無線感測網路中可克服障礙物之網路佈建及毀損修復之機器人演算法

張志勇
私立淡江大學資訊工程研究所副教授
台北縣淡水鎮英專路 151 號
cychang@mail.tku.edu.tw

張栩瑞
私立淡江大學資訊工程研究所博士生
台北縣淡水鎮英專路 151 號
hrchang@wireless.cs.tku.edu.tw

謝珍琪
私立淡江大學資訊工程研究所碩士
台北縣淡水鎮英專路 151 號
jenchi@wireless.cs.tku.edu.tw

摘要

現今無線感測網路(Wireless Sensor Networks 或 WSNs)已廣泛地應用在闖入者監測、軍事作戰監控系統，如何將大量的感測點佈置於特定欲監測區域內，並維持高效率的監測效能是個重要的議題。本論文研發一機器人在無線感測網路上佈點、修復及回家(Home)的演算法，我們所研發的佈點演算法使機器人能快速且花費最少的感測點數量佈點於欲監測區域，並有效地避開障礙物，使得闖入者不論處於監測區域的任何位置，都能被我們所佈置的感測點監控到。此外，機器人在行走過程中留下其行走的軌跡，使無線感測網路中的感測點皆能追蹤機器人的位置資訊，並能以較短路徑發送修復封包以通知機器人進行毀損修復，使機器人能以動態、快速、省電及有效率的方式往修復區移動，以避免闖入者處於毀損的感測點附近，使得監控者無法掌握闖入者的行蹤。最後，我們提出回家演算法來讓機器人在進行修復的過程中，能以剩餘電量來安排較佳行走路徑，以達到回家充電及補充新的感測點的目的，使無線感測網路的覆蓋範圍得以更有效地維護。實驗顯示本論文所提出的演算法可以使網路中的感測點有效掌握機器人的移動軌跡，並利用此資訊提供機器人一個動態且快速的巡邏及修復演算法。

關鍵字：佈點、修復、無線感測網路、機器人、障礙物

一、導論

無線感測網路是由大量的感測點所建構而成，每個感測點均具有嵌入式的處理器、記憶體、微感應裝置及無線通訊設備，這些微小、成本低廉且由電池供電的感測點被大量地放置於特定區域，作爲環境偵測或軍事作戰監控等應用。而無線感測網路的偵測效能決定於所有感測點其感測範圍的涵蓋面積，將大量的感測點以隨機佈建(Random Deployment)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

較易造成感測點佈點不均，導致某區域的感測點過於稀疏或密集，分別造成感測空洞或浪費過多的感測點等問題。

利用機器人來佈置感測點，除了可以採用規則的方式來佈置感測點外，更能達到使用最少的感測點數量且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爲佈點的地方諸如有毒氣、化學污染等區域佈點[1][2][3]。在感測點佈建的研究中，[1][4][5][8][9][12]利用機器人來進行無線感測網路中許多重要的工作，包括佈建感測點、巡邏或修復毀損之感測點等任務。在[1][4][5]的研究中，機器人隨南西北東四個方向的優先權逐步移動並佈置感測點，在不需要位置資訊的條件下，使用已佈下的感測點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點時，機器人即佈置新的感測點。而在機器人之通訊範圍內的感測點，能夠區域性地建議機器人最適當的探勘方向，機器人將這些感測點的建議結合起來並選擇出一個最適當的方向進行探勘，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，將無法保證整個欲監測區域爲全區感測覆蓋(Full Coverage)，可能造成感應空動或感測區域之重疊。此外，此論文僅將機器人所擔負的任務局限於佈點及監控，沒有考慮到機器人所配帶的感測點會耗盡的問題。

利用機器人在無線感測網路中佈滿了感測點後，由於無線感測網路在感測點的長期監測下，有些感測點可能因傳送或代傳感測資料的機會較多，導致其有限電量容易耗盡，許多感測點亦可能因發生故障而無法正常運作，這些毀損的感測點甚至可能造成整個無線感測網路斷裂，縮短了無線感測網路的生命週期，因此，爲了維持有效的環境偵測，針對無法運作的感測點進行再次佈建是必要的。在這方面的研究中，[1][4][5][8][9] 利用機器人進行網路的佈建，但對於網路中的空洞不能有效地改善，需要等到機器人走至此空洞，並發現沒有感測點可以通訊後，才放置感測點填補此空洞，無法在空洞形成時即立刻前往修補，因此，對於網路中有空洞的情形，欲完成全區感測覆蓋的要求是相當地耗費時間。研究[2]以機器人來佈置感測點且監控網路狀況，而已佈置的感測點將用來導航機器人，機器人在此感測區域內進行監控網路與修復故障或損毀感測點的工作，此外，其更探討機器人回復的問題，提供機器人回家的機制，讓機器人能夠回家補充電量，然而，其對多個毀損區的行走路線並未有效規畫，機器人無法針對多個毀損區建構一最短行走路徑以達到節省電量的目的，另一方面，由於毀損區旁的感測點無法有效追蹤機器人的行走軌跡，加之機器人行走路徑非最短路徑，因此造成毀損區無法即時修復與機器人耗電過多等問題。

本論文利用移動的機器人(Mobile Sink)來佈建感測點，使機器人能在最短時間內以最少的感測點將整個區域佈建使其具有全區感測覆蓋的能力，當網路中有天然障礙物存在時，機器人亦能以最有效率的方式避開障礙物並完成具全區感測覆蓋的網路佈建。此外，機器人會在無線感測網路中留下其行走的軌跡，讓網路中每個感測點皆可追蹤(Tracking)機器人目前的位置，當感測點損毀或是電量即將耗盡時，將能利用機器人所留下的軌跡以較短的路徑及較少的控制封包(Control Packet)通知機器人進行修復，當機器人接收到許多毀損區域的資訊時，將能建立最短的行走路徑使其以最省電、最即時的方式以其所攜帶的感測點來修復這些毀損區，及早替換電量快耗盡或已耗盡的感測點，維持網路的偵測效能。此外，機器人在決定其行走路徑時亦將回家充電、回收舊的感測點、以及補充新的感測點等需求納入計算。我們所研發的演算法使機器人以具有避障、動態、

快速、省電及有效率等優點在無線感測網路中進行網路佈建、往修復區移動與巡邏、以及回家充電及補充新的感測點，使無線感測網路的覆蓋範圍得以更有效地維護。

以下，我們將在第二章中介紹本論文的環境與假設，以及流覽本論文的工作範圍及內容;在第三章中，我們將介紹本論文所研發有能力克服障礙物且可達到全區感測覆蓋的佈點技術；在第四章中，我們將介紹機器人以省電方式修復網路的技術，在第五章中，我們將以實驗來檢視本論文所提出的方法其效率，最後則是本論文的結論。

二、網路環境與基本概念

在本章節中，我們介紹本論文的環境假設及基本概念。

1. 網路環境與假設

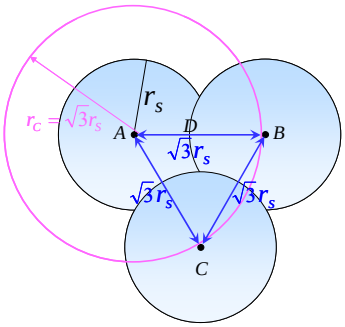
本論文的環境假設如下，我們所使用的單一機器人將配帶有個數有限制的感測點，機器人可以決定行走的方向爲東南西北，一開始環境假設機器人被放置在欲監測區域的邊界，機器人可以不需知道此欲監測區域周圍的邊界資訊。在此，我們假設通訊半徑(Communication Range)至少是感測半徑(Sensing Range)的 $\sqrt{3}$ 倍。

2. 基本概念

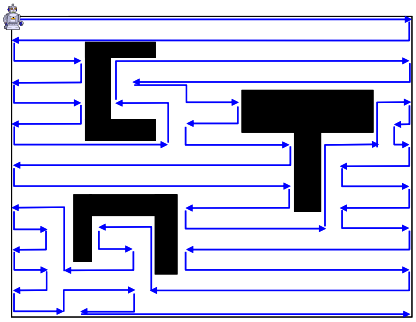
由於感測網路的效能與感測網路中感測點的感測覆蓋息息相關，爲了使佈建的感測點能達到全區感測覆蓋的目的並節省感測點硬體上的成本，我們希望盡量減少感測點其感測範圍的重疊區域[11][12]，圖一(a)所示爲感測點能擁有最大感測覆蓋其彼此之間最好的距離，假設半徑 r_s 爲感測點的感測半徑， r_c 爲感測點的通訊半徑，在滿足 $r_c \geq \sqrt{3} r_s$ 的條件下，圖一(a)中感測點 A 與感測點 B 距離爲 $\sqrt{3} r_s$ 時，其彼此之間可以互相通訊，若感測網路中的感測點排列如圖一(a)所示時，將能使感測點的感測覆蓋重覆範圍最小但卻可滿足全區感測覆蓋的要求，具有感測覆蓋最大且所需佈置的感測點點數最少的優點，並使每個感測點與其周圍的感測點皆具有通訊上的連結性(Connectivity)。我們將利用此距離作爲本論文中佈建的依據並設計一個機器人佈建的演算法。而機器人佈建感測點的方法有許多種，爲了有效運用圖一(a)中感測點彼此之間的最佳距離來執行網路佈建，在本論文中，我們將採用由蛇行狀的佈建方式，機器人從感測區域的邊界出發並依序放置感測點，在機器人佈點的同時，機器人也依序分配座標給每個感測點，並以蛇行狀的方式佈點且分配座標給每個感測點。此外，在有障礙物的監測環境中，本論文所設計的蛇行狀佈點也能隨著障礙物的不同與複雜度，克服障礙物並有效地佈點，如圖一(b)所示，在有多個障礙物的區域環境中，機器人也能以本論文所設計的蛇行狀佈點法，繞障礙物快速佈點，使網路中的每個區域皆能被感測點所覆蓋而不致於因障礙物使佈點產生空洞。

