METODY PLANOWANIA ŚCIEŻKI OPARTE NA RASTROWYM MODELU OTOCZENIA

Ze względu na sposób reprezentowania otoczenia podczas poszukiwania optymalnej ścieżki można wyróżnić dwie podstawowe klasy metod planowania:

- metody sieciowo-grafowe
- metody rastrowe

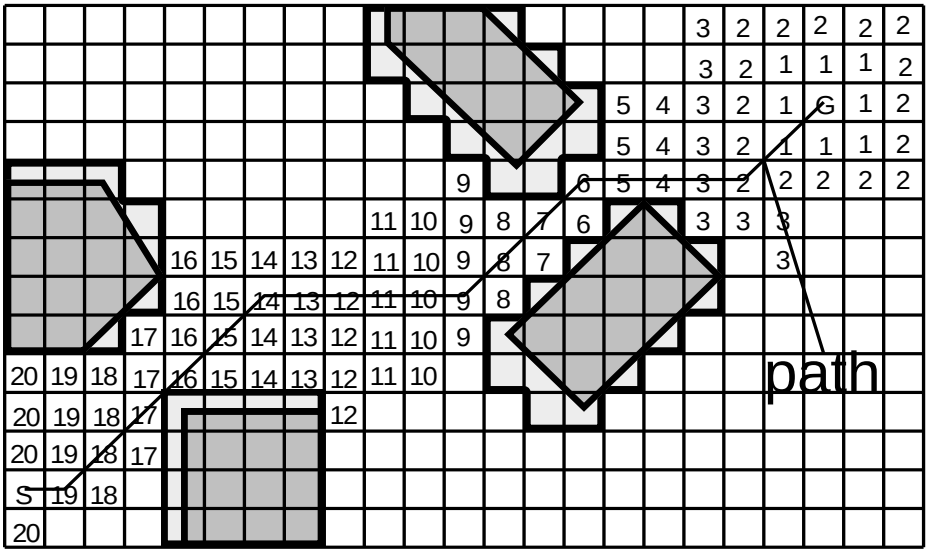
Metody sieciowo-grafowe opierają się na transformacji pierwotnego opisu otoczenia na graf połączeń określający sieć możliwych dróg, a następnie poszukiwania w takiej sieci najlepszego połączenia. Operują opisami wysokiego poziomu, (czyli takimi, których struktura zależy bezpośrednio od konfiguracji przeszkód i natury otoczenia).

Metody rastrowe wykorzystują podział pierwotnej przestrzeni na komórki równomiernej siatki i poprzez odpowiednie kodowanie tych komórek wyznaczają sposób (kierunek) poruszania się robota w każdym punkcie przestrzeni swobodnej.
Struktura mapy rastrowej nie zależy od natury otoczenia (konfiguracji przeszkód), co powoduje, że jest ona stosunkowo mało wrażliwa na niepewność informacji o otoczeniu podczas transformacji pierwotnego opisu otoczenia na postać siatki, a ponadto umożliwia także proste modelowanie tej niepewności.

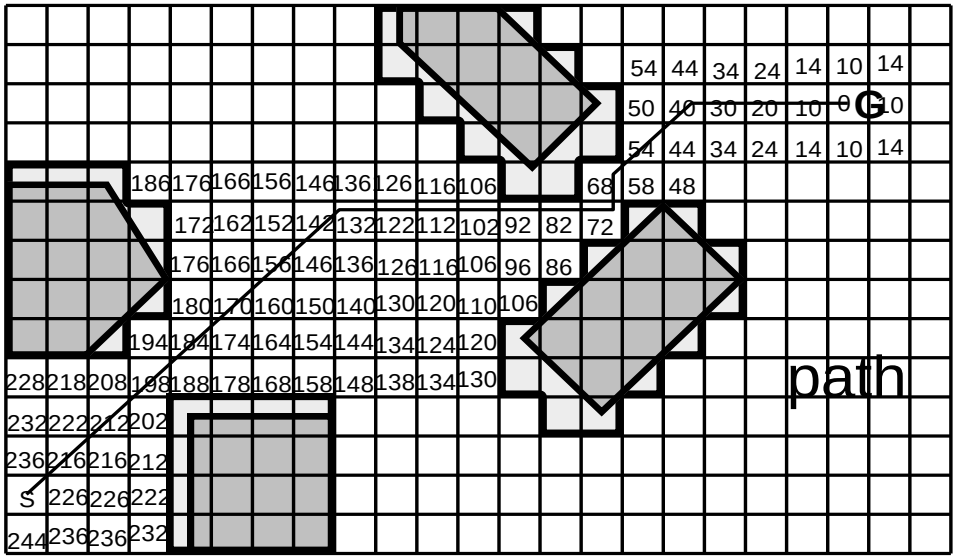
Dodatkowa atrakcyjność metod rastrowych polega na łatwości modyfikowania kryterium optymalności poszukiwanej ścieżki przez zmianę algorytmu kodowania rastra.

