無線感測網路具克服障礙物之佈建演算法

謝震宇
淡江大學
avinson83@wireless.cs.tku.edu.tw

陳昱价
淡江大學
ycchen@wireless.cs.tku.edu.tw

張志勇
淡江大學
cychang@mail.tku.edu.tw

***摘要*—無線感測網路(wireless sensor networks)中，良好的網路佈建方式，使機器人有效率地達到完全覆蓋及省電等目的，是無線感測網路中重要的議題。現存的佈建方法大都容易受到障礙物的影響，進而使機器人進入死巷或留下空洞。不同於現有的機器人佈建演算法，本論文所提出機器人佈建無線感測網路的演算法，僅以少量的記憶體成本讓機器人可以克服任何複雜的障礙物，並將適量的感應器佈建於監控區中以達到縮短佈建時間、節省感測器硬體成本及全區覆蓋等目的。此外，機器人僅需與已佈建的感測器進行少量通訊，使大多數已被佈建的感測器可進入省電狀態。實驗顯示，本論文所提出的演算法具有高佈點率、低電力成本，且可以在複雜的監測區域中達到全區覆蓋。**

***關鍵詞*—佈點、感測網路、機器人、克障、死路(dead-end)**

一、簡介

無線感測網路是由大量的感測器結點所構成，其中，每個感測器均具有嵌入式處理器、記憶體、與微小的感測元件等設備，這些微小、成本低廉且由電池供電的感測器，被大量地放置於特定區域，作為環境偵測或軍事作戰監控等用途。在網路佈建的議題上，佈建大量的感測器 將使硬體成本提高，而佈建少量的感測器 卻可能導致整個感測範圍無法覆蓋監控區域的面積，並使監控區域出現感測空洞。為達到節省硬體成本及感測範圍全區覆蓋之雙重目的，良好的網路佈建方法將是重要的研究議題。現在已經提出的佈點方法大致可區分為隨機佈建、行動感測器佈建以及機器人佈建三類。

將大量的感測器以隨機佈建的方式，佈置於欲監測範圍內是最簡易的佈點方法，然而，隨機佈建的方式較易造成感測器佈建不均，導致某區域的感測器過於稀疏或密集，分別造成感測空洞或浪費過多的感測器等問題。利用行動感測器的移動性雖可以調整感測器的位置，以改善佈點過於稀疏或密集的問題，但行動感測器的硬體成本及耗電量將分別對佈建的數量及網路生命期產生嚴苛的限制。利用機器人來佈建感測器，除了可以採用規則的方式來佈建感測器外，更能使用適量的感測器來達到完全覆蓋欲監測區域之目的，並可藉由機器人到達不適合人為佈點的地方，諸如有毒氣、化學污染等區域佈點[1][4][5]。

以機器人佈建無線感測網路，在近年來已成為一熱門的研究議題，在眾多的研究中，大多以覆蓋率、佈建的感測器數量以及佈建時間或機器人行走的路徑長度為佈建演算法優劣的標準。當監控區域有障礙物存在時，將會影響到機器人佈點的效率與感測器感測的覆蓋率，機器人有可能走進死路中無法繼續佈點，監控區域也會因此而產生機器人佈點不完全的空洞問題。現存關於機器人佈點的相關研究[6][7]中，仍無法克服空洞及死路的問題。本論文利用攜帶感測器的機器人來佈建監控區，以最有效率的方式避開障礙物，並在較短的時間內，以適量的感測器達到全區覆蓋的目的。

本論文後續的章節安排如下，在第二章中將介紹相關研究，並討論其作法可供精進之處;第三章將介紹本論文的假設及待解決的問題，以及本論文的貢獻;第四章中將介紹在無障礙物環境下，本論文所提出的機器人佈點演算法；第五章將介紹本論文所提出的克服障礙物佈點技術以及關於感測器省電的機制;第六章以實驗來檢視本論文相較於以往研究的效能改善程度。最後則是本論文的結論。

二、相關研究

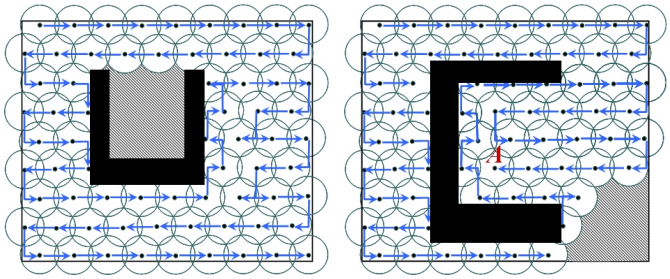
以機器人佈建無線感測網路，在近年來已成為一熱門的研究議題，在眾多的研究中，大多以覆蓋率、佈建的感測器數量以及佈建時間或機器

人行走的路徑長度為佈建演算法優劣的標準。在[1][2][3]的研究中，機器人依南西北東四個方向的優先權逐步移動並佈置感測器，在不需要位置資訊的條件下，使用已佈下的感測器來引導機器人在欲監測環境中行走與佈點，當機器人其通訊範圍內不存在任何一個感測器時，機器人即佈置新的感測器。而在機器人之通訊範圍內的感測器，能夠區域性地建議機器人最適當的方向行進，機器人將這些感測器的建議結合起來並選擇出一個最適當的方向進行探測，亦可不受障礙物的阻擋而在無線感測網路中完成佈點的工作，然而，當機器人遇到障礙物時，將可能造成空洞問題。

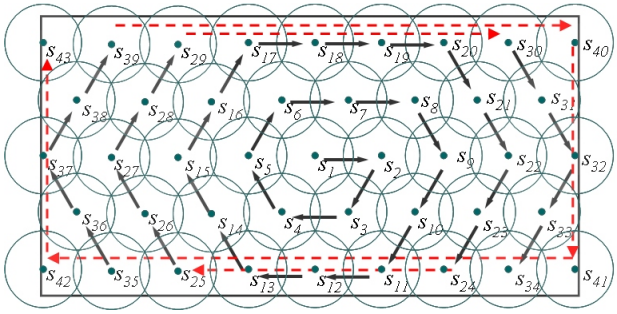
在[6]的研究中，機器人採用蛇行的方式佈點，機器人可以避開簡單的障礙物，且解決了在遇到障礙物或邊界時可能產生的空洞問題，但當遇到較複雜的障礙物或是上凹型的障礙物時，其所提出的佈建方法便可能出現空洞問題，如圖(一)(a)所示，由於機器人採蛇形佈點，自左而右、自上而下方式佈建感測器，遇上凹型障礙物時，將認為蛇形佈點在後續行進時可佈建感測器，因此不與理會而造成空洞，如圖(一)(a)中的灰色區域；由於佈建後之感測器 亦視為障礙物，如圖(一)(b)所示，當機器人行進至*A*點時，便進入死路，如此一來機器人必需耗費大量的電量、時間及記憶體等成本，才可以走出死路繼續佈建未完成的區域。此外，在[6]的研究中，對於已佈署過的感測器必須提供機器人資訊來幫助機器人順利的完成佈點的工作，因此，在佈點的過程中，已佈署的感測器均無法進入省電機制。

