiento estocástico. La principal ventaja de CollisionGP sobre los enfoques de redes neuronales es que nuestro método se basa en un conjunto limitado de puntos de datos y puede actualizarse con obstáculos dinámicos, mientras que las redes neuronales requieren decenas de miles de puntos de datos y no son tan flexibles. Además, CollisionGP no requiere muchos datos ni el ajuste de parámetros para el entrenamiento, al no tener que elegir la arquitectura ni necesitar optimizar los cientos o millones de pesos de la red. Adicionalmente, CollisionGP puede proporcionar distribuciones probabilísticas completas como predicciones del modelo.

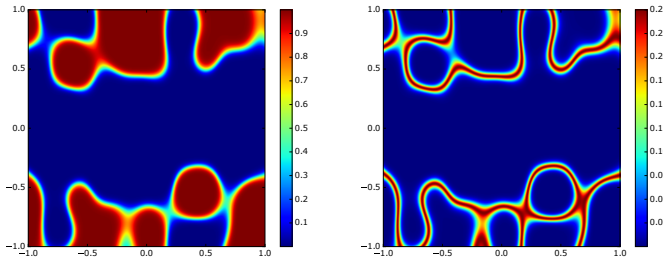
Pan and Manocha (2016) almacenan muestras de consultas anteriores y utilizan la búsqueda KNN (k-nearest neighbor) para encontrar muestras de consultas anteriores cercanas a la nueva configuración de consulta. A continuación, estiman la probabilidad de que la nueva configuración de consulta pertenezca a C_{libre} o C_{obs} basándose en las muestras de consulta encontradas. CollisionGP tiene la ventaja de ser escalable, ya que todas las comprobaciones de colisión no se almacenan en tablas que crecen con el tamaño del dataset, y también es capaz de propor-

cionar varianzas predictivas.

Das and Yip (2020) proponen Fastron, un algoritmo basado en el aprendizaje que utiliza una SVM para clasificar los puntos pertenecientes a C_{libre} y C_{obs}. Demuestran que pueden entrenar el modelo y comprobar colisiones un orden de magnitud más rápido que las bibliotecas de colisión más avanzadas, lo que lo hace capaz de adaptarse a entornos dinámicos. Demuestran la aplicabilidad a robots realizando tareas de pick and place y cirugía. CollisionGP mejora este enfoque utilizando ARD sobre las entradas del modelo, lo que le permite mantener su precisión con cualquier robot, y proporcionando valores de incertidumbre para las salidas del modelo.

3. Metodología

Esta sección describe los componentes del algoritmo CollisionGP. Comenzamos describiendo el problema de clasificación binaria aplicado a los GP. A continuación, ofrecemos una visión general de la selección del kernel para la tarea. Posteriormente, entramos en los detalles de la generación del dataset y la creación y entrenamiento del modelo GP. En la siguiente subsección mostramos cómo el algoritmo propuesto toma la media y la varianza de la salida generada para hacer predicciones sobre los puntos de consulta. Finalmente, destacamos la capacidad del modelo para aprovechar las ventajas de CUDA y utilizar lotes de datos como entradas en lugar de puntos de consulta individuales, lo que puede paralelizar los cálculos y proporcionar predicciones más rápidas según se evalúa en una comparación con el método FCL tradicional.



(a) (b)

Figura 1: Media y varianza obtenidas al realizar comprobaciones de colisión en todos los puntos disponibles para los dos primeros DoF del robot KUKA iiwa. a) Valores medios obtenidos al realizar predicciones para todos los puntos. b) Valores de varianza obtenidos al realizar predicciones para todos los puntos.

3.1. *Variables auxiliares Pòlya-Gamma: clasificación binaria*

Los procesos gaussianos en el contexto de la clasificación no admiten la inferencia exacta con una expresión de forma cerrada. Una posible solución es introducir variables latentes adicionales que restauren la conjugación. Introducimos una variable auxiliar Pòlya-Gamma para cada punto de datos en el dataset de entrenamiento. Wenzel et al. (2019) demuestran que este método de clasificación binaria es hasta dos órdenes de magnitud más rápido que el estado del arte, al tiempo que es competitivo en términos de rendimiento de predicción, y es capaz de trabajar con conjuntos de datos con decenas de millones de puntos

La inferencia en modelos GP exactos típicamente tiene una complejidad temporal de $O(n^3)$, donde n es el número de puntos de datos en el modelo, y $O(n)$ y $O(n^2)$ para calcular la media y la varianza respectivamente. Esto hace a los modelos GP exactos inviables cuando se abordan conjuntos de datos grandes, aunque el desarollo de librerías como GPyTorch (Gardner et al. (2018)) permiten usar aceleración por GPU, lo que los hace más rápidos de entrenar y acelera las predicciones. Hensman et al. (2015) proponen un método para reducir la complejidad a $O(m^3)$, donde m es el número de puntos de inducción.