Metoda transformaty odległości

Metoda transformaty odległości (*Distance Transform Method*) została zaproponowana przez Kanga i Byrne. Polega ona na propagacji odległości, począwszy od komórki, w której znajduje się cel, przez całą przestrzeń swobodną (każda komórka ma ośmiu sąsiadów). Czoło powstałej w ten sposób fali opływa przeszkody i przechodzi przez wszystkie wolne komórki zaznaczając w nich ich odległość od celu (wartość transformaty odległości). Przy takim kodowaniu najkrótsza ścieżka do celu przebiega po malejących wartościach transformaty, przechodząc po najbardziej stromym spadku.



Rys.1. Ośmio-spójna transformata odległości wykazująca niejednoznaczności



Rys.2. Ośmio-spójna transformata odległości oparta na normie euklidesowej

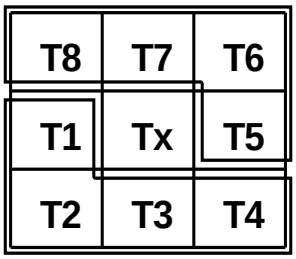
W metodzie z rys.1 występuje problem niejednoznaczności, spowodowany tym, że komórki sąsiadujące diagonalnie mają tą samą wartość transformaty, co komórki sąsiadujące ortogonalnie. Problem można rozwiązać stosując ośmio-spójną transformatę opartą na rzeczywistej odległości (np. $10 \cdot \sqrt{2} = 14$), która generuje jednoznaczną ścieżkę (rys.2).

Algorytm propagacji transformaty odległości, zastosowany na obszernej siatce, ma znaczną złożoność obliczeniową (graf utworzony na podstawie struktury danych siatki zawiera pętle). Dlatego stosuje się tu ***algorytmy skanujące***, takie jak przedstawiony na rys.3.

W tym algorytmie skanowanie jest realizowane w dwu kierunkach: do przodu - transformata Tx kodowanej komórki jest obliczana na podstawie transformat sąsiadów T1, T2, T3, T4, oraz do tyłu - T_x jest obliczana dla T5, T6, T7, T8.
Sposób liczenia transformaty T_x określa zależność:

$$T_x^{n+1} = \min_i \left[T_x^n, T_{xi} \right] \quad \text{dla } i \in [1,2,3,4] \text{ lub } i \in [5,6,7,8] \tag{1}$$

gdzie: T_{xi} =(T_i +odległość)*czynnik

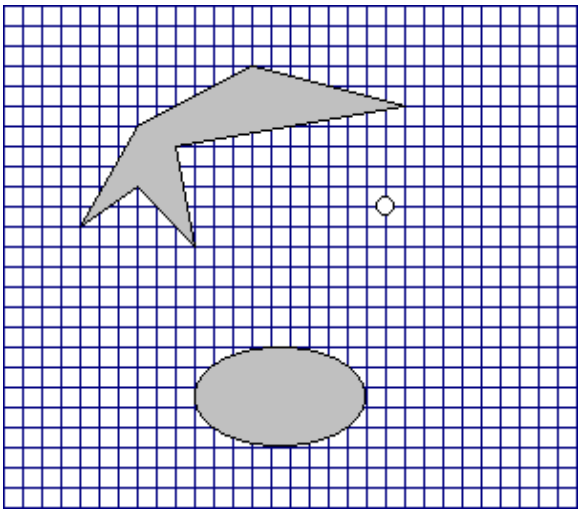


Rys. 3. Ilustracja skanowania siatki przy użyciu transformaty ośmio-spójnej. T1-T4 – Transformaty sąsiadów dla skanowania do przodu,
T5-T8 – Transformaty sąsiadów dla skanowania do tyłu.