研究[7]假設機器人之初始的所在位置為監控區中心，並以螺旋的方式由內而外佈點。當遇到障礙物時，將進入一 “obstacle state”，緊貼著障礙物移動並包圍障礙物佈點，直到離開障礙物才回復為正常狀態，佈建在障礙物旁的感測器亦視為虛擬障礙物，然而當機器人遇見一些較複雜的障礙物時，仍會產生空洞問題，另一嚴重的問題如圖(一)(c)所示，機器人需要耗費大量的電能，額外繞行很長的虛線路徑，才可以達到全區覆蓋的目的。

研究[8]考慮有障礙物的環境，利用機器人探索及蒐集資訊，但機器人必須在一開始知道探測區域以及重要目標的資訊;論文中只利用機器人的八個方位與目標的相對關係判斷其行走的路徑，沒有考慮到機器人繞進障礙物時回程的方法，所以當目標位於複雜的障礙物地形中時，機器人會浪費大量的路徑，甚至無法自障礙物中走出，因此而停止探測的工作；而探測區域中有多重障礙物時，其耗費冗長的路徑以及受困於死巷的問題將更加嚴重。



(a)蛇行佈點[6]所會產生的空洞問題　(b)蛇行佈點[6]所會產生的死巷問題



(c)螺旋型佈點[7]會耗費多餘的路徑

圖(一) 機器人佈點問題

研究[9]中，機器人會把其佈點時所走過的路徑利用具有stack功能之記憶體紀錄下來，當機器人因為障礙物而遇到死路時，可以沿著記憶體中所儲存的路徑資訊由死路中走出。但如此將耗費大量的記憶成本;當遇到較複雜的障礙物時，機器人要走出死路將繞行冗長的路徑，機器人也因此要花費很多的時間及電量來處理死路問題。

在以上的相關研究中，我們可以發現，複雜的障礙物將影響機器人的佈點效率以及感測器的

覆蓋率，如何使機器人在障礙物未知的情況下，克服複雜的障礙物所引起的死路及空洞問題，並能有效率地完成佈點及達成良好的覆蓋率，是本論文所提出的演算法所欲達成的目標。

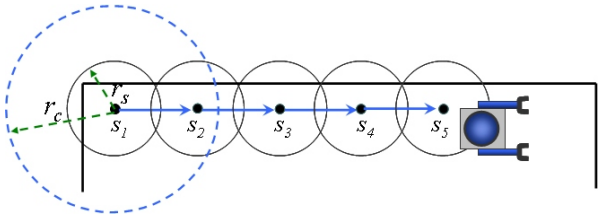
三、網路環境與問題描述

在本章節中，我們將介紹本論文的環境假設及問題描述。

(一) 網路環境

本論文的環境假設如下，機器人身上帶有靜態感測器，其攜帶之數量可佈滿整個欲監測之區域，假設通訊半徑 r_c 至少是感測半徑 r_s 的 $\sqrt{3}$ 倍。設S={ $s_1, s_2, \ldots, s_n$ }代表已被佈署完成感測器集合，其中 $s_1, s_2, \ldots, s_n$ 代表已被佈署的感測器，如圖(二)所示。

我們可以依機器人所攜帶的 GPS，佐以機器人輪子的轉速及其行進的時間計算出機器人移動的距離。機器人身上帶有 GPS 及超音波裝置可以測知地圖邊界及障礙物資訊。此外地圖的邊界以及已佈建感測器之區域都將視為虛擬的障礙物。



圖(二) 符號說明

(二) 待解決問題

由於感測網路的偵測效能取決於感測器所涵蓋的範圍，因此感測範圍覆蓋監測區域以避免感測空洞的產生，將是本論文的首要目標。感測網路中可監測的範圍C 將是所有感測器的感測範圍之聯集，如下列公式：

$$C = \bigcup_{1 \le i \le n} s_{i.coverage} \qquad (1)$$

設A為整個感測範圍的面積，$O_{.area}$ 為障礙物所佔的面積，我們必須讓監控區域中空洞的區域達到最小，如下式：

$$\text{Minimize}\left[A - O_{.area} - C\right] \qquad (2)$$

在另一方面，若所佈置的感測器點數過多，彼此之間感測範圍的重疊區域也較大，導致許多感測器負責偵測的區域重複，增加感測器的硬體成本及通訊時的耗電量。所以我們要在可以全區覆蓋的前提下，達到使感測範圍的重疊區域達到最小的目標，如下列公式所示。

$$\text{Minimize}\left[\sum_{1 \le i \le n} s_{i.coverage} - C\right] \qquad (3)$$

此外，機器人的行走路徑長度代表其佈建感測器的時間，總行走路徑越短，代表機器人佈點的效率越高，所以我們也必須盡量使機器人的總行走路徑*route*最小化，如下式(4)。

$$\text{Minimize}\left[route\right] \qquad (4)$$

四、Basic Deployment Mechanism

在本章節中將介紹機器人在無障礙物時的基本移動規則以及佈點規則。

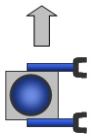
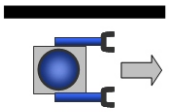
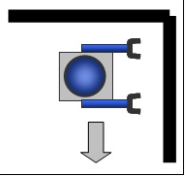
(一) Basic Movement Rule

在機器人行走的規則方面，我們以機器人所面對的方向為基準，令 *direction* 表示機器人行走的方向，其值可能為 *left*、*straight*、或*right*，分別表示左轉、直行及右轉。以下，我們將提出機器人行走的規則。