Los hiperparámetros del kernel y la ubicación de los puntos de inducción se optimizan utilizando el optimizador Adam. La función de pérdida utilizada para el optimizador es la variational evidence lower bound (ELBO), que se utiliza comúnmente para optimizar GPs variacionales.

3.2. *Selección del kernel*

La función kernel $k(\mathbf{x_1}, \mathbf{x_2})$ compara una configuración $\mathbf{x_1}$ con $\mathbf{x_2}$ mediante el mapeo a algún espacio de características y tomando un producto interno. Esta función debe proporcionar un valor grande para configuraciones similares (lo que significa que están fuertemente correlacionadas) y un valor pequeño para configuraciones diferentes. El kernel seleccionado para esta aplicación es el kernel RQ (cuadrático racional), que puede verse como una mezcla de escalas (una suma infinita) de kernels RBF (función de base radial) con diferentes escalas de longitud características. Además, una opción típica para los modelos GP es añadir un vector de hiperparámetros Θ de determinación automática de relevancia (ARD) (MacKay (1995)),

$$k_{\mathrm{RQ}}(\mathbf{x_1}, \mathbf{x_2}) = \left(1 + \frac{1}{2\alpha}(\mathbf{x_1} - \mathbf{x_2})^\top diag(\Theta)^{-2}(\mathbf{x_1} - \mathbf{x_2})\right)^{-\alpha} \quad (1)$$

donde $diag(\Theta)$ es una matriz diagonal con d entradas Θ a lo largo de la diagonal, siendo d el número de dimensiones de los datos de entrada. Intuitivamente, si un Θ_i particular tiene un valor grande, el kernel se vuelve independiente de la i-ésima entrada, eliminándola automáticamente. Por lo tanto, se descartan las dimensiones irrelevantes.

3.3. *Generación del dataset y creación del modelo*

Para crear el dataset, se genera un conjunto aleatorio uniforme de N configuraciones sin etiquetar X, y las etiquetas verdaderas Y las proporciona la biblioteca de detección de colisiones FCL (Pan et al. (2012)). Para tratar los límites de las articulaciones del robot, asignamos el espacio delimitado de articulaciones d-dimensional a un espacio de entrada d-dimensional $[-1, 1]^d$. Para asignar los valores articulares q_{joint} al espacio de entrada q_{input}, utilizamos:

$$q_{input} = (2q_{joint} - q_u - q_l) \oslash (q_u - q_l) \quad (2)$$

donde q_l y q_u son vectores que contienen los límites inferior y superior de las articulaciones, y $\oslash$ realiza la división elemento a elemento.

Una vez que el dataset está listo, la distribución PG y el modelo GP se inicializan y sus hiperparámetros se optimizan como se explica en la sección 3.1. A continuación, se realizan las comprobaciones de colisiones con el modelo actualizado para evaluar su rendimiento.

3.4. *Hacer predicciones*

La Figura 1 proporciona un ejemplo de comprobaciones de colisión para los dos primeros DoF de un manipulador KUKA iiwa, utilizando 128 puntos de inducción. Los valores medios obtenidos se encuentran en el intervalo [0, 1], ya que están normalizados con una distribución Bernoulli. Los puntos predichos como puntos libres en el espacio tienen valores medios más cercanos a cero, mientras que los puntos predichos como obstáculos tienen valores medios más cercanos a uno. La varianza refleja los límites de los obstáculos, ya que los valores de la varianza son próximos a cero en el espacio libre y en el interior de los obstáculos, pero alcanzan sus valores máximos en los límites entre C_{libre} y C_{obs}.

Cuando el algoritmo decide basándose únicamente en la media para determinar si un punto pertenece a C_{libre} o C_{obs}, tomando todos los puntos con una media igual o superior a 0,5 como obstáculos y los puntos con una media inferior a 0,5 como espacio libre, corremos el riesgo de obtener falsos negativos, es decir, puntos clasificados como espacio libre cuando en realidad pertenecen a un obstáculo, lo que provocaría colisiones al ejecutar un movimiento con el robot. Para evitar falsos negativos, aprovechamos el hecho de que los GPs proporcionan la varianza σ además de la media μ, y proponemos un nuevo valor de decisión para clasificar los puntos, γ, como

$$\gamma = \mu + \beta * \sigma \tag{3}$$

donde β es la variable que modifica la influencia de la varianza σ en la decisión.