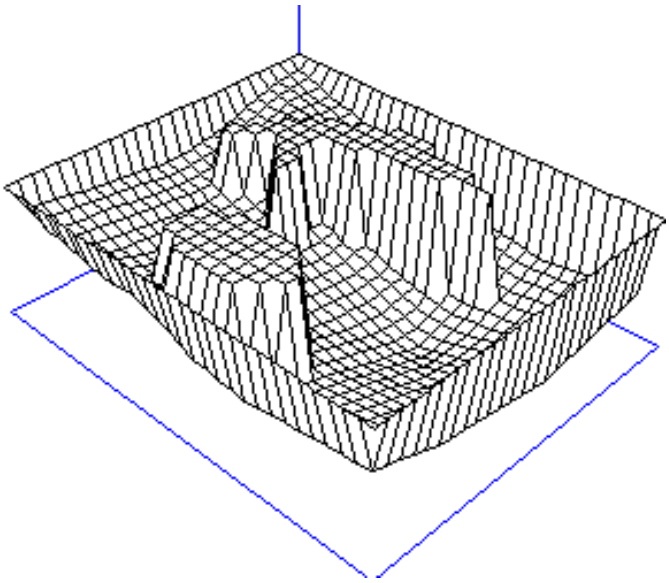
Aby uniknąć testowania przeszkód należy im przypisać duże wartości. W tym celu mnoży się wartość (T_i +odległość) wyliczoną dla danej komórki „*i*" przez czynnik określający prawdopodobieństwo zajętości komórki. Przykładowo, dla czynnika 1 (komórka wolna) i 3 (komórka zajęta) transformata wokół przeszkód ma wystarczająco strome nachylenie, więc ścieżki będą omijać przeszkody. Zbieżność podanego algorytmu jest całkiem dobra - wystarczają dwie do pięciu iteracji.

Algorytm planowania ścieżki dla transformaty odległości przedstawia się następująco:
1. Przejdź do komórki startu.
2. Sprawdź wszystkich sąsiadów, czy ich wartości są niższe

If komórka o niższej wartości nie została znaleziona **then** zakończ program, ponieważ szukana ścieżka nie istnieje.
Else (ścieżka istnieje)
    Dodaj komórkę startową do listy (ścieżki)
    Repeat
        Sprawdź sąsiadujące komórki by znaleźć komórkę o najmniejszej wartości
            (używając sąsiedztwo cztero- lub ośmio-spójne)
        Przejdź do tej komórki
        Dodaj identyfikator komórki do listy
    Until cel został zdobyty
End
Return -flaga istnienia ścieżki i lista identyfikatorów komórek ścieżki



Rys.5. Mapa rastrowa otoczenia określająca konfigurację przeszkód i położenie celu.



Rys.6. Mapa rastrowa rozkładu transformaty odległości dla konfiguracji otoczenia z rys.5.

Metoda transformaty ścieżki

Istotną wadą transformaty odległości jest fakt, że otrzymane ścieżki przechodzą zbyt blisko przeszkód ("too close" problem). Aby tego uniknąć Zelinsky wprowadził rozszerzenie nazwane transformatą ścieżki. W tej metodzie używa się fali, która jest kombinacją odległości od celu i miary wyrażającej poruszanie się zbyt blisko przeszkód. W efekcie tworzy to specyficzną transformatę wykazującą cechy metod pól potencjalnych. Wyliczenie tej transformaty przebiega dwuetapowo. Najpierw obliczana jest transformata przeszkód, w której komórki przeszkód są celami. W efekcie otrzymuje się funkcję kosztu „**obstacle(c)**” wyrażającą wpływ najbliższej przeszkody na komórkę **c**.

3	2	2	2	2	2	2	2
3	2	1	1	1	1	1	2
3	2	1	**0**	1	**0**	1	2
3	2	1	**0**	1	**0**	1	2
3	2	1	**0**	1	**0**	1	2
3	2	1	1	1	1	1	2

Transformatę przeszkód oblicza się tak jak transformatę odległości, ale z przeszkodami traktowanymi jak cel.

Następnie oblicza się drugą transformatę przy użyciu funkcji kosztu:

$$PT(c) = \min_{p \in P} \left(length(p) + \alpha \sum_{c_i \in p} \overline{obstacle}(c_i) \right) \qquad (2)$$

gdzie: $\overline{obstacle}(c_i)$ - odwrócona funkcja kosztu

$length(p)$ - długość ścieżki p

P - zbiór możliwych ścieżek od komórki c do celu

$\alpha \geq 0$ - waga określająca wpływ przeszkód na transformatę ścieżki.