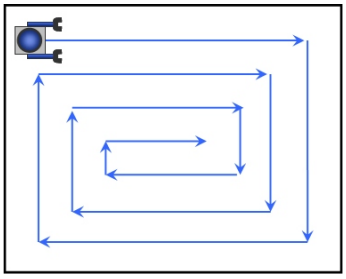
Movement Rule：令機器人行走的方向其最高優先權為左側邊，其次為機器人正前方，第三優先才是機器人右側邊方向，也就是說，*direction = left* 的優先權為最高，其次為*direction = straight*，最後才為*direction = right*。 **q**

如圖(三)(a)所示，機器人行進的方向*direction*將以機器人面對的方向為基準，如果可以左轉則向左轉，若不能左轉則向前行進，若左方及前方都無法行走，則向右轉。在這樣的移動法則下，若機器人一開始放置在欲監測地區的邊界，在區域沒有障礙物的情況下，機器人將會沿著邊界由外往內佈點如圖(三)(b)所示。若機器人一開始放置在非邊界區，則依照上述的移動法則，機器人將先以逆時針方向螺旋型的行走方式移動，直到遇到邊界時，才又回復由外往內的行走方式移動，如圖三(c)所示。

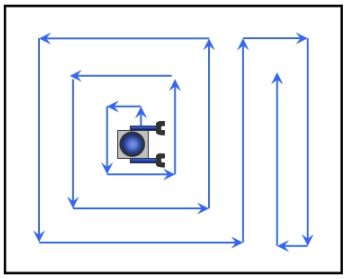
利用這個方法佈點，機器人會不斷的把尚未佈點的監測區域地形由外而內簡化、縮小，直到監測區域達到全區覆蓋。

Priority	**Priority 1**	**Priority 2**	**Priority 3**
direction	*left*	*straight*	*right*
Example			

(a) Basic Movement Rule



(b) 機器人起始位置位於地圖邊界之移動軌跡。



(c) 機器人起始位置位於非邊界區域之移動軌跡。

圖(三) Basic Movement Rule

(二) Basic Deployment Rule

由於感測網路的效能好壞取決於感測網路中感測器的覆蓋範圍，為了使佈建的感測器能達到全區覆蓋的目的並減少佈建感測器的數量，圖(四)(a)顯示了感測器能貢獻其最大的覆蓋範圍且不產生空洞的佈建結果，假設半徑 r_s 為感測器的感測半徑，r_c 為感測器的通訊半徑，在滿足 $r_c \geq \sqrt{3}r_s$ 的條件下，若感測網路中的感測器排列如圖(四)(a)所示，將能以最少量的感測器達到全區覆蓋的要求，並使每個感測器與其周圍的感測器皆可互相通訊。我們將利用此距離作為本論文設計佈點法則的依據。

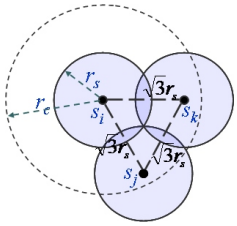
為了用最少的感測器數完成全區覆蓋，機器人必須依照上述規則佈建感測器。為實現此佈建規則，機器人的佈點分為兩個模式：Horizontal mode (*H* mode) 及Vertical mode (*V* mode)。在 *H* mode 中，機器人每前進 $\sqrt{3}r_s$ 佈建一個感測器；在 *V* mode 中，機器人會先前進 $3/2\,r_s$ 再依障礙物資訊決定左轉或右轉繼續行走 $\sqrt{3}/2\,r_s$ 後佈建感測器如圖(四)(b)。在 *V* mode 中的左轉或右轉是為了讓感測器佈建在最佳的位置如圖(四)(a)，並不是機器人真實的方向改變，所以佈建完感測器後機器人必須轉回原本面對的方向。以下我們提出機器人佈點法則。

Deployment Rule：機器人初始的佈點模式會先從 *H* mode 開始，只要一遇到機器人行進的方向改變 (*direction* = *left* or *direction* = *right*)，機器人佈點的模式會從當前的模式切換成另一個模式。 **q**

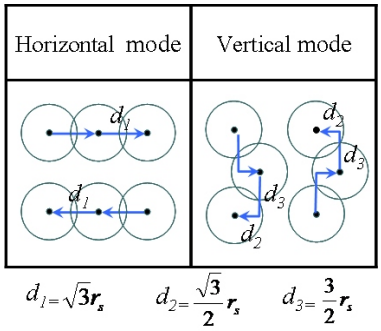
如圖(四)(c)所示，機器人由地圖的左上角開始佈點，一開始會依 *Deployment Rule* 以 *H* mode 開始佈點，當機器人佈建到*A*點時，機器人遇到地圖邊界即虛擬障礙物無法繼續前進，必須向右轉 *direction* = *right* 才可以繼續佈建感測器 node，當機器人轉彎，其佈點模式就會依 *Deployment Rule* 從 *H* mode 切換至 *V* mode 繼續佈建感測器。當機器人以 *H* mode 佈建感測器到*B*點時，行進方向

再度改變 *direction* = *right*，機器人佈建感測器的模式也會依照 *Deployment Rule*，由 *V* mode 切換到 *H* mode 繼續佈建感測器。

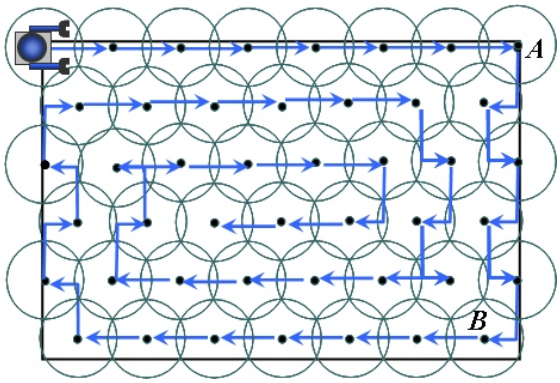
在機器人佈點的同時，會依序產生虛擬座標給每個被佈建的感測器，此座標系統為二維的座標系統，我們將機器人一開始面對的方向設定為*y*軸，並設定起始點所佈的感測器座標為原點，機器人每佈下一個感測器，就會依其移動的方向計算出虛擬座標的位置給予感測器。



