1
ROS 入門

今日足以改變世界的科技之一就是機器人技術。機器人在很多面向都可以取代人類，以至於我們很擔心機器人是否會搶走飯碗。不過有件事是確定的，機器人技術一定是未來極具影響力的科技。當一項新技術問世時，相關領域的機會也大大增加。這代表機器人與自動化在不久的將來會產生數以千計的工作機會。

機器人技術中能提供大量工作機會的主要領域之一，就是機器人的相關軟體開發。咱們都知道，軟體能賦予機器人或任何機器生命。機器人的效能也可透過軟體延伸。機器人的各種能力，像是控制、感測與智能，都是透過軟體來達成的。

機器人的軟體是許多相關科技的組合，像是電腦視覺、人工智慧以及控制理論。簡言之，開發機器人的軟體並非易事，它需要許多領域的專業知識。

如果您想要開發 iOS 或 Android 的手機應用程式，有現成的**軟體開發套件**（**Software Development Kit, SDK**）可讓您在其上來開發應用程式，但機器人怎麼辦呢？有什麼通用性的軟體框架嗎？當然有。諸多熱門機器人軟體框架其中一款就是**機器人作業系統**（**Robot Operating System, ROS**）。

本章要瞧瞧 ROS 的抽象概念和安裝步驟，以及概述模擬器及其在虛擬系統上的使用。然後，涉及 ROS 的基本概念，以及支援 ROS 的不同機器人、感測器及致動器。也瞧瞧產業及研究機構的 ROS。本書完全是為了 ROS 專題而生，所以本章就是您各個專題的入門指南，可幫助您順利設定 ROS。

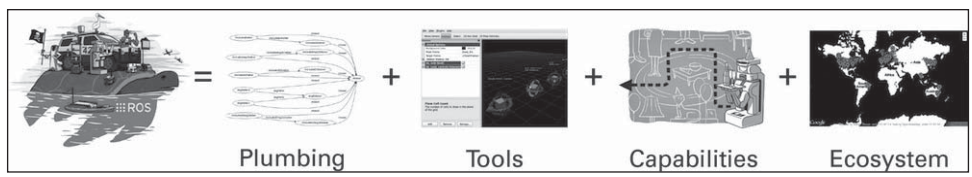
ROS 入門

ROS 是一款用於開發機器人程式的開放原始碼軟體框架。ROS 提供了硬體抽象層，開發者不必煩惱底層的硬體就可開發各種機器人應用程式。ROS 也提供了多種軟體工具來視覺化呈現機器人的資料，也能除錯。ROS 框架的核心是一個通訊中介軟體，各程序可以彼此通訊並交換資料，就算執行在不同的機器上也可以。不論是同步與非同步通訊模式，ROS 都支援。

ROS 中的軟體都是以套件（package）的形式來管理，模組化與可再用性都很不錯。透過傳遞 ROS 訊息的中介軟體與硬體抽象層，開發者就能開發千千萬萬種機器人功能，例如建圖與導航（針對移動式機器人）。幾乎所有的 ROS 功能都是獨立於機器人硬體之上，因此各種型號的機器人都能運用它。新款的機器人甚至不需要修改套件內的任何程式碼就能直接使用某一功能套件。

ROS 與許多大學與有貢獻的開發者都有密切合作。甚至可以說 ROS 是個由社群所驅動的專題，並且受到世界上諸多開發者的大力支援。就是這樣活躍的開發者生態系統讓 ROS 與其他機器人框架有所不同。