在機器人佈點的同時，將會留下適當的軌跡，當感測網路經過長時間的偵測之後，某些感測點發生故障或是即將耗盡電量需要修復時，偵測到故障區域的感測點或是即將耗盡電量的感測點，將會利用機器人所留下的軌跡以較短的路徑追蹤機器人，並通知機器人至此區域進行修復，此過程稱之爲修復的演算法。若機器人收到許多感測點等待修復的資訊時，爲達到迅速修復及機器人省電的目的，機器人將以 Dynamic Programming 的技巧來建立一通過各毀損區的最短路徑。最後，我們將擴充機器人修復的演算法，使其兼具考量機器人因電量耗盡或爲補充感測點而回家的機制。機器人有許多原因必須回

家，首先，機器人在執行修復與巡邏感測點的過程中，機器人本身也可能會耗盡電量，其次，機器人身上所配帶的良好的感測點可能用完，再者，毀損感測點亦需透過機器人回收以免影響環境，因此，我們在機器人執行修復感測點的過程中，將機器人回家以充電、補充良好的感測點及回收毀損感測點之需求加入考量，以防止機器人在執行修復與巡邏感測點的過程中，因電量耗盡而來不及回家補充電量。



(a) 感測點彼此相距 $\sqrt{3}$ 倍的感測半徑，將能以花費最少的感測點數量來達到最大的感測覆蓋。



(b) 機器人以蛇行狀佈點克服障礙物的環境。

圖一：機器人佈點的規則與蛇行狀佈點法。

綜合上述所提及的相關問題，我們將設計一機器人在無線感測網路中佈點、修復的演算法，讓機器人能以最少的點數，在最短時間內佈建完欲監測的區域，並有能力克服障礙物，當有感測點電量即將耗盡或是損毀時，機器人能以最省電的方式來修復待修復的區域，此外，更加考量機器人能回家及避開障礙物修復的條件，使無線感測網路的覆蓋範圍得以更有效地維護。以下，在後續章節中，我們將詳細介紹我們所研發的演算法。

三、克服障礙物之佈點方法

在本章節中，我們將正式敘述機器人克服障礙物的蛇形佈點技術。由於感測網路的偵測效能與感測點所涵蓋的範圍是相輔相成的，所以我們希望機器人在佈點時盡量能完全覆蓋欲監測的區域，以達到較高的偵測效能，此外，若所佈置的感測點點數過多，彼此之間感測範圍的重疊區域也較大，導致許多感測點負責偵測的區域重複，增加感測點的硬體成本及通訊時的耗電量，所以我們希望能在花費最少感測點點數的條件下，達到全區感測覆蓋的目的。

1. 機器人蛇形佈點方法

我們根據圖一(a)中$\sqrt{3}r_s$的距離，利用機器人設計一個蛇行狀的佈點演算法，當機器人佈置完一排感測點後，將往南方移動 $(3/2)r_s$，也就是圖一(a)中的 $\overline{CD}$，並在上排感測點彼此之間的中垂線上佈置感測點，使感測點彼此之間皆相距$\sqrt{3}r_s$，由於機器人配帶有羅盤針，具有辨識方向與角度的能力，所以能決定其行走的方向與距離。以下，我們首先詳述機器人在欲監測區域內其蛇行狀佈點的規則。

蛇形狀佈點的規則主要分爲兩個狀態，分別爲向東行走(East Movement，簡稱 East)狀態，與向西行走(West Movement，簡稱 West)狀態，在向東行走與向西行走狀態中，皆具有兩個行走方向的選擇權且依優先權先後決定機器人應行走的方向，如表一所示，其中欲行走之方向 1(Prefer Direction 1)的設計使機器人有能力東西向蛇行狀行走佈點，而欲行走方向 2(Prefer Direction 2)的設計使機器人能在遇邊界時能由北往南移動一步，再以東西向蛇行佈點：

表一：蛇行狀佈點

States	Prefer Direction 1	Prefer Direction 2
East	➔	➓
West	➔	➓

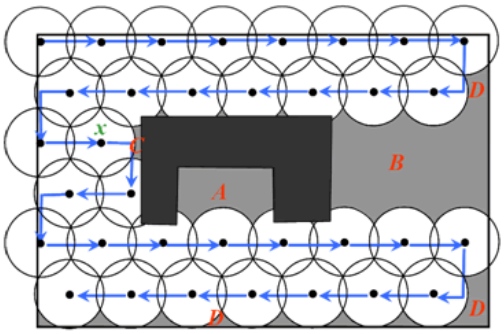
若單以表一的蛇行狀佈點來佈點，需有機器人必須在其欲監測區域最北方的限制，若機器人在其最北方的最東邊邊界出發，機器人將先進入向西行走狀態，反之，若機器人在其最北方的最西邊邊界出發，機器人將先進入向東行走狀態。而機器人判斷行走方向的時機可隨時判斷或是行走固定長度後才判斷，由於機器人佈點是以圖一(a)中每點相距$\sqrt{3}r_s$來佈點，爲簡化敘述，在本論文中我們採用機器人每佈下一感測點後再判斷下一行走方向來說明；當機器人處於某一狀態時，機器人每次行走，皆先嘗試往欲行走方向 1 的方向行走，亦即爲欲行走的方向，若遇到欲行走方向 1 的方向已有感測點存在或是遇到邊界或障礙物時，表示往欲行走方向 1 方向行走失敗，則機器人將會選擇欲行走方向 2 的方向行走，每執行欲行走方向 2 成功後，機器人將會切換至另一個狀態，如此，向東行走與向西行走兩個狀態交替執行，形成蛇行狀佈點。

2. 克服障礙物之機器人蛇形佈點方法

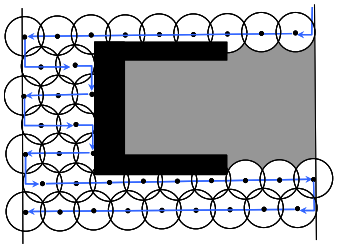
上述蛇行狀的佈點方式，在機器人遇有障礙物時，將可能造成部分區域沒有佈置感測點，且偵測區域的邊界以及障礙物的邊界，皆會有空洞的產生，如圖二中的灰色區域所示，其原因在於，機器人由東西向蛇行狀佈點方式，當遇到障礙物後，如圖二中的 *x* 點，機器人僅能以蛇行狀佈點於障礙物的左邊，而對於障礙物的右邊如圖二中的 *B* 區，將造成空洞，此外，對於障礙物的凹洞如圖二的 *A* 區，亦將因蛇行狀佈建之東西向佈點方式而造成空洞。另一方面，在障礙物的邊界如圖中的 *C* 區，與欲監測區域的邊界如圖中的 *D* 區，也會有空洞的產生。因此，我們將設計出一個具有克服障礙物且解決邊界空洞能力的蛇行狀佈點規則。

爲了要達到蛇行狀佈點時克服障礙物的目的，機器人在執行從左至右或從右至左佈點時，必須隨時檢視其北方是否有空洞如圖二中的 *A*、*B* 區所示，若有空洞，機器人將須先往北方行走，且以防此障礙物有內凹的情形如圖三(a)所示，機器人需具有能沿著此障礙物的凹洞往北方行走的能力，直到此空洞的最北方，再以其蛇行狀佈點的方式往南佈點，如在圖三(b)中，當機器人行走至 *a* 點時，其目前所處的狀態爲向東行走的狀態，

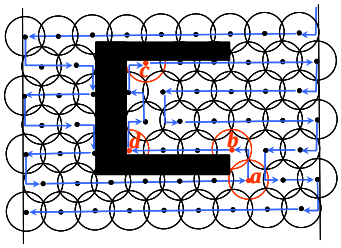
發現其北方有空洞且尚未有感測點存在時，機器人將往北方行走並放置感測點，在往北方行走的過程中，若遇到與自己所處向東行走的相反方向(即向西方向)的區域也存在空洞時，如圖中的 *b* 點，則機器人將往此方向行走，如此，對於機器人北方的空洞，機器人可以沿著空洞的邊緣一直行走至此空洞的最北方，如圖中的 *c* 點，然後再繼續採用由北往南的東西向蛇行狀佈點，將此空洞區域佈滿感測點。