Ta funkcja kosztów jest właśnie określana mianem transformaty ścieżki ***PT*** (path transform). Wartość przechowywana w transformacie ścieżki każdej komórki przedstawia minimalny koszt przejścia od tej komórki do celu. Ponieważ transformata ścieżki wyznacza koszty wszystkich możliwych ścieżek do celu dla każdej komórki, dlatego nie zawiera ona minimów lokalnych.

Wyniki eksperymentów pokazały, że gdy $\alpha = 0$ to najlepsza jest najkrótsza ścieżka do celu. Często jednak przebiega ona wzdłuż krawędzi przeszkód. W przypadku, gdy $\alpha = 0.5$ wówczas otrzymana ścieżka do celu jest bardziej bezpieczna. Gdy natomiast $\alpha = 1$ wtedy rozwiązaniem jest droga najbardziej bezpieczna.

Metoda sztucznych pól potencjalnych

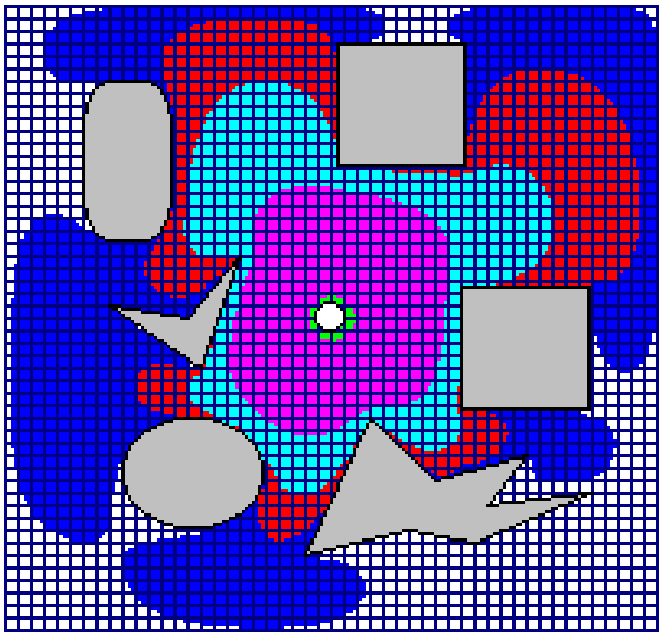
Metoda sztucznych pól potencjalnych została po raz pierwszy przedstawiona przez Khatiba.

Niech robot będzie robotem mobilnym o tylko dwóch stopniach swobody (tylko translacje), w przestrzeni konfiguracji, która leży poza przeszkodami powiększonymi o rozmiar robota. Wówczas robota można traktować jako punkt. W punkcie startowym ustawiamy pewien dodatni, odizolowany ładunek elektryczny. Następnie powiększone przeszkody ładujemy dodatnio, a pozycję docelową ujemnie. Wówczas taki odizolowany ładunek będzie w naturalny sposób przyciągany do celu i jednocześnie odpychany od przeszkód.

Największym problemem w metodzie pól potencjalnych jest uniknięcie sytuacji, w której robot znajdzie się w lokalnym minimum. Pomimo tego typu problemów w ostatnich latach powstało wiele prac opartych na metodzie pól potencjału.

Jedną z takich metod jest podejście rastrowe. Otoczenie robota zostaje naniesione na siatkę o kwadratowych komórkach. Komórkom przeszkód nadaje się wysoki potencjał natomiast celowi bardzo niski potencjał. Wówczas w całym obszarze, a dokładnie w jego wolnej przestrzeni, dla każdego skrzyżowania linii siatki wyliczany jest jego

potencjał. Wartość pola w punktach niebędących skrzyżowaniami siatki obliczana jest poprzez interpolację wartości czterech najbliższych skrzyżowań.



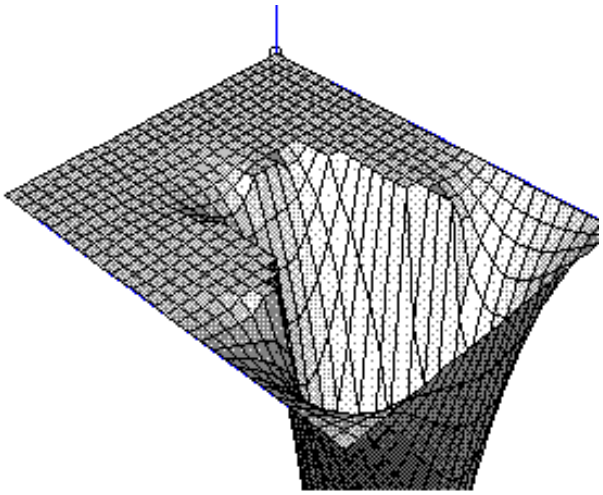
Rys.4. Przykładowy widok pola potencjalnego, z celem w środku (biały okrąg) i przeszkodami o różnych kształtach.
Drobna kratka przedstawia siatkę, na której dokonywane są obliczenia.