4. Experimentos

En esta sección evaluamos CollisionGP frente a Fastron (Das and Yip (2020)), un método de aprendizaje automático del estado del arte para la comprobación de colisiones, para 2 y 7 DoFs actuados de un robot de 7 DoFs en 15 entornos diferentes con 4 obstáculos. En la Figura 2 se muestra un ejemplo de entorno de prueba.

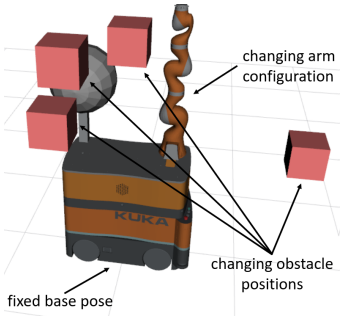
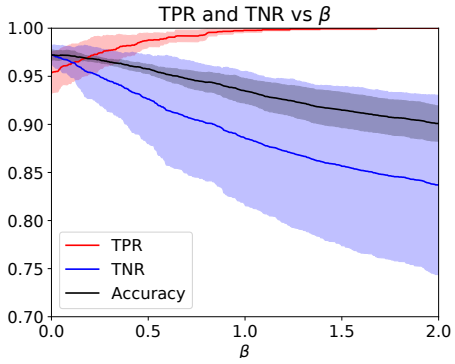


Figura 2: Modelo RViz del robot usado para los experimentos. Consiste en una base móvil KUKA KMR con un brazo KUKA iiwa. Las cajas rojas representan obstáculos en el entorno.

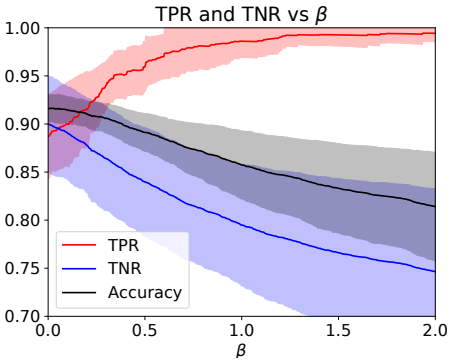
Comparamos el número de puntos de inducción, los tiempos de consulta y los tiempos de entrenamiento para entornos estáticos con un conjunto de datos de $N = 10000$ configuraciones de entrenamiento. Las pruebas de CPU de Fastron y CollisionGP se ejecutan en un procesador Intel Xeon E5-1620 v3 @3.5 GHz, y las pruebas de GPU de CollisionGP se ejecutan en una tarjeta gráfica Nvidia RTX 3060.

4.1. *Selección de β*

Para seleccionar β para este caso particular, evaluamos los modelos de dos y siete DoF para diferentes valores de β para ver cómo evolucionan la precisión, TPR (True Positive Rate) y TNR (True Negative Rate). La Figura 3 muestra la evolución del TPR y TNR ante el aumento de β. Para valores de β superiores a 0,5 el TPR está por encima de 0,95 mientras que el TNR se sostiene por encima de 0,85, reduciendo el número de falsos negativos a un umbral aceptable. En los experimentos posteriores seleccionamos $\beta = 0{,}5$ para todos los DoFs.



(a) Dos DoF



(b) Siete DoF

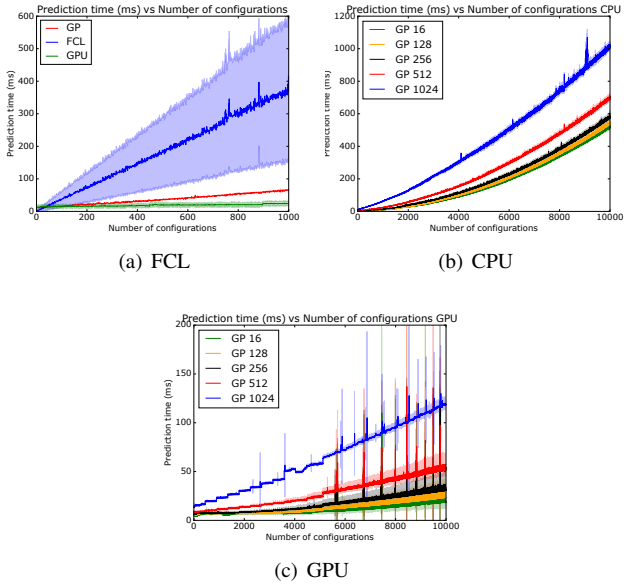
Figura 3: TPR y TNR de CollisionGP para valores de β para el robot de siete DoF en cinco escenarios distintos con cuatro obstáculos estáticos.