圖二：簡單的蛇行狀佈點，容易有空洞或區域沒有佈點的情形。



(a) 簡易蛇行狀佈法產生的空洞有內凹的情形。



(b) 本論文的蛇行狀佈點克服內凹的障礙物之情形。

圖三：簡易蛇行狀佈法產生的空洞有內凹的情形。

因此，本論文所設計之克障的蛇行狀佈點，機器人每次行走除了使用簡易蛇行狀佈點(表一)中的欲行走方向(Prefer Direction 1)與往南行走方向(Prefer Direction 2)外，還需在此兩個欲行走方向行走之前，先後檢視兩個方向，亦即爲欲行走方向的反方向行走(如表二中的 Check Direction 1)與往北行走(如表二中的 Check Direction 2)，如此，機器人可有效地克服障礙物佈點，達到圖三(b)中，由 *a* 點可行走至 *b* 點，並沿著障礙物的邊界直到此空洞的最北方，*c* 點，再以蛇行狀佈點將此空洞佈滿感測點。由於遇到障礙物後，必須先處理障礙物所可能引發的空洞問題，因此表二中的確認行走方向(Check Direction)1 及 2 其優先權必須較表一中的欲行走方向爲高，當機器人檢視確認方向但卻因無障礙物而無法採用確認方向所建議的方向前進時，即可採用表一中的規則來進行簡易蛇行佈點之走法。

簡易蛇行狀佈點規則加入此兩個確認方向後，機器人開始執行蛇行狀佈點時，將以其在欲監測區域的東方或是西方邊界來決定其所處的狀態，不再須在欲監測區域最北方的限制，若機器人在欲監測區域的西方邊界出發，則機器人將先進入向東行走狀態，反之，若機器人在欲監測區域的東方邊界出發，機器人將進入向西行走狀態。

表二：爲了克服障礙物的確認方向。

States	Check Direction 1	Check Direction 2
East	←	↑
West	→	↑

機器人依序執行下列四個規則在網路中行走並佈放感測點，其中規則 1 及 2 的設計主要爲解決障礙物所引發的空洞問題，而規則 3 及 4 可使機器人以蛇行狀方式佈點，機器人每次在行走前，均會以下列四個規則從規則 1 至規則 4 來決定一行走方向，每次皆從規則 1 開始判斷，若失敗則判斷下一個規則，當規則 1 至規則 4 其中有一個規則執行成功，則機器人行走規則中所指定的距離，然後佈下感測點，並再次以下列四個規則來判斷下一次的行走方向：

規則 1：檢查其蛇行狀佈點欲行走方向 1 的反方向，亦即爲確認行走方向 1，是否不存在任何感測點，若否，則規則 1 執行失敗並中止此規則。若是，則機器人往確認行走方向 1 行走 $\sqrt{3}r_s$。若行走途中遇到障礙物與邊界時，則規則 1 執行失敗並中止此規則。

規則 2：檢查確認行走方向 2 的方向是否有空洞且不存在任何感測點，若否，則規則 1 執行失敗並中止此規則。若是，則機器人往確認行走方向 2 行走 $(3/2)r_s$。若行走途中遇到障礙物與邊界時，則規則 2 執行失敗並中止此規則。

規則 3：若欲行走方向 1 的方向可行走且未存在任何感測點，則機器人往欲行走方向 1 行走 $\sqrt{3}r_s$，若欲行走方向 1 的方向已存在感測點或是遇到障礙物與邊界時，則規則 3 執行失敗並中止此規則。

規則 4：若欲行走方向 2 的方向可行走且未存在任何感測點，則機器人往欲行走方向 2 行走 $(3/2)r_s$。若欲行走方向 2 也已存在感測點或是遇到障礙物與邊界時，則規則 4 執行失敗並中止此規則。

若機器人從規則 1 至規則 4 皆執行失敗，且佈點工作並未結束時，機器人將會走回上一佈點的位置，並再重新自規則 1 依序測試，若規則 1 至規則 4 還是沒有一個方向可行走，機器人回再上一佈點位置，直到規則 1 至規則 4 有一個規則可以成功繼續執行佈點爲止。

而由於上下兩排的感測點彼此水平座標相差 $(\sqrt{3}/2)r_s$ 單位距離，因此機器人執行規則 2 向北行走並佈點時並非直線向北行走，如圖(六)(b)中由 *d* 點走至 *c* 點，其分爲兩種向北行走的方式，第一種爲往北行走 $(3/2)r_s$ 後再往確認行走方向 1 方向行走 $(\sqrt{3}/2)r_s$，第二種爲往北行走 $(3/2)r_s$ 後再往確認行走方向 2 的反方向，即爲欲行走方向 1 方向行走 $(\sqrt{3}/2)r_s$，當機器人要往北行走時，會先執行第一種方式，若第一種失敗，機器人才會執行第二種方式，如表三所示，因此，結合表二與表三，規則 2 將融入表三的細節，成爲規則 2'，其規則如下：

規則 2'：檢查確認行走方向 2 的方向是否不存在任何感測點，若是，則機器人往北行走，當機器人欲往北行走時，機器人會先執行往北行走 $(3/2)r_s$ 後再往確認行走方向 1 方向行走 $(\sqrt{3}/2)r_s$，若失敗，則再嘗試爲往北行走 $(3/2)r_s$ 後再往欲行走方向 1 方向行走 $(\sqrt{3}/2)r_s$。若皆失敗，則此規則執行失敗且終止。

表三：確認行走方向 2 的細節。

States	Check Direction 2	
East	↰	↱
West	↱	↰

表四：欲行走方向 2 的細節。

States	Prefer Direction 2	
East	↳	↵
West	↵	↳

表五：克服障礙物之移動規則。

States	Check Direction 1	Check Direction 2		Prefer Direction 1	Prefer Direction 2	
East	←	↰	↱	➔	↳	↵
West	→	↱	↰	➔	↵	↳

同理，機器人執行規則 4 向南行走並佈點時也並非直線向南行走，同樣分爲兩種向南行走的方式，第一種爲往南行走 $(3/2)r_s$ 後再往欲行走方向 1 方向行走 $(\sqrt{3}/2)r_s$，第二種爲往南行走 $(3/2)r_s$ 後再往欲行走方向 1 的反方向，即爲確認行走方向 1 方向行走 $(\sqrt{3}/2)r_s$，當機器人要往南行走時，會先執行第一種方式，若第一種失敗，機器人才會執行第二種方式，如表四所示，因此，結合表二與表四，規則 4 將融入表四的細節，成爲規則 4'，其規則如下：

Rule4'：若欲行走方向 2 的方向可行走且未存在任何感測點，則機器人往南行走，當機器人欲往南行走時，機器人會先執行往南行走 $(3/2)r_s$ 後再往欲行走方向 1 方向行走 $(\sqrt{3}/2)r_s$，若失敗，則在嘗試爲往南行走 $(3/2)r_s$ 後再往確認行走方向 1 方向行走 $(\sqrt{3}/2)r_s$。若欲行走方向 2 也已存在感測點或是遇到障礙物與邊界時，則規則 4 執行失敗並中止此規則。

因此，整合表一的欲行走方向(Prefer Directions)與表二的確認行走方向(Check Directions)，並結合表三與表四，克服障礙霧之移動規則的詳細內容爲表五所示。

3. 資訊維護

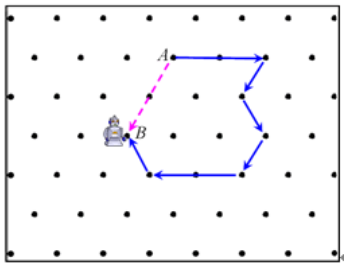
機器人在佈點的同時，將會爲每個感測點設定特殊的資訊，方便之後無線感測網路的維護；機器人會建立一個虛擬的二維座標系統，爲每個感測點設定虛擬座標(Virtual Coordination Location，簡稱 VCL)，虛擬座標以機器人所放置的第一個感測點爲原點，並以其第一個行走的方向爲 x 軸，x 軸每單位長爲 $(\sqrt{3}/2)r_s$，而 y 軸與 x 軸夾角爲 90°，y 軸每單位長爲 $(3/2)r_s$，由於機器人具有分辨其行走方向與距離的能力，因此，機器人能根據其所走的方向與距離，給予感測點其所應表示的虛擬座標，如此，佈置於監測區域的感測點皆有其虛擬座標來代表其在此欲監測區域中的位置。

而機器人在佈點時，除了給予每個感測點其虛擬座標外，每個感測點還需紀錄一個計數值(Counter Value)，此計數值主要目的在於使感測點皆具有追蹤機器人位置資訊的能力，透過計數值的維護，在感測點電量即將耗盡或是偵測到其他感測點損毀時，可以利用計數值追蹤機器人的位置資訊來通知機器人至待修復區域進行修復。計數值共包含兩個值，<廣播 ID, 步數>，廣播 ID 初始爲零，其紀錄此感測點收到機器人執行廣播封包的次數，此外，步數主要爲讓感測點追蹤機器人的資訊，機器人在佈點時，被放置的每個感測點其所記錄的步數值將隨機器人佈點的順序增加，故當廣播 ID 爲零時，感測點所記錄的步數值越大，表示此感測點越靠近機器人所在的位置，因此機器人目前所在位置的感測點必定紀錄最大的步數。