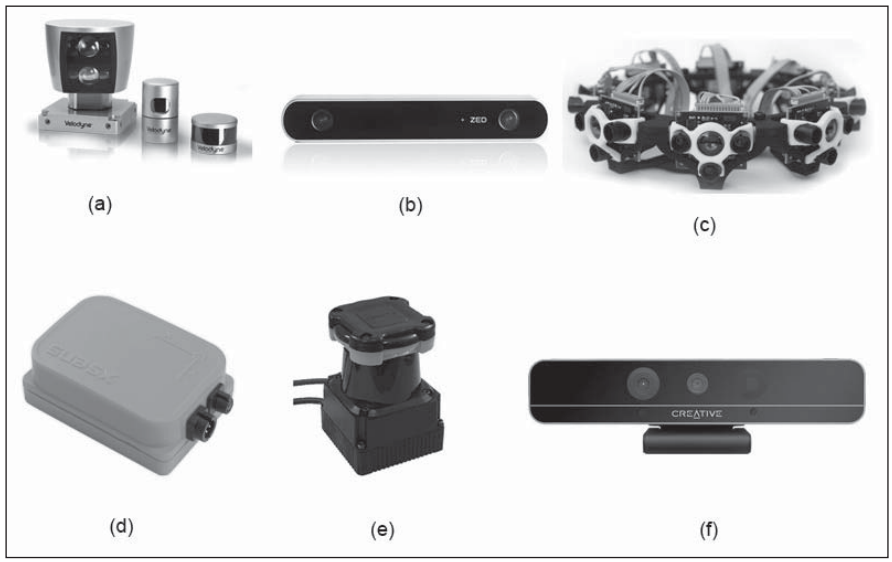
簡言之，ROS 是通訊機制、工具、功能及生態系的組合。如下圖所示：



ROS 方程式（來源：ros.org，採用 Creative Commons CC-BY-3.0 授權）

ROS 專題是以 Switchyard 這個名字在 2007 於美國史丹佛大學發起。到了 2008，開發由 Willow Garage 這家機器人研究新創公司所接手。ROS 的主要開發都是 Willow Garage 負責。到了 2013，Willow Garage 的研究員成立了開放原始碼機器人基金會（Open Source Robotics Foundation, OSRF）。ROS 現在由 OSRF 積極維護。現在，讓我們來看看一些 ROS 版本。

下圖是支援 ROS 的常用感測器：



支援 ROS 的常用機器人感測器

上圖中的感測器名稱分別是 Velodyne(a)、ZED Camera(b)、Teraranger(c)、Xsens(d)、Hokuyo 雷射測距儀 (e) 以及 Intel RealSense (f)。

支援 ROS 的感測器清單請參考：*http://wiki.ros.org/Sensors*。

這些感測器的 ROS wiki 頁面如下：

- **Velodyne (a)**：*http://wiki.ros.org/velodyne*
- **ZED Camera (b)**：*http://wiki.ros.org/zed-ros-wrapper*
- **Teraranger (c)** ：*http://wiki.ros.org/teraranger*
- **Xsens (d)**：*http://wiki.ros.org/xsens_driver*
- **Hokuyo 雷射測距儀 (e)**：*http://wiki.ros.org/hokuyo_node*
- **Intel real sense (f)**：*http://wiki.ros.org/realsense_camera*

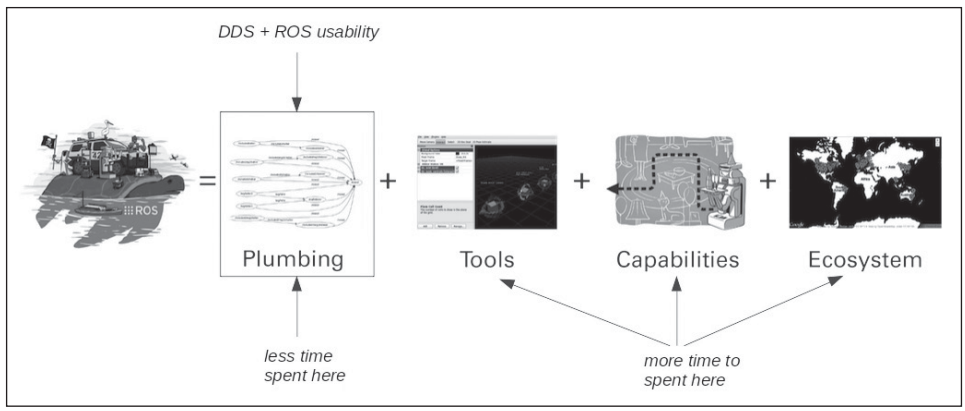
現在，讓我們來看看使用 ROS 的優點。

開始使用 ROS-2

如果談到用於即時系統和生產就緒之解決方案的通訊網路框架，ROS-2 可說是決定性的改良關鍵。ROS-2 希望能做到以下幾點：

- 提供元件之間的安全通訊
- 即時通訊
- 易於連接多個機器人
- 不論那種通訊媒體，都可改善通訊品質
- 直接裝設 ROS 層於硬體上，例如感測器及嵌入式板
- 使用最新的改良軟體

