Julia Caso de Estudio: Locomoción de Robots

Sabemos que los robots no son solo algo de películas. Estos ya son una realidad y están entre nosotros. Debido a esto, hay muchas empresas dedicadas a su fabricación.

El Grupo de Locomoción de Robots del MIT programa a los robots con patas para que caminen y puedan explorar regiones remotas o viajar a lugares que no son seguros para los humanos, como campos de minas y sitios contaminados.

Controlar un robot con docenas de articulaciones requiere algoritmos de optimización, teoría de control y aprendizaje automático, y la implementación de esos algoritmos requiere un software eficiente. Es por eso que los robotistas Robin Deits y Twan Koolen eligieron a Julia.

Y claro! Durante todo el trabajo vimos como Julia es exageradamente más eficiente que muchos otros lenguajes. Con su capacidad para realizar algoritmos de machine learning, por ejemplo, esto es ideal para la locomoción.

“En comparación con C ++, descubrimos que Julia es mucho más productiva, al tiempo que conserva un rendimiento similar. Percibimos que las principales razones de este aumento de productividad son las siguientes:

- Los proyectos de C ++ suelen tener una gran sobrecarga debido a la configuración y el mantenimiento de un sistema de compilación, ya que no existe un sistema de administración de paquetes estándar en C ++. Este problema se agrava al trabajar con dependencias externas, cada una de las cuales tiene su propio sistema de compilación personalizado. Entonces, la elección a menudo se convierte en: ¿realmente quiero dedicar tiempo a trabajar en un sistema de construcción que integre una dependencia potencial en mi proyecto, o reinventar la rueda? Julia es capaz de evitar esto debido a su sistema de paquetes estandarizado, que... es mucho mejor que atravesar el salvaje oeste de los sistemas de compilación C ++. El sistema de paquete estandarizado también facilita la configuración de la integración continua básica, que se reconoce cada vez más como una necesidad en el campo de la robótica. Además, creemos firmemente que los investigadores de robótica deberían publicar su código con sus artículos para mejorar la reproducibilidad y quizás la reutilización del código, y nuevamente, el sistema de paquetes de Julia facilita la distribución de dicho código.
- Julia evita muchos errores sutiles al tener un comportamiento menos indefinido y / o definido por la implementación de C ++. Estos bugs tienden a tardar mucho en ser detectados. Otra clase de errores (y malos patrones de diseño) es eliminada por la elección de Julia de no permitir punteros nulos (o referencias).
- Julia no requiere que pases tanto tiempo pensando en la vida útil de los objetos. No hay necesidad de preocuparse por elementos básicos de C ++ como la 'regla de tres' o la 'regla de cinco', o preocuparse por si otros colaboradores se adhirieron a esas reglas.
- Julia facilita la compatibilidad con múltiples plataformas, lo que permite a los usuarios, por ejemplo, simular el algoritmo de control en el sistema operativo que elijan antes de ejecutar el código en un robot real.

“Una última ventaja de Julia para nuestro trabajo es su excelente interoperabilidad con el código C y Python existente. Existe una gran cantidad de software para robótica y optimización escrito en C y Python, y la funcionalidad ccall incorporada de Julia y el paquete PyCall.jl facilitan la interacción con las herramientas existentes. Por ejemplo, confiamos en la biblioteca LCM (Lightweight Communications and Marshalling) para la comunicación de bajo nivel entre nuestros robots y sus controladores. En lugar de reinventar todo ese código, pudimos exponer fácilmente la biblioteca C existente a Julia con poca o ninguna sobrecarga. Aprovechar las herramientas de C existentes como LCM nos da más tiempo para concentrarnos en las partes importantes de nuestra investigación ".

Como vemos, todas las maravillas del lenguaje que vimos en el trabajo fueron correctamente aplicadas a un área tan compleja como es la robótica. Su velocidad y eficiencia son clave y este lenguaje tiene un futuro increíble.