由於機器人在佈點時，會將邊界旁的感測點與障礙物旁的感測點設定爲邊界點(Boundary nodes)，因此，除了邊界點外，區域中的感測點會收集且維護其周圍六個鄰居的資訊，只有邊界點的鄰居資訊是少於六個鄰居資訊，因此，感測網路中的感測點可以藉由鄰居資訊來偵測是否有感測點損毀或是沒電；在感測網路經過長時間的偵測之後，感測點可能會發生故障或是耗盡電量而需要修復，由於感測網路中的每個感測點皆能利用計數值來追蹤機器人目前的位置，因此偵測到有感測點故障或是自己即將耗盡電量的感測點將會利用計數值的追蹤來傳送修復封包通知機器人進行修復，這些感測點將會廣播一個修復封包(Repair Packet)，由於每個感測點皆有其鄰居的資訊，因此，修復封包中包含有此待修復感測點的位置資訊與其鄰居中含有最大計數值的感測點，其他非具有最大計數值的感測點將會忽略此封包不再續傳，只有具有最大計數值的感測點會同樣再廣播此封包，因此，此修復封包將會依序傳送到具有最大計數值的感測點，亦即爲機器人所在的位置，機器人將會等待一段系統事先設定之可接受時間，收集許多待修復感測點的位置資訊後，便執行修復演算法(Repair Algorithm)以建立一修復路徑，並循路徑依序走訪且修復這些待修復的感測點。

雖然上述利用計數值追蹤機器人的方法可達到通知機器人修復毀損區的目的，然而其效率仍不如預期，如圖四所示，機器人在感測網路中行走，感測點利用計數值來追蹤機器人的路徑可能會過於蜿蜒導致追蹤路徑過長，因此，我們將設定一固定長度的門檻值，當機器人每行走固定長度，機器人將會發送修正路徑的封包來修正感測點的計數值，使感測點能改善追蹤機器人的路徑，使用泛流的方式將是最簡單且可修正所有感測點的計數值，使其能掌握機器人現有最新的位置，但卻將造成大量的控制封包成本並浪費感測點的電量，在控制封包成本與電量消耗之間的考量下，我們將使用 X 型的廣播方式，將修正封包以兩條交叉線的方式在無線感測網路中流動以修正計數值，在此我們將稱其封包爲 X 形修正封包(X-Correction Packet)，減少無線感測網路中感測點繞遠路的情形卻不致產生過多的泛流成本；當機器人已行走固定門檻值長度時，機器人會發送 X 形修正

封包來修正感測點的計數值，由於任一感測點均在六個方向佈有感測點, 在進行 X 形修正的時候, 機器人會選定其所在位置的六個方向其中的四個方向來廣播 X 形修正封包，X 型傳送之四個方向的選擇由感測點其所記錄的虛擬座標值來判別，圖五(a)所示爲 X 形修正封包傳送的規則，若機器人目前所在位置座標爲(s, t)，且假設收到 X 形修正封包的感測點其座標爲(x, y)，以下我們依機器人行走的方向分三種情況來討論那些感測點該將收到的 X 形修正封包繼續廣播出去。情況 I:當機器人是從 d_1 或 d_4 方向走至(s, t)，則滿足下列條件的感測點會將此封包再次廣播給鄰居爲$(x-s)=-(y-t)$或$y=t$；情況 II:當機器人是從 d_0 或 d_3 方向走至(s, t)，則滿足下列條件的感測點爲$|x-s|=|y-t|$會將此封包再次廣播給鄰居；情況 III:當機器人是從 d_2 或 d_5 方向走至(s, t)，則滿足下列條件的感測點爲$(x-s)=(y-t)$或$y=t$會將此封包再次廣播給鄰居。



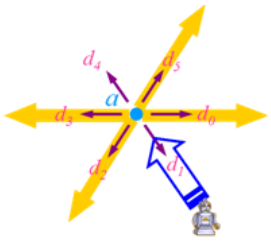
圖四：感測點追蹤機器人路徑過長的情形。

在 X 形修正封包中帶有機器人所在的位置、廣播 ID、步數與機器人行進的方向，廣播 ID 代表機器人發送 X 形修正封包的次數，而步數以機器人目前所在感測範圍內之感測點其記錄步數值最大值爲起始值，當機器人發送 X 形修正封包後，收到此封包的感測點，將會複製封包中的廣播 ID，且將此封包中所記錄的步數紀錄下來後，並依上述三個情況之規則決定是否將封包內之步數減 1 後並廣播此封包。

以圖五(b)爲例，左圖爲機器人行走至 A 點的情形，機器人是以 A 點的 d_i 方向靠近 A 點，若此時機器人發送 X 形修正封包，圖中 A 點座標爲$(9, 3)$，當 A 點發送 X 形修正封包時，此封包的資訊有 A 點的位置資訊$(9, 3)$、廣播 ID=1，步數=63，其鄰居均將計數值更正爲 62，而收到此封包且虛擬座標滿足$(x-9)=-(y-3)$的有鄰近感測點$(10, 2)$、$(8, 4)$，滿足 $y=3$ 的有鄰近感測點$(11, 3)$、$(7, 3)$，此四個感測點將把廣播 ID 改爲 1，步數減 1 爲步數=62，然後此四個感測點再次將此封包廣播出去，封包資訊爲 A 點的位置資訊$(9, 3)$、廣播 ID=1，步數=62，其鄰居中收到此封包且虛擬座標滿足$(x-9)=-(y-3)$的有$(11, 1)$、$(7, 5)$，滿足 $y=3$ 的有$(13, 3)$、$(5, 3)$，此四個感測點也更新自己的計數值，廣播 ID 改爲 1，步數改爲 61，並再將此封包廣播出去，依此類推。

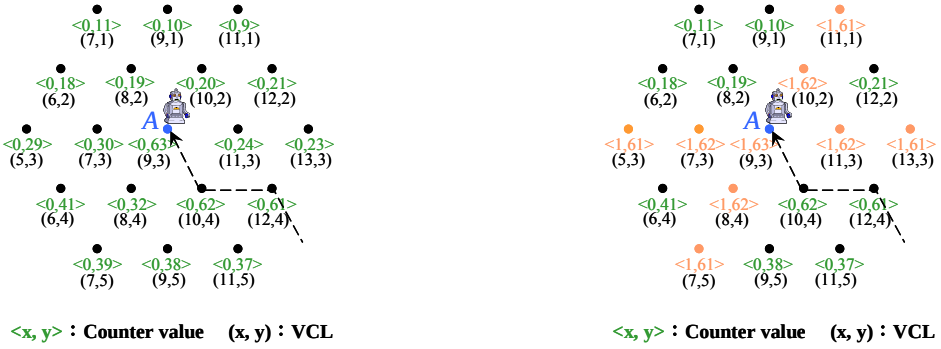
X 形修正方法的效能如圖六所示，在圖六(a)所示，黑色虛線爲機器人的行走路線，機器人目前位置爲感測點 A，由於每個感測點皆有其周圍鄰居的資訊，並得知其鄰居相對位置，機器人將以其所在感測點爲中心，並以其行走方向爲基準，以 X 型擴散此封包，收到此封包的感測點將會更新其廣播 ID 與步數，此封包將擴散到遇到邊界點爲止，邊界點將不會再擴散此封包，因此，當感測點欲傳送修正封包通知機器人進行修復時，將會把修正封包往其鄰居中計數值最大者傳送，亦即最大廣播 ID 且最大步數值的感測點傳

送，逐步傳送給機器人，如圖六(b)所示，橙色爲傳送 X 形修正封包的感測點，以感測點 *A* 爲中心，並依序更新其計數值，其中橙色的感測點皆會發送 X 形修正封包，因此其鄰居若聽到此 X 形修正封包也會更新其計數值，在此爲表達方便，僅以橙色點來說明 X 形修正封包，由圖六可明顯看出，待修復感測點 *B* 其追蹤機器人的路徑明顯減短。



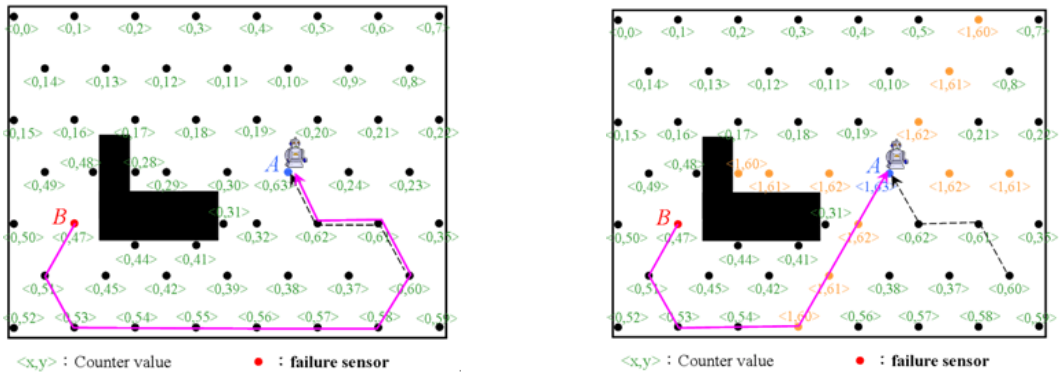
VCL of node a ： (s, t)
Receiver：(x, y)
If the direction that the robot lastly moved is
d_0,d_3：if $|x\text{-}s| = = |y\text{-}t|$
forward the packet
d_1,d_4：if $(x\text{-}s) = = -(y\text{-}t)$ or $y = = t$
forward the packet
d_2,d_5：if $(x\text{-}s) = = (y\text{-}t)$ or $y = = t$
forward the packet