記得前一章的 ROS 方程式示意圖嗎？ ROS-2 也遵循相同的方程式但是方式稍微不一樣：



有 DDS 實作的 ROS 方程式（來源：ros.org。Creative Commons CC-BY-3.0 授權：*https://creativecommons.org/licenses/by/3.0/us/legalcode*）

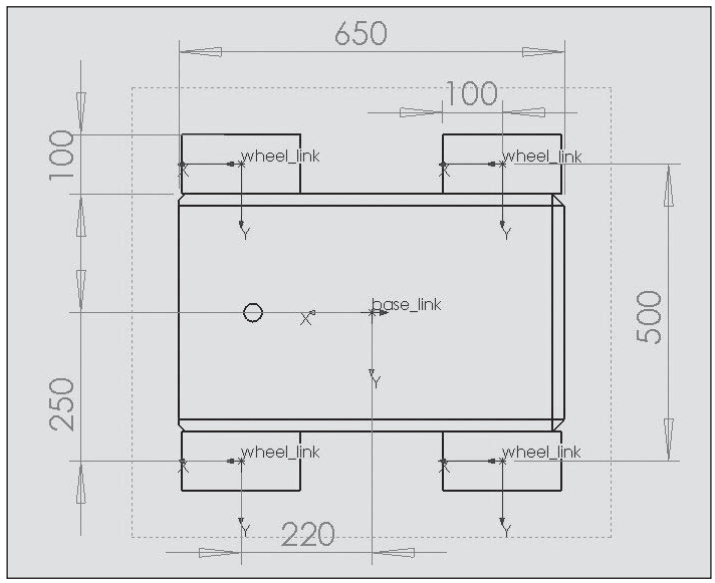
3
建置工業用移動機械手

機器人技術為當今廣泛應用的跨學科工程領域。不像移動及固定機器人的標準分類，它們也會以移動機器人、空中機器人、海上及太空機器人的應用來分類。最常見的機器人是在環境中企圖了解及學習其變化而四處移動的移動機器人。它們可以共用環境，並執行應用程式所要求的必要動作。在工業有巨大需求的另一類常用固定式機器人是機械手臂。它們起初用來實行舉起笨重負載，這些通常都是高度重複的機械性任務。但是機械手的智慧已成長足以了解環境及其中的感興趣物件且與人類一起執行任務。這類型的機械手臂現在被稱為 cobot。想像把機械手臂和有足夠負載能力的移動機器人組裝在一起，此類機器人通稱為移動機械手（mobile manipulators）。

移動機械手在工業很有用，因為它們可帶著物體四處走動且按需要輸送到各個工作單元或工作站。不像機械手臂的工作容積多有受限，這讓機器人幾乎可自由觸及空間中的所有定點。這對大部份產業為了工廠自動化而遵循的彈性管理系統是極佳的附加功能。除了工業之外也可用在其他領域，例如太空探索、採礦、零售業和軍事應用。基於應用需要，它們可為自主控制型、半自主控制型或手動控制型。

這是開始使用 ROS 來建置機器人的第一章。會介紹市售移動機械手的清單，好讓您更了解如何使用它們。然後，查看建置移動機械手專題的先決條件及辦法。此外，可個別建模及模擬機器人基座及機械手臂，然後把它們組裝起來並觀察它們的動作。

我們的機器人基座建模成具備底盤及四個輪子，下圖呈現各連結的資訊：



標示出連結 / 座標系統資訊的機器人

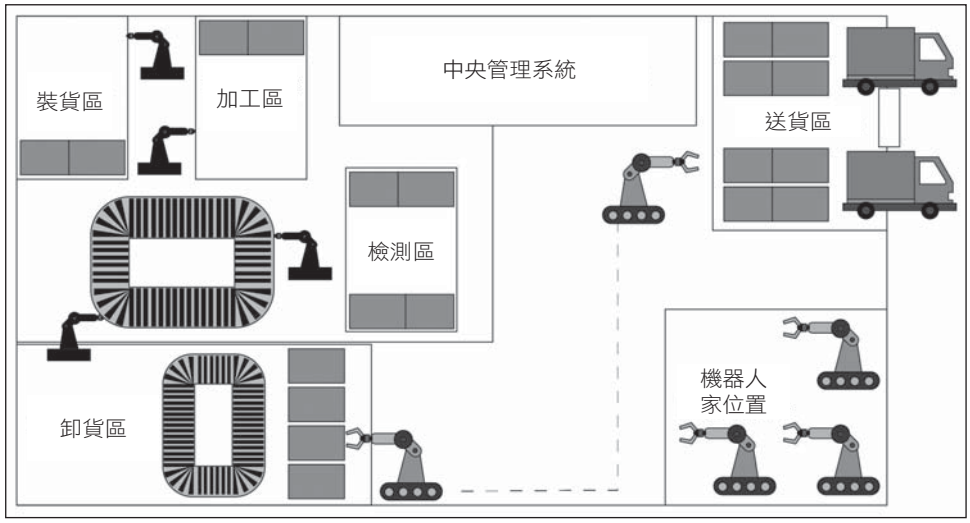
底盤取名為 `base_link` 且在其中心可看到座標系統。輪子（或 `wheel_frames`）則是相對於 `base_link` 框架來安放。您可看見，按照 REP，模型的座標系統遵循右手法則。您現在可明白，機器人的正向永遠朝向 x 軸，而機器人的旋轉是以 z 軸為中心。另外也請注意，輪子是相對於參考框架來以 y 軸為中心旋轉（下一節會在程式碼中看到這個參考）。