4.2. *Cálculos por lotes*

PyTorch permite procesar los datos por lotes, lo que agiliza las predicciones ya que el software paraleliza los cálculos necesarios. La Figura 4 muestra el tiempo necesario para calcular un determinado número de comprobaciones de colisión a la vez para el método FCL y para CollisionGP utilizando la CPU y la GPU con CUDA.

La Figura 4 muestra la media y la desviación estándar del tiempo necesario para calcular las colisiones con los modelos GP en función del número de configuraciones y puntos de inducción. La implementación de CPU muestra que los tiempos de predicción crecen exponencialmente a medida que aumenta el número de configuraciones, mientras que para la implementación de GPU aumentan linealmente. Obsérvese que la implementación para GPU es el método más rápido y eficiente, y que los tiempos de cálculo para CollisionGP presentan un intervalo

de confianza estrecho, mientras que el método FCL tiene un intervalo de confianza más amplio. Para la comparación con FCL se utilizan 512 puntos de inducción.



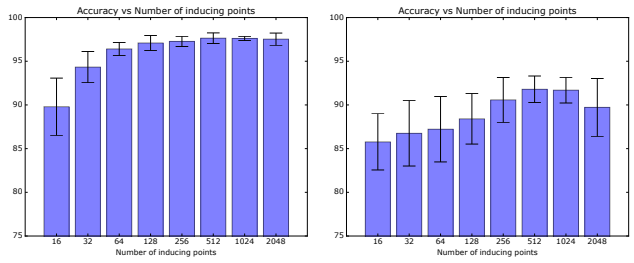
(a) FCL (b) CPU

(c) GPU

Figura 4: Tiempo de predicción frente a número de configuraciones para distintos números de puntos de inducción para el caso de siete DoF.

4.3. *Seleccionar el número de puntos de inducción*

Para seleccionar el número de puntos de inducción para cada DoF, entrenamos los modelos para diferentes conjuntos de puntos de inducción para evaluar el rendimiento del modelo a medida que aumenta el número de puntos de inducción. La Figura 5 muestra la precisión de los modelos al accionar dos y siete DoFs del robot dependiendo del número de puntos de inducción seleccionados y utilizando $\beta = 0{,}5$. La precisión aumenta gradualmente a medida que aumenta el número de puntos de inducción hasta que el valor alcanza un límite, que es la máxima precisión posible que puede alcanzarse con CollisionGP para la β seleccionada.



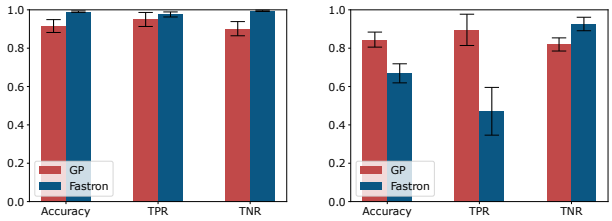
(a) Dos DoF (b) Siete DoF

Figura 5: Precisión del modelo CollisionGP para $\beta = 0{,}5$ con diferentes números de puntos de inducción para el robot de siete DoF en quince escenarios diferentes con cuatro obstáculos estáticos.

Para ilustrar el rendimiento tanto de Fastron como de CollisionGP en el C-space, proporcionamos un ejemplo con los dos primeros DoFs del robot, ya que una visualización en dos dimensiones permite una comparación cualitativa. La Figura 6 muestra el C-space obtenido para Fastron y CollisionGP, y los mapas de media y varianza obtenidos para el modelo CollisionGP con 128 puntos de inducción al accionar los dos primeros DoFs del robot. Los C-spaces obtenidos con CollisionGP y Fastron son muy similares a la realidad, aunque CollisionGP utiliza menos puntos de inducción y también proporciona una medida de incertidumbre, lo que resulta en el resaltado de los bordes de los obstáculos. Dado que la mayoría de los manipuladores actuales contienen articulaciones de revolución, el C-space es suave incluso para obstáculos no suaves y puede aproximarse bien mediante el modelo GP.