(a)感測器之最佳佈建方式。



(b) 機器人佈點模式。



(c)機器人於無障礙物感測區域佈署感測器。

圖(四) Basic Deployment Rule

五、Obstacle-Free Robot Deployment Protocol

在本章節中將介紹機器人克服障礙物之佈點演算法。

(一) 基本概念

上一個章節所提出的Movement Rule及eployment Rule可在無障礙物的監測區域中，以最少的感測器達到全區覆蓋。而在監測區域有障礙物的環境，如圖(五)(a)所示，機器人會因真實障礙物以及自己走過的路徑(虛擬障礙物)產生死路問題，在這個章節中，我們將提出Obstacle-Free Robot Deployment Protocol，使機器人在有障礙物的監測區域依然可以解決死路問題，並順利的佈建感測器，以較少的感測器數量使監測區域達到全區覆蓋。

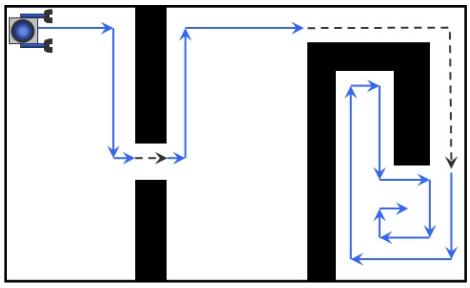
為了處理障礙物，機器人佈點時將會依周邊障礙物的艱困狀況而處於不同的狀態，並因狀態的不同而執行不同的處理程序，機器人所處的 *status* 分為 *Normal* state、*Narrow Lane* state、*DeadEnd* state、*GoBack* state四種。令 *status* 表示機器人佈點時周邊的障礙物狀態，當機器人在佈點的途中尚有兩個方向沒有障礙物或佈過的點時，我們稱這個狀態叫做 *Normal* state；當機器人在佈點的途中因為障礙物或是自己走過的路徑(虛擬障礙物)導致機器人進入一個狹路中，只有一個方向可以行進時，我們稱這個狀態叫做 *Narrow Lane* state；當機器人進入死路中，他的周圍都佈署過感測器或有障礙物時，我們稱這個狀態叫做 *DeadEnd* state ;當機器人進入*DeadEnd* state但監測區域尚未完全佈點，必須讓機器人可以離開 *DeadEnd* state 到達尚未佈點的區域繼續佈點，這個離開 *DeadEnd* state的過程機器人是純粹行進而不佈點，我們稱其狀態為 *GoBack* state。機器人每走一單位，便會檢查其周遭的障礙物情況，並設定自己所處的狀態。

當機器人進入 *Narrow Lane* state 時，由於機器人走過的路徑會被視為障礙物，機器人將有可能已經進入了一個出口被堵住的情境，如圖(五)(a)，為了避免機器人繞進死路後無法脫困，我們必須把每個 *Narrow Lane* 的入口及出口以及轉彎處的虛擬位置座標記錄起來，幫助機器人在遇到 *DeadEnd* state 的時候，可以順利脫困，

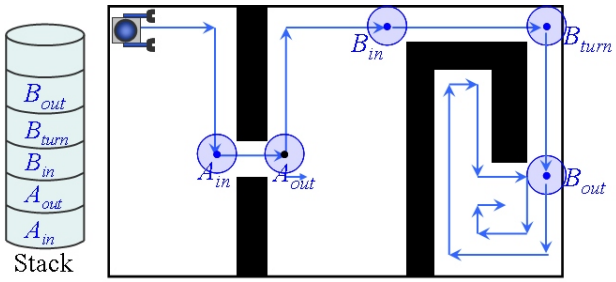
繼續完成佈點工作。

為使機器人克服因障礙物產生的死路問題，並有效率地回到未佈建區域，機器人會在記憶體中維護一個 stack，用來記錄機器人所要標記起來的虛擬座標位置。由於 stack 資料結構的特性為後進先出，所以愈晚被 push 進 stack 的資料將會愈早被 pop 出來，我們提出的演算法利用 stack 後進先出的特性，可以解決機器人進入多重死路的問題。

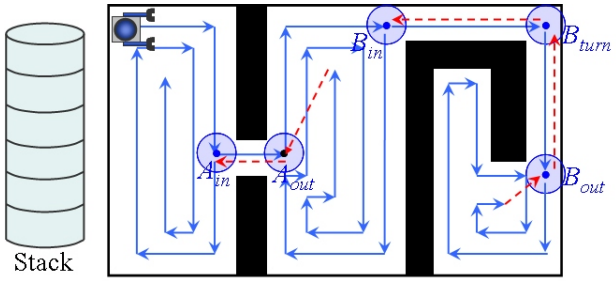
如圖(五)(b)所示，機器人會先進入*Narrow lane A*，將進入 *A* 之位置座標 A_{in}以及離開 *A* 之座標 A_{out} 依序 push 至 stack 中，機器人繼續佈點隨即進入 *Narrow lane B* ，一樣將進入 *Narrow lane B* 的位置座標 B_{in} push 到stack 之中，隨後機器人在 *Narrow lane B* 中改變佈點方向，機器人會將改變方向所在的感測器資訊 B_{turn} push 進stack 中，離開 *Narrow lane B* 的同時也會在 stack 中 push B_{out} 之資訊，接著機器人會繞進因自己走過的路徑(即虛擬障礙物)而遇到 *DeadEnd* state，這時就會進入 *GoBack* state，自 stack 依後進先出的原則 pop 出感測器資訊，並前往該座標繼續進行佈點的工作。機器人會不斷重複上述動作，直到遇到 *DeadEnd state* 且 stack 已為空，這代表感測區域已達到全區覆蓋，佈點工作完成如圖(五)(c)。



(a)機器人會因真實障礙物以及自己走過的路徑(虛擬障礙物)產生死巷問題。虛線表示為機器人行經Narrow lane之移動軌跡。



(b)將*Narrow Lane* 的入口及出口以及轉彎處所佈署的感測器位置資訊Push至Stack。



(c)當遇到*DeadEnd* state時依序pop出感測器資訊，並進入*GoBack* state，圖中虛線表示為機器人於*GoBack* state之移動軌跡。

圖(五) Obstacle-Free Robot Deployment Protocol

(二) Obstacle Handling Rules

機器人每走一單位，便會檢查其周遭的障礙物情況，並設定自己所處的 *status*。機器人會依周邊的障礙物狀況進行四種狀態之間的轉換並執行不同的處理程序，如圖(六)(a)。以下提出一 Rule 來描述機器人處於 *Normal* State 的處理程序。

Normal Rule：當機器人周邊障礙物狀態*status* = *Normal* state 時，分別依 *Movement Rule* 及 *Deployment Rule* 移動及佈點。 **q**