初始化工作空間

現在需要定義用於機器人的網格為 `<link>` 及 `<joint>` 標籤。網格檔請由此取得：*https://github.com/PacktPublishing/ROS-Robotics-Projects-SecondEdition/tree/master/chapter_3_ws/src/robot_description/meshes*。請根據以下步驟來初始化工作空間：

1. 建立 ROS 套件並加入檔案。在新終端機中使用以下指令建立套件：

```
$ initros1
$ mkdir -p ~/chapter3_ws/src
```



工業機器人案例

假設有多台像是第三章「建置工業移動機械手」中的這類移動機器手，它們要協同運作來裝卸貨物，而至少需要五台機器人才能完成這件事。再者，另外還有幾台機器手臂與負責管理的機台一起工作，例如把貨物放上加工區、放上輸送帶，最後取下貨物來配送。最後，還需要一台中央系統來監控所有機器人的任務及健康狀況。對於這樣的設定，會有多個區域網路需要連接至公共網路（中央系統所在處）。

ROS 提供了一個機制，可透過公共 ROS 工具讓多個 ROS 子系統彼此交換資訊。這個方案就是 `multimaster_fkie` 這款 ROS 套件。

multimaster_fkie 套件簡介

`multimaster_fkie` 套件有兩個主要節點：

- `master_discovery` 節點
- `master_sync` 節點

`master_discovery` 節點可偵測網路中的其他 ROS 主端，並識別網路是否發生任何變動，如果有變動就會分享給其他 ROS 主端。`master_sync` 節點可讓其他

ESP8266 板子的尺寸型態相當多元，並各自有不同的 SDK（軟體開發套件）。例如，在 ESP8266 SoC 上就可執行 NodeMCU 這款以 Lua 腳本語言為基礎的韌體。

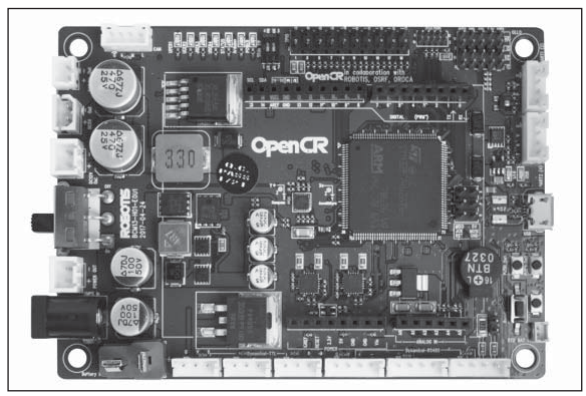
現在，來看看一些支援 ROS 的嵌入式開發板。

ROS支援的嵌入式開發板

ROS社群推出了一些有趣的嵌入式開發板，可直接整合各種感測器及致動器，並提供基於 ROS 的訊息來與其他 ROS 元件介接。接著要介紹的幾款板子包含 Robotis 公司的 OpenCR 與 Vanadium labs 公司的 Arbotix-Pro 控制器。

OpenCR

OpenCR，或所謂開放原始碼控制模組（control module），是基於 STM32F7 系列晶片的嵌入式開發板，具備可進行浮點運算的超強 ARM Cortex-M7。它有板載的 IMU（MPU9250），18 支 GPIO 腳位、6 個 PWM I/O 腳位與各種通訊埠，包含 USB、幾個 UART、3 個 RS485 與 CAN 埠，以及相容於 Arduino 板的 32 腳接頭，可以搭配各種 Arduino 擴充板。OpenCR 板如下圖：