(a) X 形修正封包的規則。



(b) X 形修正方法的例子。

圖五：X 形修正方法。



(a) 感測點 *B* 追蹤機器人路徑過於彎曲且長。

(b) 以 X 形修正封包來修正感測點的計數點。

圖六：利用 X 形修正封包來修正感測點追蹤機器人路徑過於彎曲且長的情形。

以下我們將詳述在無障礙物環境與有障礙物環境下，修復階段的演算法。

四、電量保存之修復方法

機器人每固定時間若收集到固定數量的感測點後即執行修復階段的演算法去修復感測點，當機器人收到許多感測點的修復封包後，其依這些感測點的虛擬座標執行本論文所研發之修復演算法(Repair algorithm)，規劃出一條行走省電且迅速之修復路徑來依序修復感測點。我們所設計的修復演算法將分爲沒有障礙務與存在障礙物兩種環境中執行，以下，我們將詳述其演算法的內容。

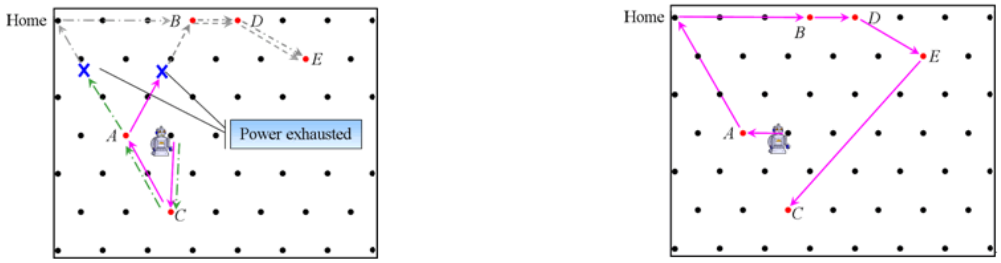
```
   Void travel ( int n, const number W[ ], minlength, Energy(R) )
   {
1    /* n: the number of nodes ( including Home )
2       W[] : adjacency matrix
3       R: the location of Robot
4       V: set of all the nodes ( including Home and R )
5       U=all subsets of V which includes Home
6       D[vi][A] =length of a shortest path from v_i to v_1
                  passing through each vertex in A exactly once
7       Energy(R): the energy of Robot at R
8    */
9    index i, j, k
10   number D[1..n][subset of V-{R}]
11   for (i=2;i<=n,i++)
12     D[i][ϕ]=W[i][1]
13   for (k=1;k<=n-1;k++)
14     for (all subset A ⊆ V-{R} containing k vertices)
15       for (i such that i≠R and v_i is not in A)
16       {
17         D[i][A]=minimum( W[i][j]+D[j][A-{v_j}])
                        j:V_j∈A
18       }
19   minlength=minimum(D[j][V-R-Home])
                    j∈A,j≠R
20   if (Energy(R) <= minlength*一單位距離所花的電量)
21   {
22     minlength=minimum(D[j][V-R])
23     if (Energy(R) <= minlength*一單位距離所花的電量)
24     {
25       for (k= n-1;k>=1;k--)
26         for (all subset A ⊆ U-{R} containing k-1 vertices )
27           if (Energy(R) >= minlength *一單位距離所花的電量)
28           {
29             minlength=minimum(D[j][V-R])
30             break
31           }
32       }
33   }
   }
```

圖七：考量機器人電量之修復演算法。

1. 沒有障礙物環境

在沒有障礙物的環境下，機器人只要收到待修復感測點的虛擬座標，即可得知其目前所在位置與這些感測點彼此的距離，以及其與感測點到家的距離，因此，當機器人收集多個待修復感測點的虛擬座標後，即可以 Dynamic programming 演算法計算出修復所有感測點的最短路徑，圖七爲修復階段(Repair Phase)與回家階段(Home Phase)的演算法，其中 1-19 爲修復階段的演算法，可求得一條行經所有待修復感測點的最短路徑，若機器人剩餘的電量能行經所有待修復的感測點，並在修復完最後一個感測點後其剩餘電量仍足夠回家時，則機器人將立刻執行此修復路徑。但若機器人的剩餘電量不足夠走完此路徑或走完此路徑卻不足電量走回家時，則機器人將執行圖七之 20-33 的回家階段演算法，

其目的在於尋找另一條可讓機器人回家充電且在其回家充電前盡量修復較多感測點的路徑。



(a) 由於機器人的電量有限，在執行修復的過程中，機器人可能耗盡電量。

(b) 將家納入機器人巡邏與修復演算法的考量，使機器人可以在其電量耗盡前，及時回家補充電量。

圖八：以機器人所剩餘的電量來看家存在的重要性。

如圖七之 22，當機器人電量不足夠以最短路徑修復所有待修復的感測點時，機器人將家納入最短路徑演算法的考量，求得一條可回家充電且修復所有待修復感測點的路徑，如圖八(a)所示，$O \rightarrow C \rightarrow A \rightarrow$Home$\rightarrow B \rightarrow D \rightarrow E$ 爲修復所有待修復感測點且回家充電的最短路徑，但若機器人其剩餘電量，在此最短路徑中，仍無法足夠經由 C、A 且走回家時，機器人將尋找另一條路徑，透過圖七之 25-32 步驟之執行，使機器人除了能在其電量耗盡前走回家充電外，並盡可能在回家充電的途中修復較多的感測點，如圖八(b)所示，機器人回家的過程中，其剩餘電量仍足夠先修復感測點 A 後再回家充電。

2. 有障礙物環境

在機器人佈建無線感測網路的同時，便知道環境中是否有障礙物存在，若有障礙物，則修復時將啓動本節所發展之演算法來建立其修復之行走路徑。在有障礙物的環境中，雖然機器人收到待修復感測點的虛擬座標資訊，但可能因障礙物的阻隔，使機器人無法由虛擬座標資訊來得知彼此之間真正的距離，與每個感測點與家的真正距離，這將可能導致機器人因估計路徑的長度有誤，產生機器人在行走路途中耗盡電量且來不及回家的情形，但要得知待修復感測點其所有彼此之間的距離又需傳送 $n!$次的封包，其中 n 爲待修復感測點的數量，在待修復感測點剩餘電量與封包成本的考量下，我們將仍採用圖七之演算法，首先我們求得一條修復路徑，但因機器人不知此路徑中點與點彼此之間的真正距離，因此機器人將在行走前先發送一探測封包(Probe Packet)，此探測封包將會模擬機器人修復所有待修復感測點的路徑，藉由探測封包封包行走的過程得知修復路徑中，前後待修復感測點彼此之間的實際距離，此外，障礙物也影響感測點與家的距離無法預測，故當機器人執行完佈建後，由最靠近家的感測點將家的位置資訊泛流給所有感測點，使每個感測點能得知其到家的距離與路徑，因此，當機器人以圖七中的演算法求得一條路徑後，將傳送探測封包模擬機器人行走的路徑，每個待修復的感測點收到此探測封包時，將會紀錄其回家的步數以及其距離上一個感測點的步數，藉由這些資訊，機

器人將可正確地判斷其電量是否足夠行走至下一個待修復感測點或回家。其演算法如圖九所示，當欲監測區有障礙物時，機器人將 obstacle_flag 設爲 true，利用圖七的修復演算法求得一條修復路徑後，機器人將執行下列演算法，首先機器人發送一探測封包模擬此修復路徑，並求得機器人剩餘的電量最多可行走至此修復路徑中的哪一個感測點且能回家充電，若機器人剩餘電量只能行走至 v_i 且回家充電，則將家插入此路徑中 v_i 的後面。