Wyliczanie wartości pola potencjalnego

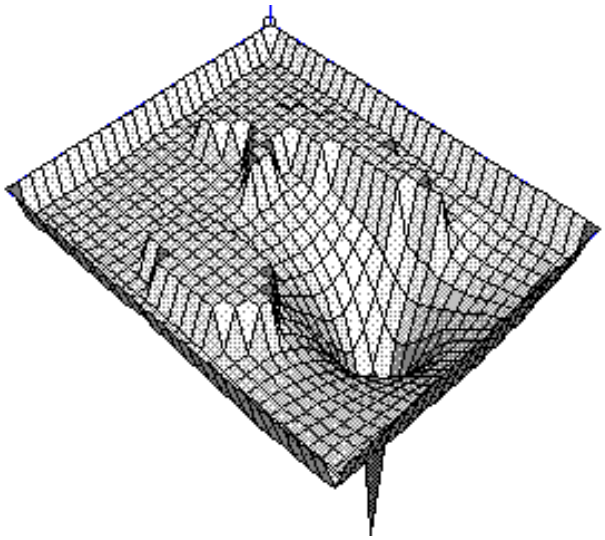
Niech Ω będzie zamkniętym obszarem o ciągłej, równej spójności. Pole potencjalne, $\Phi(x,y)$, w tym obszarze jest opisane równaniem Laplace'a:

$\nabla^2\Phi = 0$, które w 2-wymiarowym przypadku przybiera formę: $\frac{\partial^2\Phi}{\partial x^2} + \frac{\partial^2\Phi}{\partial y^2} = 0$ (3)

Zewnętrzna granica obszaru nie musi być połączona, ale musi zawierać powierzchnie wszystkich przeszkód oraz punkt docelowy. Powierzchniom przeszkód zostaje przyporządkowana pewna stała, wysoka wartość potencjału, a punktowi celu inna, niska wartość.
Powyższe równanie jest eliptycznym, cząstkowym równaniem różniczkowym i znajduje ono jednoznaczną, "statyczną" funkcję $\Phi(x,y)$, która spełnia to równanie w podanym obszarze. Jest to typowy problem wartości brzegowych. Istnieje kilka metod numerycznych służących do obliczania tego równania (a właściwie jego ogólniejszej postaci zwanej równaniem Poissona).



Rys.7. Mapa rastrowa rozkładu pola potencjału uzyskana dla konfiguracji otoczenia z rys.5.



Rys.8. Zmodyfikowany rozkład pola potencjału dla konfiguracji z rys.5.

Connolly pokazał, że w $\Phi(x,y)$ nie występują, żadne lokalne minima. Jednakże pomimo braku lokalnych minimów, w $\Phi(x,y)$ mogą istnieć obszary, w których moduł gradientu $\left|\nabla\Phi(x,y)\right|$, jest bardzo mały, podczas gdy jego cały przedział może być bardzo duży. Jest to spowodowane ekspotencjalnymi zmianami w wartościach pola w każdym punkcie.
Przeszkody są modelowane jako krzywe o pewnych wartościach pola, natomiast cel jest pojedynczym punktem o niskim potencjale. W rezultacie, zmiany pola od punktowego celu w kierunku przeszkód są początkowo bardzo szybkie, a dalej od celu zmiany są tylko nieznaczne.
Z tego powodu w sterowaniu robotem nie używa się wartości gradientu, lecz jedynie kierunek wektora gradientu. Dodatkowo, aby móc rozróżnić wartości pola w dwóch różnych, ale bliskich sobie punktach, wymagana jest znaczna dokładność obliczeń. Najważniejsze metody numeryczne obliczania równania Laplace'a to:

- Metoda Jacobi'ego
- Metoda Gauss-Seidel'a
- Metoda SOR
- Metoda elementów skończonych