OpenCR 開發板（來源：*http://wiki.ros.org/opencr*，影像：ros.org，Creative Commons CC-BY-3.0 授權）

D. 請注意，在此不會出現任何圖形化使用者介面，因此需要安裝輕量級的 GUI 讓後續操作更方便。請用以下指令安裝 LDXE desktop：

```
$ sudo apt-get install lxde
```

E. 請記住，如果要從 micro SD 而不是 eMMC 啟動，開機時請都要按著開機按鈕，直到 LED 開始閃爍。

3. 正如在 " 安裝 Armbian 和 ROS" 這段中所述，安裝 ROS 非常簡單。請回顧該段的所有步驟即可。

現在，來看看如何在 Raspberry Pi 3/4 上設置 ROS。

在 Raspberry Pi 3/4 上設置 ROS

Raspberry Pi 3/4 的作業系統設定方式相當類似也很方便。由於使用者眾多與應用層面廣泛，Raspberry Pi 支援了一系列的作業系統。本書將介紹如何設置機器人專題常用的作業系統：Raspbian 與 Ubuntu MATE。首先來看看前置條件。

前置條件

為了在這片單板電腦上設定作業系統，請先準備以下硬體：

- SD 卡
- 額定電源供應器
- 避免靜電接觸的外殼，但非必須

在軟體要求方面，假設您使用 Ubuntu 筆記電腦，您會需要稱為 Etcher 的開放原始碼的作業系統映像檔燒錄程式。請由此下載：*https://www.balena.io/etcher/*。它不需要安裝，因此下載後請將檔案解壓縮到適當路徑，再執行應用程式就好。接著介紹如何在 Tinkerboard 上的 Debian 與 Armbian 作業系統上設置 ROS。現在，讓我們學習安裝 Raspbian 與 ROS。

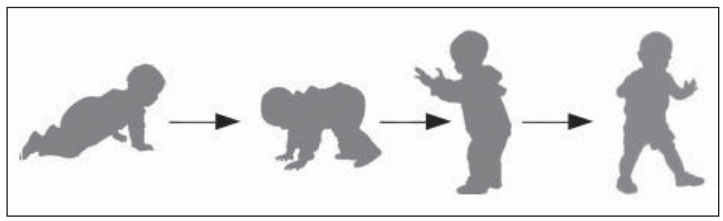
強化學習常見於遊戲領域，其中電腦會試著去了解使用者的動作，並做出適當的回應來確保它會贏。強化學習也在自駕車及機器人技術中被大量測試。由於本章將特別聚焦在機器人技術，因此也會更深入探討這項學習技術。如果機器已充分了解與其互動的環境，強化學習可變成監督式學習。後續在 " 探索與利用 " 這一段會更深入討論這個觀點。

現在，讓我們進一步深入本章的主要概念：強化學習。

認識強化學習

如前所述，強化學習是一種邊做邊學的學習技術。在此用一個簡單的比喻來理解強化學習：一個九個月大的小寶寶試圖起身行走。

下圖是這個比喻：



小寶寶行走比喻

小寶寶要做的第一步是試著把雙腿壓向地面來起身，然後平衡自己並站穩。如果成功，您會在小寶寶的臉上看到微笑。現在，小寶寶向前邁出了一步，並試著再次保持平衡。如果在過程中失去了平衡並跌倒，則小寶寶可能會皺眉或哭泣。如果沒有行走動機，小寶寶則可能會放棄行走，但者可能再次起身行走。如果小寶寶成功向前邁出了兩步，您可能會看到他展現燦爛的笑容以及發出一些表示快樂的聲音。走的步數愈多，小寶寶就會變得愈有信心，最終就會一直走路，有時甚至能跑起來：