Input：repair path(v_0, v_1,…, v_{n-1}), v_0 =*location of robot*

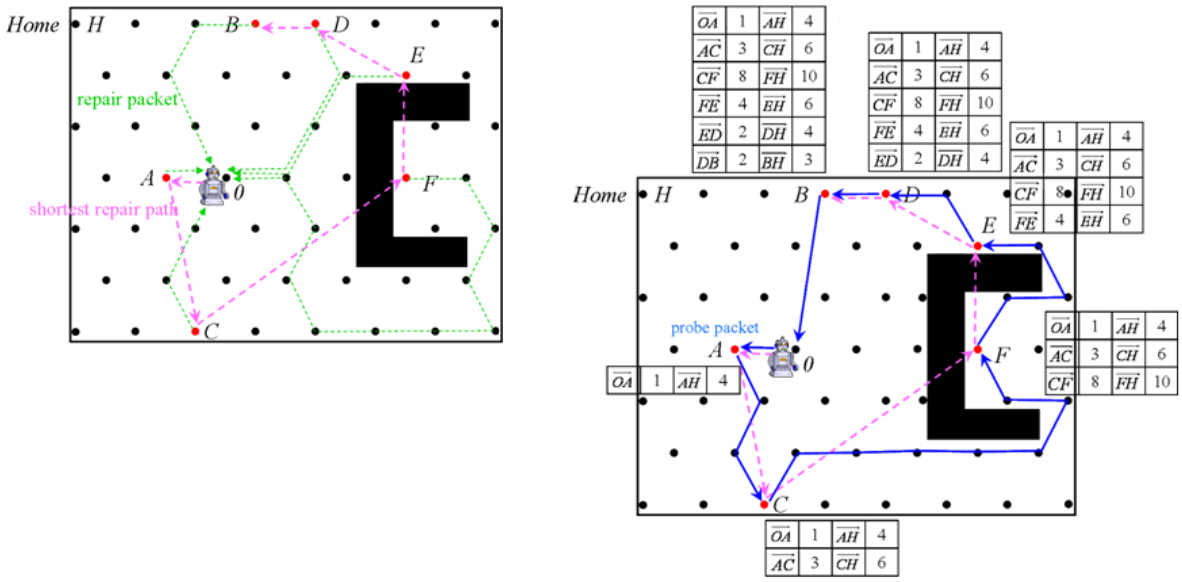
```
1   if( obstacle_flag = true )
2   {
3         transmit a probe packet to simulate the repair path
4         simulate_energy = Energy(R)
5         i=0
6         for( i=0 ; temp < 0 ; i++ )
7         {
8               simulate_energy = simulate_energy – energy of v_i to v_{i+1}
9               temp = simulate_energy – energy of v_{i+1} to Home
10        }
11        insert Home behind v_i
12        repair path = v_0, v_1,…, v_i, Home, v_{i+1},…, v_{n-1}
13  }
```

圖九：探測流程之演算法。

如圖十(a)所示，待修復感測點發送一修復封包，此修復封包中將定義鄰近感測點所記錄之計數值最大者之感測點爲代傳點(Forwarding Node)，如此修復封包便可以較短的路徑來通知機器人待修復之訊息，機器人收集待修復感測點 的位置資訊，利用修復階段演算法計算出無障礙空間的最短修復路徑，爲紅色路徑所示，但因障礙物的存在使得機器人無法直線行走導致此路徑的長度無法真正的估量，因此，爲了確切得知路徑的距離，以防機器人估計路徑實際長度錯誤，使機器人行走至一半而耗盡電量的情形發生，如圖十(b)所示，機器人將先發送一個探測封包來模擬其行走的路徑，此探測封包含有機器人剩餘電量所能行走的距離，供待修復感測點計算機器人的電量是否足夠，當探測封包每行經一個待修復感測點後，將會記錄其封包實際行走的步數與其所在位置回家的步數，如探測封包由感測點 O 行經感測點 A 時，此探測封包將會紀錄$\overline{OA}$的步數與由感測點 A 回家的步數，而當探測封包由感測點 A 行經感測點 C 時，探測封包也會紀錄 $\overline{AC}$ 的步數與由感測點 C 回家的步數，依此類推，直到當探測封包傳送到感測點 B 時，若感測點 B 估量機器人剩餘電量足夠走至感測點B且機器人走至感測點 B後其剩餘電量仍足夠由感測點 B 走回家，則感測點 B 將此探測封包傳回感測點 O，通知機器人實行此修復路徑。

在有障礙物的環境中，雖然探測封包可以模擬機器人，但是當修復路徑因機器人電量不足而依據圖七之 20-33 的演算法改變時，因路徑已不同，每次估量不同的路徑都必須再發送一次探測封包來估量路徑的長度，除了採用圖七之 20-33 的演算法外，在節省感測點電量與機器人快速修復的考量下，機器人在使用圖七之 1-19 的演算法求出一條不回家的最短路徑，若發送探測封包模擬後發現其電量不足，機器人將使用圖七之 22 求出一條可回家的最短路徑，再發送探測封包來估量此路徑的實際長度，若機器人剩餘的電量仍不足夠在此路徑中回家充電，則機器人將不再執行 23-33 的演算法，僅將家的排序

在 23 所求出的排序中往前插入，直到機器人能來得及在電量耗盡前充電爲止。因此同樣地，即使在有障礙物的環境中，機器人利用圖七求得一條修復路徑後，機器人將執行圖九之演算法來求得一條可回家充電的修復路徑。



(a) 機器人收集待修復感測點 的位置資訊，利用修復階段演算法計算出無障礙空間的最短修復路徑。

(b) 探測封包收集修復路徑實際的長度。

圖十：在有障礙物環境中，修復階段的執行方法。

五、效能分析

爲了評估本論文所設計的克服障礙物之佈點方法(Obstacle-Free Deployment Mechanism)與電量保存之修復方法(Power-Conservation Repair Mechanism)，我們將與[2](在此簡稱爲 CED)所設計的機器人佈點方法做比較。首先，我們先說明本論文的模擬環境，再探討本論文與 CED 之各參數的模擬結果。

表六：實驗參數

Parameter	Value
Communication range	40m
Sensing range	20m
Packet transmission cost	0.075J/s
Packet reception cost	0.030J/s
Idle cost	0.025J/s
Maximum energy consumption in motes	324J/hr
Total initial energy	32400J (100hr)

1. 實驗模型

在本論文中，我們參考 Berkeley motes [5] 之特性來設定相關參數，其詳細內容如表六：

其中機器人配帶有羅盤針來指引方向與具有 RF 可與網路中感測點通訊之無線設備，且機器人配帶有固定數量的感測點 [5]，其移動成本參考 Robomote I [4]爲 8.267J/m，我們假設機器人其速率爲 3m/s，以及其總電量爲 64800J。在環境方面，模擬環境大小爲 400*400m，家的位置則爲機器人的出發點，並假設在環境之左上角，此外，各模擬結果爲 10 個 independent runs 的平均值。

本模擬將分別在佈點與修復方面來評估效能，在佈點方面，將針對 (1) 機器人將整個監測區域放滿感測點的佈點時間(Deployment Time)、(2) 所花費的感測點數目(Number of Sensors)來評估佈點方法的效能，在修復方面，將針對 (1)損壞的感測點其從損壞至被修復的延遲時間(Delay Time)、(2)機器人其行走單位路徑長度內修復感測點點數的修復效能(Repair Efficiency)來評估修復機制的效能。

本模擬包含兩個部份，第一個部分我們將先檢視本論文所設計的客障礙物佈點機制的佈點效果，以及比較 ODR 與 CED [2]在無障礙物與有障礙物環境中，其將欲監測區域佈滿感測點所花費的佈點時間與感測點數目，在此，我們定義當網路中感測點的感測範圍達到欲監測區域面積的 98%，則機器人執行佈建機制結束。

第二個部分我們針對電量保存之修復方法中，因爲感測點利用計數值追蹤機器人位置資訊會有繞遠路的情形，我們將比較利用泛流修正(Flooding-Correction)、X 形修正(X-Correction)與線性修正(Line-Correction)三種修正計數值的方式，其所產生的控制封包成本以及修正感測點追蹤機器人路徑成效的影響，並透過模擬選擇機器人每行走多少距離傳送 X 形修正方法，能達到傳送較少次數的 X 形修正封包且有效修正感測點追蹤機器人的路徑；另一方面，我們將比較 ODR 與 CED 在網路中有感測點耗盡電量或損毀時，這些待修復感測點等待修復所需花費的平均延遲時間，最後，比較機器人分別利用 ODR 與 CED 機制下，機器人平均花費多少距離修復一個感測點的修復效率。

2. 佈點方法之效能分析

我們首先以理論值算出一個長方形監控區域達到全區感測覆蓋所需之理想佈點數量，再以此理想佈點數量來衡量我們的佈點演算法效能。假設欲監測區域的長爲 l，寬爲 w，感測點的感測半徑爲 r_s，第一列感測點個數爲 n_1，第二列感測點個數爲 n_2，而 h 表示所需的感測點的列數，Q 爲共需花費的點數；首先藉由 l 的長度算出第一排感測點與第二排感測點所需花費的點數，

$$\begin{aligned} &if\ (l \bmod \sqrt{3}r_s) > \frac{\sqrt{3}r_s}{2},\ n_1 = \left\lfloor \frac{l}{\sqrt{3}r_s} \right\rfloor + 2, n_2 = \left\lfloor \frac{l}{\sqrt{3}r_s} \right\rfloor + 1 \\ &if\ (l \bmod \sqrt{3}r_s) \le \frac{\sqrt{3}r_s}{2}, n_1 = n_2 = \left\lfloor \frac{l}{\sqrt{3}r_s} \right\rfloor + 1 \end{aligned} \tag{1}$$

再利用 w 的長度算出 h 值來求得共需多少排的 n_1 與 n_2，

$$h = \left\lfloor \frac{w}{\frac{3}{2} r_s} \right\rfloor$$
$$if\ h\ is\ an\ even\ number, Q = (\frac{h}{2} + 1) \times n_1 + \frac{h}{2} \times n_2 \quad (2)$$
$$if\ h\ is\ an\ odd\ number, Q = \left\lfloor \frac{h+1}{2} \right\rfloor \times (n_1 + n_2)$$