機器人克障演算法會將進入 *Narrow Lane* state 之前的感測器資訊以及離開*Narrow Lane* statet 回到 *Normal* state 的感測器資訊記錄在 stack *K* 中，為了讓機器人可以克服彎曲型的 *Narrow lane* ，機器人也會將在 *Narrow lane* 中轉

彎的感測器資訊 push 進 stack K 中，利用這些感測器資訊幫助機器人克服障礙物，如圖(六)(b)所示。stack K 為四維矩陣，$K[i]$ 代表 stack 中序號為 i 的感測器資訊，其中包含 (x_i, y_i, io_i, nl_i)，其中(x_i, y_i) 記錄著感測器 i 的位置； io_i 紀錄感測器 i 為進入 *Narrow lane* 所佈署的感測器或是離開 *Narrow lane* 所佈署的感測器，若 $io_i = in$ ，表示感測器 i 為進入 *Narrow lane* 所佈署的感測器，若 $io_i = out$，則表示感測器 i 為離開 *Narrow lane* 所佈署的感測器，若 $io_i = turn$ 則表示感測器 i 位於*Narrow lane*中的轉彎處; nl_i 代表此感測器 i 所記錄的 *Narrow lane* 編號，每 push 一個進入 *Narrow lane* 的感測器資訊到 stack 中，紀錄的 *nl* 值就會累加一次。以下提出機器人遇到 *Narrow Lane* state 時的處理程序。

Narror Lane Rule：當機器人周邊障礙物狀態 *status*由 *Normal* state 轉換成 *Narrow Lane* state 時，機器人將進入 *Narrow lane* 前所佈署的感測器資訊 push 至 stack K 中；當機器人周邊障礙物狀態 *status* 由 *Narrow Lane* state 回到 *Normal* state 時，亦會將離開 *Narrow Lane* state 所佈署的感測器資訊 push 至 stack K 中；在 *Narrow lane* 中，若機器人佈點的行進方向改變 (*direction* = left or *direction* = right)，也會將此感測器資訊push 至 stack 中。 **q**

當機器人離開一個 *Narrow lane* 隨即又進入另一個 *Narrow lane* 時，有可能會使 stack 存有兩個同樣的感測器資訊。如圖(六)(c)機器人離開 *Narrow lane A* 後隨即進入了 *Narrow lane B* ，感測器 A_{out} 及感測器 B_{in} 其實記錄了同樣的感測器資訊。為了盡可能的節省機器人的記憶體空間，機器人會先將上一個離開 *Narrow lane* 的感測器資訊自stack 中 pop 出來，然後才將進入新的 *Narrow lane* 的感測器資訊 push 進 stack中。以圖(六)(c)為例，當機器人離開 *Narrow lane A* 後隨即要進入 *Narrow lane B* 時，會先將感測器 A_{out} 自 stack pop，隨後才會將感測器 B_{in} push 進 stack，如此就不會有感測器資訊重複紀錄的問題。以下提出一Rule 來描述當機器人紀錄的感測器資訊重疊所做的處裡程序。

Narrow Lane Exist_Enter Rule：當機器人離開一個 *Narrow lane* 馬上又要進入另一個 *Narrow lane* 時，機器人會將前一個離開 *Narrow lane* 紀錄的感測器資訊自 stack K 中 pop 後才將目前進入的 *Narrow lane* 之感測器資訊 push 至 stack K 中。 **q**

當機器人周邊障礙物狀態遇到*DeadEnd* state 時，機器人將利用 stack 中的感測器資訊判斷是否繼續佈點。若 stack 中尚有感測器資訊，則進入 *GoBack* state 以脫離其遭遇之 *DeadEnd* 位置，回到尚未佈點的區域，繼續完成佈點工作;當機器人遇到 *DeadEnd* state 且 stack 同時為空時，代表感測區域已無空洞，機器人佈點工作結束如圖(五)(c)。以下提出機器人遇到*DeadEnd* state 時的處理程序。

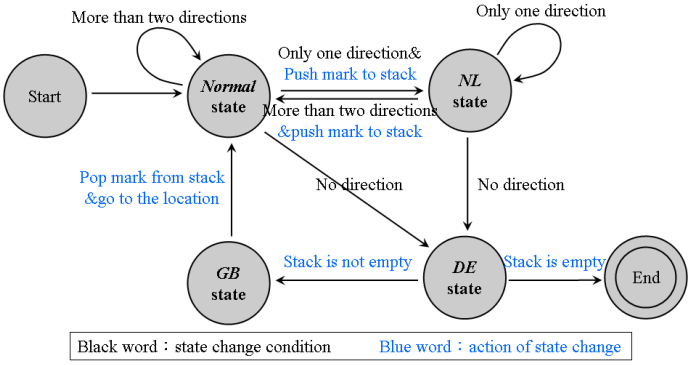
Dead End Rule：當機器人周邊障礙物狀態為 *DeadEnd* state 時，若 stack K 不為空，則進入 *GoBack* state；若遇到 *DeadEnd* state 且 stack K 為空時，演算法結束。 **q**

當機器人進入 *GoBack* state 時，機器人會利用 stack 中的感測器資訊離開已佈署過感測器的區域，前往未完成佈點的監測區域繼續佈點。如圖(五)(c)所示，機器遇到死路且 stack K 不為空，則會進入 *GoBack* state，此時機器人會從 stack K 中 pop 出感測器資訊 $K[i]= (x_i, y_i)$，然後以直線方式前往該座標 (x_i, y_i)，在 *GoBack* state 中，機器人移動時，並不會執行佈點的動作。若機器人到達該座標後，發現周圍障礙物狀態 *status* 依然為 *DeadEnd* state，就會繼續回到 *GoBack* state，繼續 pop 感測器資訊，直到機器人到達未佈點區域，之後便繼續佈點。以下提出機器人進入 *GoBack* state 時的處理程序。

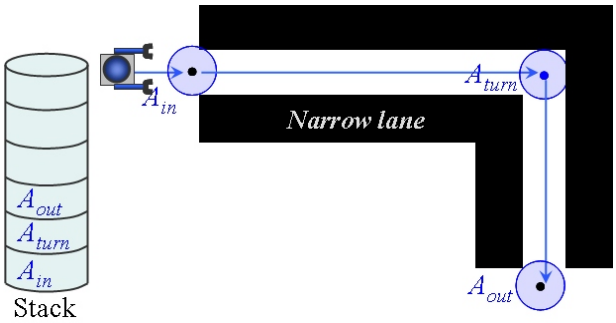
GoBack Rule：當機器人進入 *GoBack* state，機器

人會 pop 出一個感測器資訊$K[i] = (x_i, y_i)$，機器人會以直線的方式前往該座標 (x_i, y_i)，在 *GoBack* state 中機器人僅前進但不執行佈點動作。 ■