4.4. *Precisión, TPR y TNR*

La Figura 7 muestra comparaciones de precisión, TPR y TNR entre CollisionGP y Fastron en los quince entornos simulados. Uno de estos entornos se muestra en la Figura 6. Ambos métodos obtienen resultados similares en el caso de dos DoFs, pero CollisionGP mantiene la precisión, TPR y TNR para el caso de siete DoFs, mientras que los valores de Fastron disminuyen a medida que aumenta el número de DoFs.



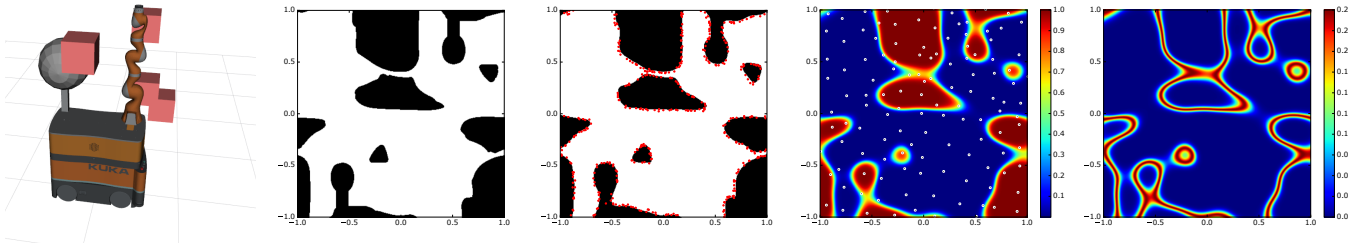
(a) Dos DoF (b) Siete DoF

Figura 7: Precisión, TPR y TNR para Fastron y CollisionGP en quince escenarios diferentes con cuatro obstáculos estáticos.

La Tabla 1 muestra comparaciones del tiempo de consulta, el tiempo de entrenamiento y el número de puntos de inducción entre CollisionGP y Fastron. CollisionGP es el método más ligero y tiene los tiempos de consulta más rápidos de ambos métodos para todos los DoF, mientras que Fastron consigue los tiempos de entrenamiento más rápidos. La implementación en la GPU es significativamente más rápida que en la CPU. Los tiempos de consulta de CollisionGP se consideran cuando se predicen 10000 configuraciones. Evaluamos la significación de los resultados con la prueba T de dos muestras para las medidas comparativas con siete DoF. Los valores p resultantes son todos inferiores a 10^{-7}, lo que demuestra una alta significación estadística.

Tabla 1: Rendimiento de CollisionGP y Fastron en quince escenarios diferentes con cuatro obstáculos estáticos, con la detección de colisiones real calculada mediante FCL.

	GP (GPU)	**GP (CPU)**	**Fastron**
Dos DoF			
m	**128**	**128**	298,64 ± 49,6
Consulta (μs)	**2,150 ± 0,76**	61,89 ± 1,01	76,62 ± 3,59
Entrenamiento (s)	0,446 ± 0,07	0,496 ± 0,04	**0,140 ± 0,07**
Precisión	91,59 ± 3,57	91,59 ± 3,57	**98,97 ± 0,35**
Siete DoF			
m	**512**	**512**	2917,80 ± 563,34
Consulta (μs)	**4,724 ± 0,32**	75,91 ± 1,22	91,02 ± 4,44
Entrenamiento (s)	0,684 ± 0,07	1,988 ± 0,02	**0,254 ± 0,05**
Precisión	**84,47 ± 3,92**	**84,47 ± 3,92**	66,88 ± 4,97



(a) Entorno (b) FCL (c) Fastron (d) Media GP (e) Varianza GP

Figura 6: Ejemplo de entorno y aproximaciones al C-space utilizando Fastron y el método de clasificación GP propuesto. La columna (d) muestra los puntos de inducción del GP.

5. Conclusiones

En este trabajo, presentamos y validamos el algoritmo CollisionGP, que modela el espacio de configuraciones para la detección de colisiones. CollisionGP permite realizar detecciones de colisiones un orden de magnitud más rápido que los algoritmos tradicionales de comprobación de colisiones, como FCL, cuando se utilizan lotes. También demostramos que CollisionGP es más ligero que Fastron, un algoritmo de aprendizaje automático del estado del arte para la comprobación de colisiones, consiguiendo predicciones más rápidas. Además, hasta donde sabemos, CollisionGP es el primer algoritmo de comprobación de colisiones que utiliza GPs, proporcionando no sólo una estimación media, sino también la varianza predictiva, que da una medida de incertidumbre sobre la predicción y es útil para eliminar falsos negativos. Esto es especialmente importante para aplicaciones críticas en las que la trayectoria final podría comprobarse con FCL para garantizar que no se producen colisiones.