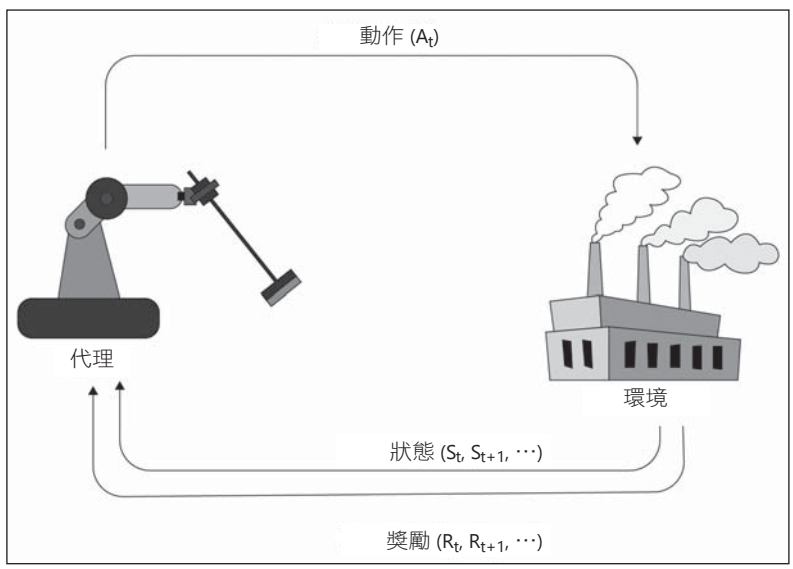
建議利用。如果代理不必擔心獎勵增加所花費的時間，反而希望在整個環境中多多嘗試的話，則建議探索。強化學習有一些用於決定探索或利用的演算法，這可用著名的多臂吃角子老虎機問題解決。

強化學習公式

現在逐一來介紹強化學習包含的重要元件：

- **代理**：代理可執行必要的步驟（或動作）來與環境互動。
- **環境**：環境為代理進行互動的對象。環境類型包含確定性型、隨機型、完全可觀察型、連續型、離散型與其他等等。
- **狀態**：狀態為代理當下展現或駐留於其中的狀況或位置。狀態通常為代理執行特定動作之後的結果。
- **動作**：代理與環境互動時所做的事情。
- **獎勵**：代理執行某個動作來遷移到某個狀態的結果。獎勵可為正面或負面。

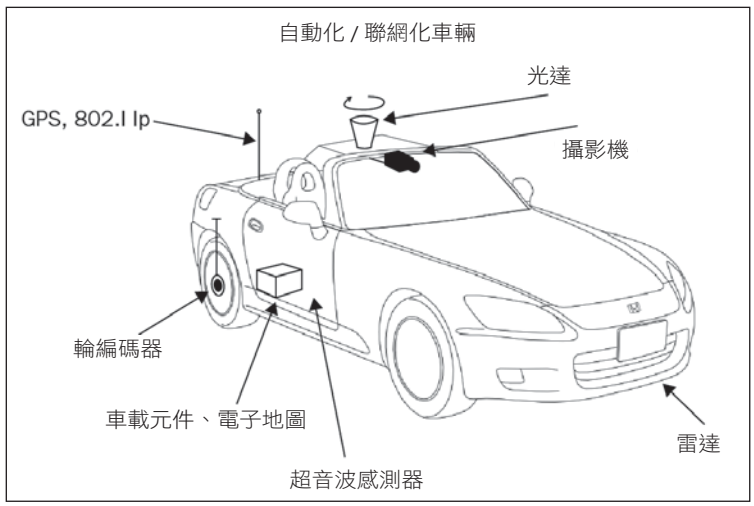
強化學習問題請參考下圖：



代理與環境 - 強化學習示意圖

典型自駕車的元件

下圖標示了自駕車的各個重要元件。本段將討論這些零件與其功能。另外也會介紹 DARPA Challenge 的參賽自駕車究竟用到了哪些感測器：



自駕車的重要元件

從 GPS、IMU 與車輪編碼器這些重要元件開始吧。

GPS、IMU 與車輪編碼器

如您所知，全球定位系統（GPS）可運用 GPS 衛星來判定車輛的位置。車輛的經緯度可從 GPS 資料算出來。GPS 的準確度會隨著感測器的類型而不同，有些感測器的誤差範圍為數公尺，有些則不到一公尺。結合 GPS、慣性測量單元（IMU）與車輛里程資料，再加上感測器融合演算法，就能判定車輛狀態。對於車輛的估計結果也會更佳。來看看 DARPA Challenge 2007 所用的位置估計模組。

史丹佛大學的自駕車，Junior，採用了 Applanix 公司的 POS LV 模組，該模組整合了 GPS、IMU 與車輪編碼器（或測距指示器，DMI）。更多資訊請參考：*http://www.applanix.com/products/poslv.htm*。