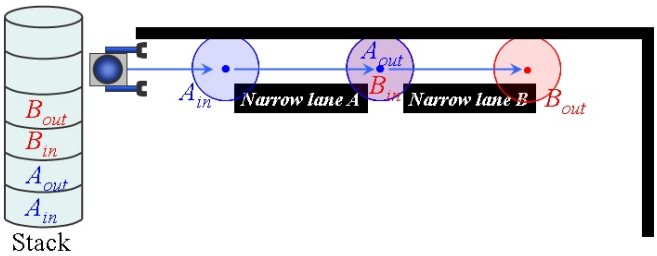
依照上述 Obstacle Handling Rules 機器人將可以避免死巷問題以及空洞問題，使感測區域達到全區覆蓋。



(a)克障演算法之狀態轉換圖



(b)克障演算法會將機器人進入、離開*Narrow lane*以及在的*Narrow lane*中發生方向改變時所佈署的感測器資訊記錄在stack中。



(c)機器人可能遇到感測器資訊重複紀錄的情形。

圖(六)：Obstacle Handling Rule

(三) Stack Free Procedure

機器人執行上述的方法，會將 *Narrow lane* 的出入口以及轉折處的所佈署的感測器資訊紀錄至 stack 中，並利用這些感測器資訊讓機器人在有障礙物的感測環境中能順利的佈建感測器，以達到全區覆蓋的目的。但當感測區域已經達成全區覆蓋時，stack 中可能會還有尚未被 pop 出的感測器資訊如圖(七)(a)，這將造成機器人依照 Obstacle Handling Rules 繼續 pop 記錄在 stack 中的感測器資訊，並前往感測器位置，直到 stack 為空才會結束工作。間接造成機器人在感測區域達成全區覆蓋後，還有行走多餘路徑浪費電力的動作發生。

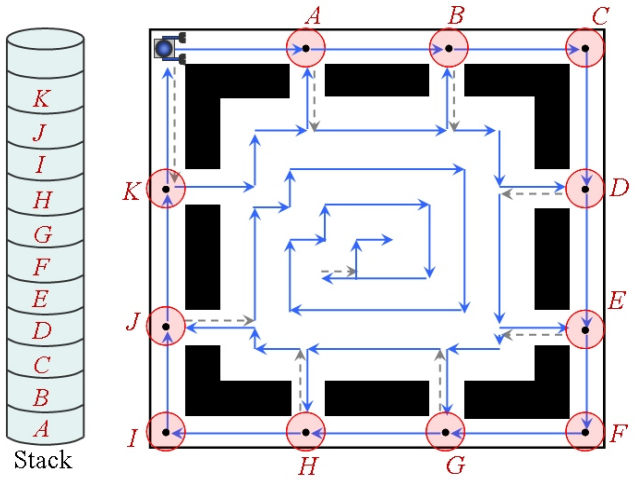
為了解決這個問題我們必須把在 stack 中周圍已無空洞的感測器之資訊自 stack 中刪除。本節提出一個消除 stack中感測器資訊的演算法，當被記錄的感測器周圍已無空洞時，機器人就會將感測器資訊自 stack 中刪除。如此一來就可避免stack中存在無用的感測器資訊而導致機器人行走多餘路徑的狀況發生。

我們進一步分析機器人進入 *Narrow lane* 後可能遇到的情形有兩種，若機器人自 *Narrow lane* 進入一個密閉式的障礙物中，機器人繼續佈點必定會遇到 *DeadEnd* state ，機器人將利用在 stack 中的感測器資訊離開 *DeadEnd* 繼續進行佈點工作如圖(五)(d)。若機器人進入 *Narrow lane* 後沒有遇到 *DeadEnd* state ，機器人繼續佈點將會再度回到已被紀錄感測器資訊 的*Narrow lane* 入口附近如圖(七)(b)。此時 *Narrow lane* 的入口周圍的空洞會被感測器感測器覆蓋，機器人將沒有必要再回到此處佈點，因此可將記錄此 *Narrow lane* 的感測器資訊自 stack 中刪除。以下 Rule 說明自 stack 中刪除已無利用必要的感測器資訊之處裡程序。

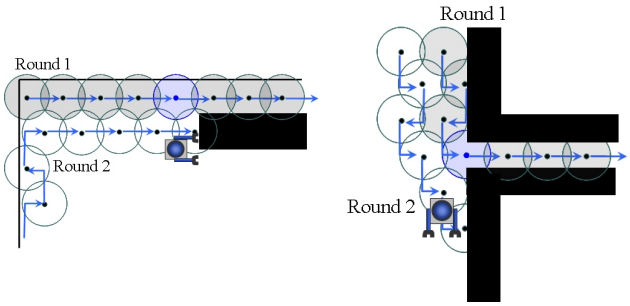
Stack Free Rule：當機器人行進至stack *K*中的任一感測器 s_i 周圍時，stack *K*中紀錄感測器s_i的陣列為$K[n]$，若$K[n]$中所記錄的 $io_n = in$，則機器人將所有與感測器 s_i 具相同 *nl*值(紀錄相同之*Narrow*

lane)之感測器資訊自stack *K*中刪除。 **q**

利用這條 Rule ，機器人可以將已沒有利用必要的感測器資訊自 stack 中刪除，在感測區域達到全區覆蓋時，機器人就不會浪費能源行走多餘的路徑。



(a)感測區域已達成全區覆蓋若 stack 中仍有感測器資訊，機器人將會繼續執行演算法。



(b)機器人行進入 *Narrow lane* 入後若無遇到 *DeadEnd* state ，機器人繼續佈點將經過此 *Narrow lane* 入口附近。

圖(七)：Stack Free Procedure

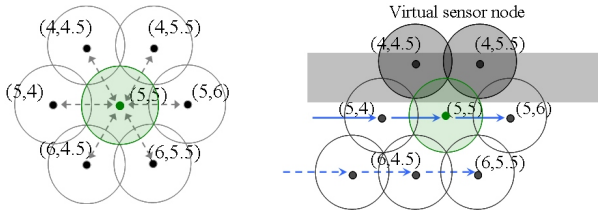
(四) Power saving Procedure

在本論文中，機器人佈點的過程必須與先前已被佈建的感測器進行溝通，將已被佈建過感測器的區域視為虛擬障礙物進行佈點。但當一個感測器周圍都已被機器人佈署滿其他感測器而沒有空洞時，此感測器就不再有與機器人進行溝通的必要，為了減少已佈建的感測器與機器人溝通所要耗費的能源成本，我們提出了一個省電機制，讓已被佈建的感測器在沒有與機器人溝通之必要時，可以進入睡眠狀態達到省電的效果。