Metoda Jacobi'ego

Aby obliczyć wartości pola w obszarze Ω, na model otoczenia zostaje nałożona siatka o określonych rozmiarach, a na niej punkt docelowy oraz granice obszaru, do których należą powierzchnie przeszkód. Funkcja $\Phi(x,y)$ jest, więc określona

wartościami w kolejnych, dyskretnych punktach. Jeśli wielkości oczek siatki są równe w całym obszarze, wówczas stosując metodę różnicową, równanie Laplace'a dla przypadku dwuwymiarowego może być przedstawione jako następująca dyskretna (iteracyjna) zależność:

$$\Phi_{i,j}^{n+1} = \frac{1}{4}(\Phi_{i+1,j}^{n} + \Phi_{i-1,j}^{n} + \Phi_{i,j+1}^{n} + \Phi_{i,j-1}^{n})$$

gdzie i=1,..,L jest pozycją komórki siatki w kierunku osi x,
j=1,..,J jest pozycją komórki siatki w kierunku osi y,
n przedstawia z którego kroku iteracyjnego pochodzi dana wartość pola.

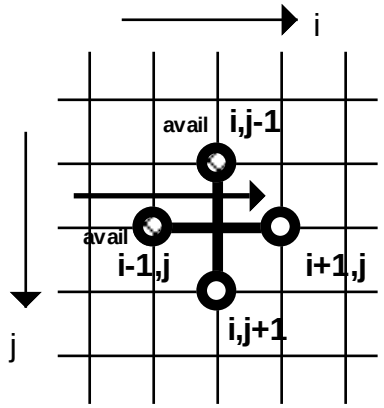
Stąd też, w komórce [i, j] działanie algorytmu polega na użyciu wartości średniej z wartości Φ otrzymanych w poprzedniej iteracji dla czterech sąsiadujących komórek. W każdej iteracji obliczana jest wartość dla każdej komórki siatki po kolei. Procedura ta jest powtarzana aż do zbieżności (ustalenia się pola).

Powyższa metoda jest klasyczną metodą zwaną metodą Jacobi'ego. Nie jest ona praktycznie stosowana jej zbieżność jest zbyt powolna. Jednakże jest ona podstawą do zrozumienia nowoczesnych technik, które zawsze są z nią porównywane.

Na siatce o rozmiarach *J*x*J* z warunkami brzegowymi Dirichleta na wszystkich czterech krawędziach obszaru, liczba iteracji potrzebna do zmniejszenia błędu o rząd 10^{-p}, przy dużym ***J*** wynosi: $r \approx \frac{1}{2} pJ^2$

Metoda Gauss-Seidel'a

Metoda Gaussa-Seidel’a jest kolejną klasyczną metodą numeryczną obliczania równania Poissona. W metodzie tej wartości Φ uaktualnione w tej samej iteracji są użyte do obliczenia wartości pola dla kolejnej komórki, gdy tylko jest to możliwe, tzn. nowe wartości są już dostępne. Ściślej, w dwóch spośród czterech sąsiadujących komórek uaktualnione wartości są dostępne. Pozostałe dwie komórki nadal zawierają wartości potencjału pola wyliczone w poprzedniej iteracji.



Innymi słowy, wyliczania wartości średniej jest dokonywane "na miejscu" a nie "kopiowane" z wcześniejszego kroku iteracyjnego do następnego. Jeśli przesuwamy się wzdłuż rzędów siatki, zwiększając i przy ustalonym j, wówczas mamy:

$$\Phi_{i,j}^{n+1} = \frac{1}{4}(\Phi_{i+1,j}^{n} + \Phi_{i-1,j}^{n+1} + \Phi_{i,j+1}^{n} + \Phi_{i,j-1}^{n+1})$$

Na siatce o rozmiarach *J*x*J* z warunkami brzegowymi Dirichleta na wszystkich czterech krawędziach obszaru, liczba iteracji potrzebna do zmniejszenia błędu o rząd 10^{-p}, przy dużym *J* wynosi:

$$r \approx \frac{1}{4} pJ^2$$

Polepszenie tej wartości w stosunku do metody Jacobi'ego jest więc w przybliżeniu rzędu dwóch.