但因爲 w 並非剛好是 $\frac{3}{2} r_s$ 的倍數，產生欲監測區域最下方可能還有一排空洞，所以還需判斷下方是否產生空洞並加上 n_1 或 n_2 來填補此空洞，

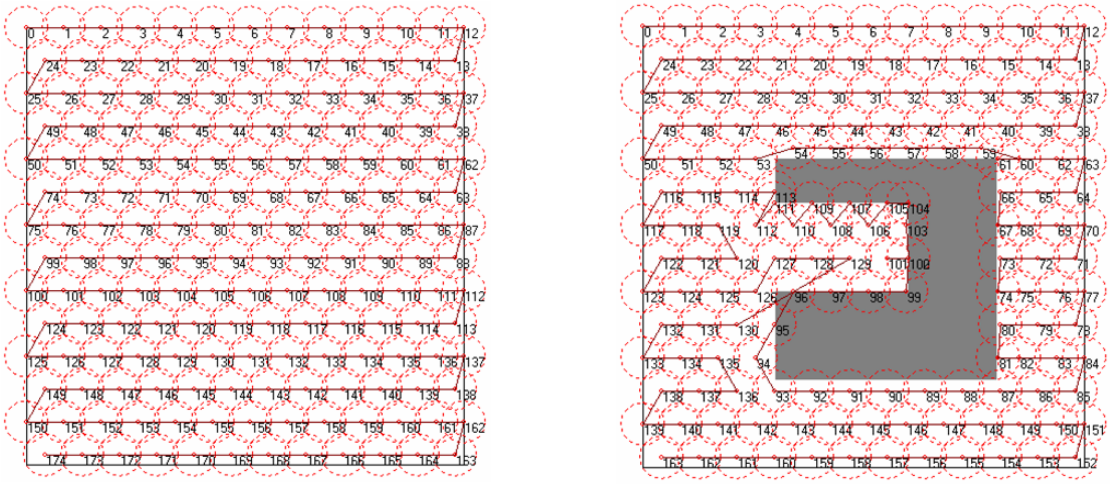
$$if\ (w \bmod \tfrac{3}{2} r_s) > \tfrac{1}{2} r_s$$
$$then\ if\ h\ is\ an\ even\ number,\ Q = Q + n_2 \quad (3)$$
$$if\ h\ is\ an\ odd\ number,\ Q = Q + n_1$$

依上述理想佈點公式(1)(2)(3)，我們將評估本論文所發展的佈建演算法其佈點數量之效能。我們定義佈點效率=理想點數/ODR 所花的點數，佈點效率越接近 1，則表示 ODR 之佈點演算法越接近理想值。在第一部份中，首先我們將先模擬出使用 ODR 佈點在有障礙物與無障礙物環境中的情形，圖十一(a)爲在無障礙物環境中使用 ODR 佈點的模擬程式所擷取的螢幕畫面，由圖中可看出機器人能以蛇行狀佈點使欲監測區域能以幾乎最少的點數來達到全區感測覆蓋的目的，在此 400m*400m 的環境中，數字爲機器人佈點的順序，機器人僅花費 175 個感測點即能將此區域佈滿，而由下列公式可算出此區域中將此區域佈滿且所花點數最少的理想點數個數爲：

$$l = w = 400, r_s = 20,$$
$$Because\ \ 400 \bmod (\sqrt{3} \times 20) = 18.94 > (\frac{\sqrt{3}}{2} \times 20)$$
$$n_1 = \left\lfloor \frac{400}{\sqrt{3} \times 20} \right\rfloor + 2 = 11 + 2 = 13,\ n_2 = \left\lfloor \frac{400}{\sqrt{3} \times 20} \right\rfloor = 12$$
$$h = \left\lfloor \frac{400}{\frac{3}{2} \times 20} \right\rfloor = 13$$
$$Because\ h\ is\ an\ odd\ number, Q = \left\lfloor \frac{13+1}{2} \right\rfloor \times (13 + 12) = 175$$
$$And\ (400 \bmod (\tfrac{3}{2} \times 20)) = 10 \le (\tfrac{1}{2} \times 20),$$
$$therefore\ \ Q = 175$$

因此，由上述公式可算出此環境中將此區域佈滿且所花點數最少的理想點數爲 175 個感測點，藉由佈點效率=理想點數/ODR 所花的點數，此圖中的佈點效率=175/175=1，可看出我們佈點與理想值之最少佈點數相同，圖十一(b)爲在障礙物環境中使用 ODR 佈點的模擬，可看出機器人採用 ODR 的克障礙物蛇行狀佈點機制能有效避開障礙物並以蛇行狀佈點法將欲監測區域佈滿感測點，在此環境中 ODR 花費 163 個點將此區域佈滿，而此環境中能將此區域佈點且花費最少點數的理想點數可使用切割法先將欲監測區域切

割成五個小區塊，再利用上述公式算出每區塊所需花費的點數並相加求得，所以算出此區域理想點數爲 161 個感測點，其佈點效率=161/163=0.98，可看出即使在有障礙物的環境中，ODR 佈點的效率也能達到近乎理想佈點的情況。



(a) 在沒有障礙物環境下的佈點結果。　　(b) 在有障礙物環境下的佈點結果。

圖十一：利用 ODR 方法來佈建網路的場景圖。

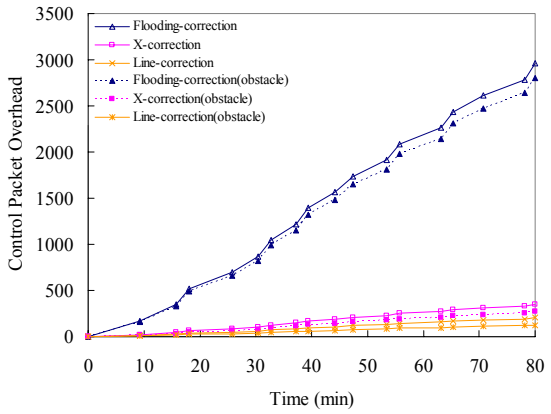
除了以理想情況來檢視我們所提出的佈點演算法之效率外，我們將比較 ODR 與 CED 在無障礙物環境與有障礙物環境中，其執行完佈建所花費的時間與所需花費的感測點數量，我們設計在障礙物環境中，障礙物形狀分別有長方形、L 型與 C 型，表七爲 CED 與 ORD 在上述不同環境中達到全區感測覆蓋的佈點時間，表八爲其分別所花費的感測點數量。從表七中顯示出，在沒有障礙物的環境中，CED 使欲監測區域達到全區感測覆蓋的佈點時間較 ODR 久，其原因在於 CED 佈點的條件使感測點佈點過密，導致感測點之間的重疊區域較大，因此需要花費較多的佈點時間才能達到全區感測覆蓋的要求，因此表八中可看出 CED 在沒有障礙物環境中的佈點時間較 ODR 大；在障礙物的環境中，CED 會因爲障礙物的影響而產生空洞，且機器人採用往其周圍最久沒行走的方向行走，空洞區域沒有有效的通知機器人至空洞修補的機制，因此從表七中可以看到機器人要達到全區感測覆蓋的時間花費相當長久，反之，ODR 的克服障礙物佈點機制幾乎能克服大部分的障礙物佈點，並對於障礙物邊界空洞的問題皆能在行走過程中即時填補，因此達到全區感測覆蓋的時間較 ODR 短，表八也顯示出 ODR 所花費的感測點數較 CED 減少許多。

表七：在不同環境下，利用 ODR 與 CED 方法所需的佈建時間。

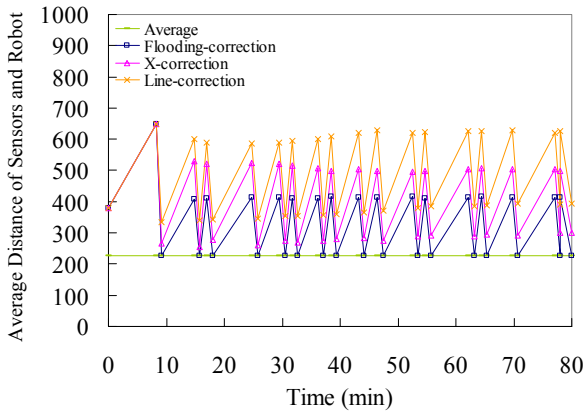
Environment	No obstacle	Square-shape obstacle	L-shape obstacle	C-shape obstacle
ODR	1933	2309	2164	2131
CED	3067	3592	3497	3764

表八：在不同環境下，比較 ODR 與 CED 方法所需的佈點個數。

Environment	No obstacle	Square-shape obstacle	L-shape obstacle	C-shape obstacle
ODR	174	171	160	163
CED	400	388	364	370