在感測區域中每個感測器都會與其周圍的一步鄰居進行溝通，交換彼此的虛擬座標位置訊息，感測器也會知道其一步鄰居的數量，當感測器收集了周圍六個一步鄰居的資訊，會依此判斷其周圍已不存在空洞，因此可以進入睡眠狀態如圖(八)(a)。

為了讓感測器在其周圍有障礙物時依然可以順利的進入睡眠狀態，機器人在佈建感測器的同時，會將超音波感測到的障礙資訊，轉化為虛擬感測器(Vitual sensor node)座標並給予被佈建的感測器圖(八)(b)，已被佈建的感測器就可以利用這些虛擬的感測器資訊判斷是否要進入睡眠狀態。

利用這個感測器的省電機制，將可以減低感測器和機器人溝通的成本，節省感測器的電量，使感測器有更充足的能源去進行感測與通訊工作。



(a)感測器會與其鄰居溝通並判斷是否進入睡眠狀態。

(b)當遇到障礙物，機器人會給予感測器虛擬的鄰居資訊。

圖(八)：Power Saving Procedure

六、效能評估

為了評估本論文所設計的克障佈點演算法，在本章節中將與 ORRD[6]、OFPD[7]以及 BSA[9]所設計的機器人佈點方法做模擬分析與比較。本模擬將針對 (1)機器人佈點的覆蓋率 (2)機器人佈點所

花費的時間 (3)機器人佈點所需要行走的路徑長度來評估佈點演算法的效能。首先，我們先說明本論文的模擬環境，再探討本論文與其他論文之各參數的模擬結果。

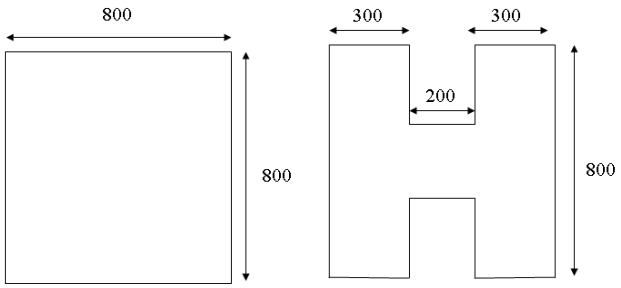
(一) Simulation Model

在模擬中我們參考 Berkeley motes [10] 之特性來設定相關參數，其詳細內容如 Table I：

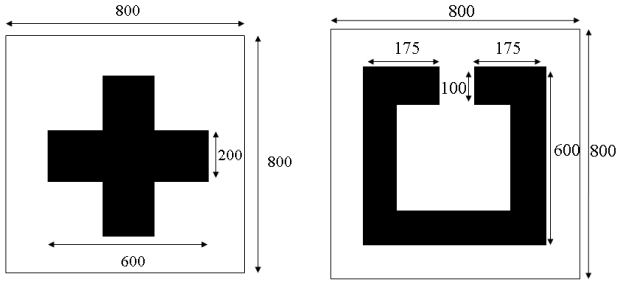
Table I : Simulation parameters

Parameter	**Value**
Simulator	C
Communication range	75m
Sensing range	40m
Map size	800m*800m
Robot speed	2 m/s
Packet transmission cost	0.075J/s
Packet reception cost	0.030J/s
Mobility cost	8.267J/m

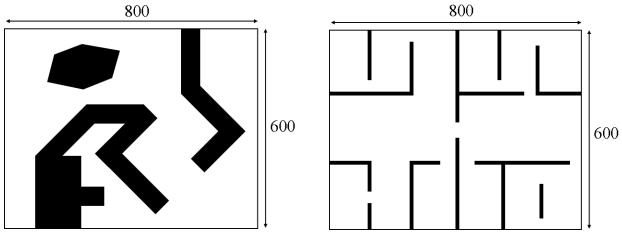
機器人配帶有固定數量的感測器，我們假設機器人其速率為 2m/s，感測器 s 的通訊半徑為 75 公尺，感測半徑為 40 公尺。在環境方面，模擬環境如圖(九)中六個實驗場景，分別為(a)正方形監測區域 (b) H 型監測區域 (c)X 型障礙物 (d)C 型障礙物 (e)複雜障礙物 (f)展覽場地型。機器人的出發點假設在環境之左上角，此外，各模擬結果為 20 次獨立模擬結果的平均值。



圖(九) (a)　圖(九) (b)



圖(九) (c)　圖(九) (d)

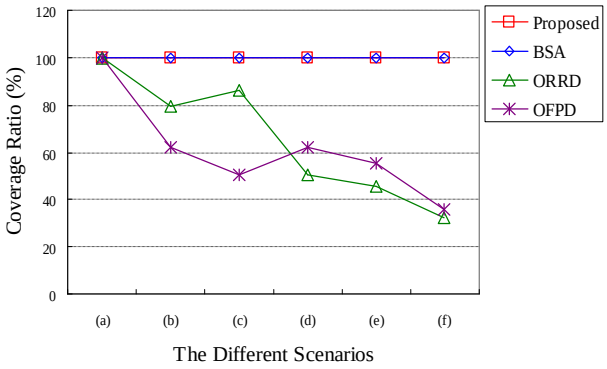


圖(九) (e)　圖(九) (f)

圖(九)實驗場景圖

(二) Performance Study

首先我們針對機器人在不同監測場景中佈點的覆蓋率進行模擬，圖(十)的實驗結果所示本論文所提出的演算法及 BSA 皆可在六個不同的監測場景中達到全區覆蓋，而 ORRD 以及 OFPD 會因監測區域及障礙物的複雜度影響到覆蓋率的高低。