Zmieniona metoda Gauss-Seidel'a

Zasada działania poniższej metody jest całkowicie oparta na metodzie Gaussa-Seidel'a. Różnica polega na tym, że obliczenia są dokonywane do przodu i do tyłu na przemian. Innymi słowy, najpierw startując w punkcie i=1, j=1 odpowiednio zwiększane są indeksy, a następnie startuje się od punktu i=L; j=J zmniejszając odpowiednio indeksy. Procedura taka jest powtarzana az do momentu ustalenia się wartości pola potencjalnego. Ten opis słowny można przedstawić następującymi równaniami, wyrażającymi pojedynczą iterację::

1: $\Phi_{i,j}^{n+1} = \frac{1}{4}(\Phi_{i+1,j}^{n} + \Phi_{i-1,j}^{n+1} + \Phi_{i,j+1}^{n} + \Phi_{i,j-1}^{n+1})$ przy zwiększaniu *i* oraz *j*

2: $\Phi_{i,j}^{n+1} = \frac{1}{4}(\Phi_{i+1,j}^{n+1} + \Phi_{i-1,j}^{n} + \Phi_{i,j+1}^{n+1} + \Phi_{i,j-1}^{n})$ przy zmniejszaniu *i* oraz *j*

Metoda SOR (Simultaneous Over-Relaxation)

Jedną z praktycznych metod rozwiązywania eliptycznych równań różniczkowych cząstkowych jest metoda zwana Simultaneous Over-Relaxation (SOR). Algorytm tej metody dokonuje "przesadnej" korekty (mocniejszej niż wymagana) wartości $\Phi^{(r)}$ w r-tym etapie iteracji Gaussa-Seidel'a, przez co jakby przewidując przyszłe korekty. Jest to osiągnięte przez zastosowanie specjalnego parametru ω zwanego parametrem ponad-relaksacji (over relaxation parameter). Gdy ω jest pomiędzy 1 i 2 metoda SOR wykazuje znacznie szybszą zbieżność niż metoda Gaussa-Seidel'a. Obliczenia SOR są dokonywane w następujący sposób:

Dla równania Laplace'a, które jest równaniem eliptycznym drugiego rzędu, reprezentacja różnic skończonych przybiera formę:

$$\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j} = 0$$

z procedurą iteracyjną opisaną:

$$\Phi^{*}_{i,j} = \frac{1}{4}(\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1})$$

Wówczas $\Phi^{new}_{i,j}$ jest następującą średnią ważoną:

$$\Phi^{new}_{i,j} = \omega\Phi^{*}_{i,j} + (1-\omega)\Phi^{old}_{i,j}$$

Aby obliczenia były łatwiejsze, wprowadza się następującą zmienną (wyrażającą resztę):

$$\xi_{i,j} = \Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j}$$

co daje łatwiejszą zależność:

$$\Phi^{new}_{i,j} = \Phi^{old}_{i,j} + \omega\frac{\xi_{i,j}}{4}$$

Na siatce o rozmiarach *J*x*J* z warunkami brzegowymi Dirichleta na wszystkich czterech krawędziach obszaru, liczba iteracji potrzebna do zmniejszenia błędu o rząd 10^{-p}, przy dużym ***J*** wynosi:

$$r \approx \frac{1}{3}pJ$$

Porównując to z poprzednimi równaniami, łatwo zauważyć, że optymalna SOR wymaga liczby iteracji rzędu J, w przeciwieństwie do wcześniejszego rzędu J^2. Równanie to wskazuje, że dokładność trzech cyfr po przecinku (p=3) wymaga liczby iteracji równej liczbie komórek wzdłuż jednego boku siatki, tzn. ***J***. Aby otrzymać sześciocyfrową dokładność należy zastosować około dwa razy więcej iteracji.

Metoda Elementów Skończonych (Finite Element Method)

Metoda elementu skończonego różni się znacznie od opisanych wcześniej technik. Pokazane zostało, że rozwiązanie równania Laplace'a jest funkcją, która minimalizuje następujący funkcjonał:

$$J(w) = \frac{1}{2}\int_{\Omega}\left[\left(\frac{\partial w}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial y}\right)^2\right]d\Omega$$

gdzie *w* oznacza pewną dozwoloną funkcję, tzn. ciągłą, z kawałkami ciągłymi pochodnymi pierwszego rzędu i spełniającą warunki Dirichleta. Obszar, w którym wartości pola potencjalnego mają być znalezione jest dzielony na *m* trójkątów (elementów skończonych). Będą one dalej oznaczane e_i (i=1,2,..,m).