圖十二：在不同計數值修正方法下，各方法的控制封包成本比較圖。



圖十三：使用不同修正機制下，機器人追蹤損壞感測點的平均距離。

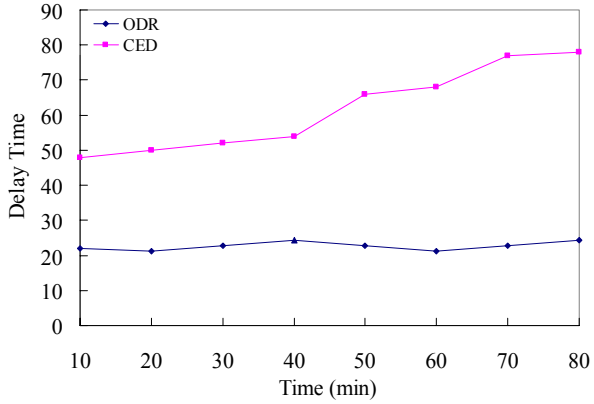
3. 修復方法之效能分析

在第二部份中，首先我們利用圖十一(a)的環境來評估 X 形修正方法的效能。由於機器人在網路中行走時會利用計數值留下其行走軌跡，但感測點利用計數值追蹤機器人的路徑可能會有繞遠路的情形，圖十二爲我們以泛流修正、X 形修正與線性修正三種修正計數值的方式來評估其對於控制封包成本的影響，其中線性修正的傳送將設定以機器人所在位置之感測點的 y 座標爲準，當修正計數值的封包傳送出去後，只有具有同樣 y 座標值的感測點才會繼續廣播此修正計數值的封包；我們分別在無障礙物與有障礙物的環境中模擬，機器人執行完克服障礙物之佈點方法後直接進入電量保存之修復方法，假設機器人每 10 分鐘執行電量保存之修復方法且每行走 400m 即發送一次修正機制，網路環境設定平均每 10 分鐘有 5 個隨機選出之損壞感測點需要修復。由圖中可以看出線性修正的控制封包成本最小，泛流修正機制的控制封包成本最大，雖然 X 形修正機制的控制封包成本大約爲線性修正的 2.5 倍，但相對於泛流修正機制則減少很多，泛流修正機制的控制封包成本大約爲 X 形修正機制的 37 倍；此外，爲了評估此三種修正機制對每個感測點其利用計數值追蹤機器人位置的路徑修正貢獻，我們計算網路中每個感測點其利用計數值追蹤機器人的路徑長度之平均值，並比較感測點追蹤機器人的理想最短平均路徑長度，在圖十三中，Y 軸即爲感測點到機器人的平均長度，我們可以看出，在啓動此三種修正機制的中間過程中，隨著機器人移動，感測點追蹤機器人的路徑長度皆會逐漸增加，但每當泛流修正機制啓動，必能使網路中每個感測點皆能以最短路徑通知機器人，而 X 形修正機制的啓動雖然無法使網路中每個感測點追蹤機器人的路徑達到理想的最短

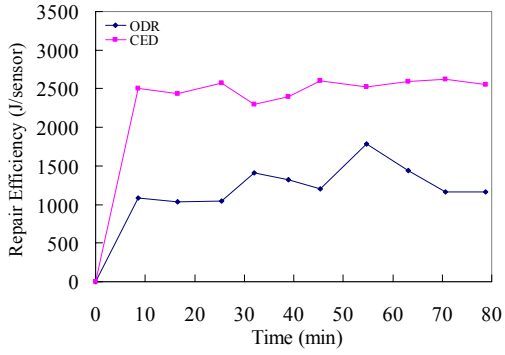
路徑，但能使此平均追蹤路徑降低至約 280m，並保持低於 520m，而線性修正機制也能使感測點追蹤機器人路徑降低至約 372m，但若以修正的效率來看，修正效率 =(節省路徑/封包數)，則 X 形修正方式的修正效率約爲 17.26，而線性修正的修正效率約爲 15.75，可見線性修正的成效不如 X 形修正明顯。經由圖十二與圖十三可以看出，X 形修正方法能在產生較少的控制封包成本下，達到使感測點追蹤機器人的平均距離較短。

依據上述，我們將以網路中存在有 C 形障礙物來做下列各參數的模擬。當機器人佈完點後，若經過一段偵測時間，網路中有感測點損毀或耗盡電量時，機器人將進入電量保存之修復方法時，我們假設機器人每 10 分鐘內若收集到 5 個待修復的感測點資訊時，則執行電量保存之修復方法，若 20 分鐘內還未收集到 5 個待修復的感測點，爲了避免待修復感測點等待修復的延遲時間過長，因此機器人仍須執行修復的機制。圖十四中顯示出，ODR 其延遲時間較 CED 減少許多，其原因在於 ODR 爲感測點主動通知機器人修復，且機器人在 10 分鐘內必會執行修復機制，因此控制住待修復感測點等待修復的延遲時間，而 CED 爲待修復感測點被動地等待機器人修復，須等待機器人下次行經這些待修復感測點的位置時，機器人才會發現這些損壞感測點並進行修復。

除此之外，因 ODR 爲感測點主動通知機器人修復使得損壞感測點等待修復的延遲時間較低外，ODR 其修復機制中利用演算法求得最短修復路徑的方法，也大大的降低機器人其執行修復所需行走的長度，圖十五我們比較機器人其平均行走多少路徑長度來修復一個感測點的修復效率，從圖中可以看出，在 ODR 中，因機器人除了知道損壞感測點的位置資訊外，機器人還利用演算法算出其最佳的修復路徑，使行走至這些損壞感測點的路徑能較短，因此 ODR 其修復效率較低，CED 對於損壞感測點因沒有良好的修復路徑，因此修復效率較高。



圖十四：ODR 與 CED 在延遲時間上的比較圖。



圖十五：ODR 與 CED 在修復效能上的比較圖。

六、 結論

本論文所研發的克服障礙物之佈點方法能以蛇行狀的方式，在花費最少的感測點個數下將欲監測區域達到全區感測覆蓋的目的，而當佈建區中存在不可預知之障礙物時，我們亦提出處理障礙物之規則予以克服。機器人在行走過程中利用計數值留下行走軌跡，使無線感測網路中的感測點皆能依機器人行走之軌跡來追蹤機器人的位置，以便通

知機器人進行修復。而 X 形修正方法以較少的控制封包的條件下，解決了感測點追蹤機器人路徑有繞遠路的情形，當感測點即將耗盡電量或是偵測到損毀區域時，X 形修正將有助於追蹤機器人路徑之學習，使修復通知的封包能以較短路徑來快速及省電地通知機器人進行修復；機器人收集待修復感測點的位置資訊後，執行電量保存之修復方法使機器人能以動態、快速、省電及有效率的方式往修復區移動，且機器人在進行修復的過程中，能以剩餘電量來安排較佳行走路徑。即使修復途徑中遭遇障礙物而使修復路徑無法預估機器人的電量是否足夠及路徑是否較佳，本論文亦使用探測封包的技術來達到最短修復路徑及解決機器人電量耗盡無法回家充電及補充新的感測點的問題，使機器人的修復工作得以持續、省電、有效率的方式進行。

參考文獻

[1] Batalin, M. A. and Sukhatme, G. S., "Efficient Exploration without Localization," *IEEE International Conference on Robotics and Automation* (*ICRA*), Taipei, Taiwan, 2003: pp. 2714–2719.

[2] Batalin, M. A. and Sukhatme, G. S., "Coverage, Exploration and Deployment by a Mobile Robot and Communication Network," *Proceedings of the International Workshop on Information Processing in Sensor Networks* (*IPSN*), Palo Alto, 2003: pp. 376-391.

[3] Batalin, M. A., Sukhatme, G. S. and Hattig, M., "Mobile Robot Navigation using a Sensor Network," *Proceedings of the IEEE International Conference on Robotics & Automation* (*ICRA*), New Orleans, LA, 2004: pp.636-642.

[4] Ganeriwal, S., Kansal, A. and Srivastava, M. B., "Self Aware Actuation for Fault Repair in Sensor Neworks," *IEEE International Conference on Robotics and Automation* (ICRA), 2004.

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

[6] Huang, Q., Lu, C., and Roman, G. C., "Reliable Mobicast via Face-Aware Routing," *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies* (*INFOCOM*), 2004.

[7] Karp, B. and Kung, H. T., "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," *Proceedings of International Conference on Mobile Computing and Networking* (*MobiCom*), Boston, Massachusetts, 2000: pp. 243-254.

[8] Mataric, M. J., "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*, (9:2-3), 1997: pp. 323-336.

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

[10] Sibley, G. T., Rahimi, M. H., and Sukhatme, G. S., "Robomote: A Tiny Mobile Robot Platform for Large-Scale Sensor Networks," *IEEE International Conference on Robotics and Automation* (ICRA), Washington DC, 2002: pp. 1143-1148.

[11] Wang, Y. C., Hu, C. C., and Tseng, Y. C., "Efficient Deployment Algorithms for Ensuring Coverage and Connectivity of Wireless Sensor Networks," *Wireless Internet Conf.* (*WICON*), 2005.

[12] Zou, Y. and Chakrabarty, K., "Sensor deployment and Target Localization in Distributed Sensor Networks," *ACM Transactions on Embedded Computing Systems* (*TECS*), (3:1), 2004: pp.61-91.