El uso de GPs con variables auxiliares PG para la comprobación probabilística de colisiones abre algunas líneas interesantes de trabajo futuro en robótica. Una posible aplicación futura de CollisionGP es para robots con muchos DoFs, como los robots humanoides. Los grandes C-spaces requieren la exploración de muchos estados y, por tanto, operaciones primitivas rápidas, como la comprobación de colisiones. Otro trabajo futuro es la implementación de CollisionGP con diferentes planificadores de trayectorias por lotes en entornos reales para probar su eficiencia. El código de CollisionGP se puede encontrar en https://github.com/jmunozmendi/CollisionGP.

Agradecimientos

Investigación financiada por la comisión europea Innovation and Networks, a través del proyecto LABYRINTH-H2020. Acuerdo de subvención H2020-MG-2019-TwoStages-861696. Además, la investigación ha recibido financiación de RoboCity2030-DIH-CM, Madrid Robotics Digital Innovation Hub, S2018/NMT-4331, fundada por "Programas de Actividades I+D en la Comunidad de Madrid" y confundada por los Fondos Sociales Europeos (FSE) de la UE.

Referencias

Arslan, O., Tsiotras, P., 2015. Machine learning guided exploration for sampling-based motion planning algorithms. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS). pp. 2646–2652.

Chase Kew, J., Ichter, B., Bandari, M., Lee, T.-W. E., Faust, A., 2020. Neural collision clearance estimator for batched motion planning. In: Int. Workshop on the Algorithmic Foundations of Robotics. Springer, pp. 73–89.

Choset, H., 2005. Principles of robot motion: Theory, algorithms, and implementations. MIT press.

Chou, G., Wang, H., Berenson, D., 2022. Gaussian process constraint learning for scalable chance-constrained motion planning from demonstrations. IEEE Robotics and Automation Letters 7 (2), 3827–3834.

Danielczuk, M., Mousavian, A., Eppner, C., Fox, D., 2021. Object rearrangement using learned implicit collision functions. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA). pp. 6010–6017.

Das, N., Yip, M., 2020. Learning-based proxy collision detection for robot motion planning applications. IEEE Trans. on Robotics 36 (4), 1096–1114.
DOI: 10.1109/TRO.2020.2974094

Dong, J., Mukadam, M., Dellaert, F., Boots, B., 2016. Motion planning as probabilistic inference using gaussian processes and factor graphs. In: Robotics: Science and Systems. Vol. 12.

Elbanhawi, M., Simic, M., 2014. Sampling-based robot motion planning: A review. IEEE Access 2, 56–77.

Gardner, J., Pleiss, G., Weinberger, K., Bindel, D., Wilson, A., 2018. GPyTorch: Blackbox matrix-matrix gaussian process inference with GPU acceleration. Adv. in Neural Information Processing Systems 31.

Hensman, J., Matthews, A., Ghahramani, Z., 2015. Scalable variational Gaussian process classification. In: Artificial Intelligence and Statistics. PMLR, pp. 351–360.

MacKay, D. J., 1995. Probable networks and plausible predictions-a review of practical bayesian methods for supervised neural networks. Network: Computation in Neural Systems 6 (3), 469.

Mukadam, M., Yan, X., Boots, B., 2016. Gaussian process motion planning. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA). pp. 9–15.

Pan, J., Chitta, S., Manocha, D., 2012. FCL: A general purpose library for collision and proximity queries. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA). pp. 3859–3866.

Pan, J., Manocha, D., 2015. Efficient configuration space construction and optimization for motion planning. Engineering 1, 46–57.

Pan, J., Manocha, D., 2016. Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing. Int. J. Robotics Research 35 (12), 1477–1496.
DOI: 10.1177/0278364916640908

Polson, N. G., Scott, J. G., Windle, J., 2013. Bayesian inference for logistic models using pólya–gamma latent variables. J. of the American statistical Association 108 (504), 1339–1349.

Qureshi, A. H., Simeonov, A., Bency, M. J., Yip, M. C., 2019. Motion planning networks. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA). pp. 2118–2124.
DOI: 10.1109/ICRA.2019.8793889

Wenzel, F., Galy-Fajou, T., Donner, C., Kloft, M., Opper, M., 2019. Efficient gaussian process classification using pòlya-gamma data augmentation. In: Proc. AAAI Conf. Artificial Intelligence. Vol. 33. pp. 5417–5424.