Wtedy, $J(w) = \sum_{i=1}^{m} J(w^i)$,

$$J(w^i) = \frac{1}{2}\int_{e_i}\left[\left(\frac{\partial w^i}{\partial x}\right)^2 + \left(\frac{\partial w^i}{\partial y}\right)^2\right]de_i$$

Dla elementu e_i o wierzchołkach i, j, k (w kolejności przeciwnej do wskazówek zegara) funkcja w ma postać:

$$w^i(x,y) = a_{i1} + a_{i2}x + a_{i3}y, \qquad x,y \in e_i$$

Aby znaleźć stałe występujące w tym równaniu używamy wartości funkcji "w" w punktach i, j, k:

$w_i = a_{i1} + a_{i2}x_i + a_{i3}y_i \qquad w_j = a_{i1} + a_{i2}x_j + a_{i3}y_j, \qquad w_k = a_{i1} + a_{i2}x_k + a_{i3}y_k$

Stąd: $$S = \begin{vmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{vmatrix} \neq 0$$

wtedy $a_{i1} = \frac{1}{S}\left(\alpha_i w_i + \alpha_j w_j + \alpha_k w_k\right)$ $a_{i2} = \frac{1}{S}\left(\beta_i w_i + \beta_j w_j + \beta_k w_k\right)$

oraz $a_{i3} = \frac{1}{S}\left(\gamma_i w_i + \gamma_j w_j + \gamma_k w_k\right)$

gdzie $\alpha_i = x_j y_k - x_k y_j, \quad \beta_i = y_j - y_k, \quad \gamma_i = x_k - x_j$ etc.

Podstawiając, mamy:

$$w^i(x,y) = \frac{1}{S}\left[\left(\alpha_i + \beta_i x + \gamma_i y\right)w_i + \left(\alpha_j + \beta_j x + \gamma_j y\right)w_j + \left(\alpha_k + \beta_k x + \gamma_k y\right)w_k\right]$$

Różniczkując i podstawiając otrzymujemy

$$J(w^i) = \frac{1}{2S}\left[\left(\beta_i w_i + \beta_j w_j + \beta_k w_k\right)^2 + \left(\gamma_i w_i + \gamma_j w_j + \gamma_k w_k\right)^2\right]$$

oraz $$\sum_{i=1}^{m} \frac{\partial J(w^i)}{\partial w_p} = 0 \qquad \text{p=1,2,...,m.}$$

Ten układ równań może już być rozwiązany jakąkolwiek metodą rozwiązywania liniowych układów równań, np. metodą Gaussa-Jordana (the Gauss-Jordan method with complete pivoting).

LITERATURA

[1] Jarvis, J.C. Byrne: *"Robot Navigation: Touching, seeing and knowing"* Proc. Australian Conf.on Artificial Intelligence, Melbourne, Australia 1986.

[2] Barraquand, J.C. Latombe: *"Robot motion planning: A distributed representation approach"* The International Journal of Robotics Research, vol. 10(6), pp. 628-649, 1991.

[3] D.W. Payton: *"Internalised plans"* Robot. Autonomous Systems, vol. 6(1-2), pp. 89-103, 1990.

[4] J. Ilari, C. Torras: *"2D path planning: A configuration space heuristic approach"* The International Journal of Robotics Research, vol. 9(1), pp.75-91, 1990.

[5] Alexander Zelinsky: *"Navigation with learning"* Proc. IEEE/RSJ Conference on Intelligent Robot Systems, Tsukuba, pp. 331-338, Japan 1989.

[6] Alexander Zelinsky: *"Environment exploration and path planning algorithms for a mobile robot using sonar"* Ph.D. dissertation, Department of Computer Science, Wollongong University, Australia 1991.

[7] Alexander Zelinsky: *"Using Path Transforms to Guide the Search for Findpath in 2D"* The International Journal of Robotics Research,vol. 13, no. 4, pp.315-325, Massachusetts Institute of Technology, August 1994.

[8] Oussama Khatib: *"Real-Time Obstacle Avoidance for Manipulators and Mobile Robots"*, The International Journal of Robotics Research, Volume 5, No. 1, pp. 90-98, the MIT Press, Cambridge Massachusetts 1986.

[9] Rob Buckingham, Andrew Graham: *"Real-Time Collision Avoidance Of Manipulators With Multiple Redundancy"* Mechatronics, vol. 3, no. 1, pp. 89-106, 1993.

[10] Rob Buckingham: *"Robotics in Surgery"*, IEE Review, September 1994.

[11] C. I. Connolly, J. B. Burns, R. Weiss: *"Path Planning Using Laplace's Equation"*, Proc. IEEE Int. Conf. Robotics Automation, pp. 2102-2106, 1990.