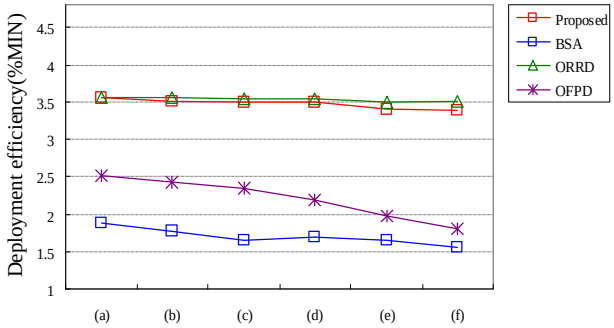


圖(十):在不同的間測區域中佈點覆蓋率比較

在不同之監測場景中，每個演算法具有不同的佈點效率，下列為計算佈點效率評估的公式：

$$Deployment\ Efficiency = \frac{Number\ of\ Deployed\ Sensors}{Deployment\ Time} \times Coverage\ Ratio \quad (5)$$

實驗中依照以上公式對每個演算法進行佈點效率的計算。由圖(11)的模擬結果可知，本論文所提出的演算法與ORRD皆具有較高的佈點效率，而BSA及OFPD佈點的效率則較低，主要原因為BSA及OFPD演算法在佈點的過程中有許多冗長的路徑。



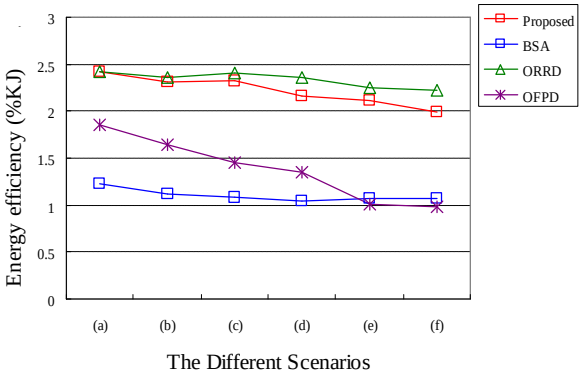
圖(十一):在不同之間測區域中佈點效率比較

接下來本論文針對了佈點演算法中機器人的能源使用效率作了比較。能源使用效率意指機器人總消耗電量與其所達成的覆蓋率的比值，如下列公式：

$$Energy\ Efficiency = \frac{Coverage\ Ratio}{Total\ Energy\ Consumption} \quad (6)$$

實驗結果如圖(十二)所示，本論文以及ORRD具有較高的能源使用效率，而ORRD的能源使用效率較本論文稍高的原因是，ORRD並不保證感測區域達到全區覆蓋，在複雜的監測區域中ORRD演算法若遇到死路即會停止運作，而本論文所提出的演算法則會克服更種複雜的障礙物，因此在克障的部分付出了一些額外的電力成本。

由上述實驗結果可知本論文所提出的演算法不但可以保證全區覆蓋還具有較高的佈點效率且只需較低的電力成本。雖然BSA也可達到全區覆蓋，但其佈點效率不高，且需要很高的電力成本，而ORRD及OFPD則無法達到全區覆蓋。



圖(十二):不同之間測區域中能源使用效率比較

七、結論

本論文提出一有效率的機器人佈點的演算法，不同於現有的機器人佈點演算法，本論文所提出的演算法以少量的記憶成本讓機器人可以克服任何複雜的障礙物，並將適量的感應器佈建於監控區中以達到縮短佈建時間、節省硬題成本及全區覆蓋等目的。此外，機器人僅需與已佈建的感測器進行少量通訊，使大多數已被佈建的感測器可進入省電狀態。實驗顯示，本論文所提出的演算法具有高佈點率、低電力成本，且可以在複雜的監測區域中達到全區覆蓋。

八、參考文獻

[1] Maxim A. Batalin and Gaurav S. Sukhatme, “Efficient Exploration without Localization,” *The 2003 IEEE International Conference on Robotics and Automation* (*ICRA*), Taipei, Taiwan, May 2003, pp. 2714–2719.

[2] M. J. Mataric, “Behavior-based control: Examples from Navigation, learning, and group behavior,” *Journal of Experimental and Theoretical Artificial Intelligence, special issue on Software Architectures for Physical Agents*, vol. 9, no. 2–3, 1997, pp. 323–336.

[3] P. Pirjanian, "Behavior coordination mechanisms-state-of-the-eart," Technic Report, *Institute for Robotics and Intelligent Systems*, University of Southern California, October 1999, IRIS-99-375.

[4] Maxim A. Batalin and Gaurav S. Sukhatme, "Coverage, Exploration and Deployment by a Mobile Robot and Communication Network," *The 2003 International Workshop on Information Processing in* sensor *Networks* (*IPSN*), Palo Alto, Apr. 2003, pp. 376–391.

[5] Maxim A. Batalin, Gaurav S. Sukhatme and Myron Hattig, "Mobile Robot Navigation using a sensor Network," *The 2004 IEEE International Conference on Robotics & Automation* (*ICRA*), New Orleans, LA, April 2004, pp. 636–642.

[6] C. Y. Chang, C. T. Chang, Y.C. Chen and H. R. Chang, "Obstacle-Resistant Deployment Algorithms for Wireless sensor Networks," *IEEE Transactions on Vehicular Technology* (*IEEE TVT*), vol. 58, no. 6, July 2009, pp. 2925–2941.

[7] C. Y. Chang, J. P. Sheu, and Y. C. Chen, "Obstacle-Free and Power Efficient Deployment Algorithm for Wireless sensor Networks," *Proceedings of IEEE Transactions on Systems, Man, and Cybernetics--Part A: Systems and Humans*, 2009.

[8] Yongguo Mei, Yung-Hsiang Lu, Lee, C.S.G., Hu, Y.C. "Energy-Efficient Mobile Robot Exploration," *The 2006 IEEE International Conference on Robotics and Automation* (*ICRA 2006*), May 2006.

[9] Gonzalez, E.; Alarcon, M.; Aristizabal, P.; Parra, C. "BSA: a coverage algorithm," *The 2003 Intelligent Robots and Systems, 2003.* (*IROS 2003*).

[10] J. Hill and D. Culler, "A Wireless Embedded sensor Architecture for System-level Optimization," Technical Report, *Computer Science Department*, University of California at Berkeley, 2002.

[11] Saurabh Ganeriwal, Aman Kansal and Mani B. Srivastava, "Self Aware Actuation for Fault Repair in sensor Neworks," *The 2004 IEEE International Conference on Robotics and Automation* (*ICRA*), April 2004.