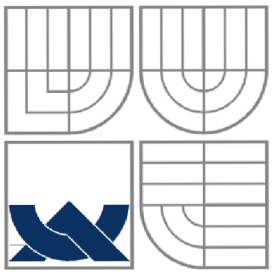


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

APLIKACE VORONÉHO DIAGRAMŮ V PLÁNOVÁNÍ DRÁHY ROBOTU

APPLICATION OF VORONOI DIAGRAMS IN ROBOT MOTION PLANNING

BAKALÁŘSKÁ PRÁCE
BACHELOR THESIS

AUTOR PRÁCE **Luděk Chaloupka**
AUTHOR

VEDOUCÍ PRÁCE **doc. RNDr. Ing. Miloš Šeda, Ph.D.**
SUPERVISOR

BRNO 2009

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2008/09

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Chaloupka Luděk

který/která studuje v **bakalářském studijním programu**

obor: **Aplikovaná informatika a řízení (3902R001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Aplikace Voroného diagramů v plánování dráhy robotu

v anglickém jazyce:

Application of Voronoi Diagrams in Robot Motion Planning

Stručná charakteristika problematiky úkolu:

Problematika plánování trasy robotu (robot motion planning) ve scéně s překážkami patří mezi netriviální optimalizační úlohy. Úloha může mít řadu modifikací daných tvarem uvažovaných překážek (bodové, pravoúhlé, polygonální, prostorové), omezeními na pohyb robotu (8-směrový, pravoúhlý, obecný). Komplikací může být pouze částečná znalost scény, v níž se robot pohybuje, a přítomnost pohybujících se překážek.

Cíle bakalářské práce:

Práce se zaměřuje na využití metod počítačové geometrie a jejich datových struktur pro danou třídu úloh. Pro dílčí problémy se předpokládá i aplikace stochastických heuristických metod a metod přibližného usuzování. Cílem je implementovat navržené algoritmy a připravit jejich budoucí uplatnění v reálném robotu vyvíjeném na Ústavu mechaniky těles, mechatroniky a biomechaniky.

Seznam odborné literatury:

[1] de Berg, M., van Kreveld, M., Overmars, M. and Schwarzkopf, O: Computational Geometry: Algorithms and Applications. Springer-Verlag, Berlin, 2000 (2nd rev. ed.).
[2] Fortune, S.: Voronoi Diagrams and Delaunay Triangulations. In Du, D.A. and Hwang, F.K. (eds.): Euclidean Geometry and Computers. World Scientific Publishing, Singapore, 1992, pp. 193-233.
[3] Okabe, A., Boots, B., Sugihara, K. and Chiu, S.N.: Spatial Tessellations and Applications of Voronoi Diagrams. John Wiley & Sons, New York, 2000.
[4] Zilouchian, A. and Jamshidi, M.: Intelligent Control Systems Using Soft Computing Methodologies. CRC Press, Boca Raton, 2001.

Vedoucí bakalářské práce: doc. RNDr. Ing. Miloš Šeda, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2008/09.

V Brně, dne 26. 03. 2009

L.S.

doc. RNDr. Ing. Miloš Šeda, Ph.D.
Ředitel ústavu

doc. RNDr. Miroslav Doupovec, CSc.
Děkan fakulty

LICENČNÍ SMLOUVA

(na místo tohoto listu vložte vyplněný a podepsaný list formuláře licenčního ujednání)

ABSTRAKT

Tato bakalářská práce je zaměřena na tématiku plánování dráhy robotu. Stěžejním problémem je pohyb bezrozměrného (tj. bodového) robotu, bez omezení pohybu a ve scéně s nepohyblivými překážkami (též bodovými). Součástí práce je implementace zmíněných algoritmů pro danou třídu úloh a vymezení vhodné metody.

ABSTRACT

This bachelor thesis is focused on the theme of robot motion planning. The motion of dimensionless (i.e. point) robot without any restriction of motion and on the stage with static obstacles (also point) forms the fundamental issue of the thesis. The thesis also contains the implementation of presented algorithm of given level of exercises and definition of appropriate method.

KLÍČOVÁ SLOVA

Voroného diagram, inkrementální algoritmus, zametací algoritmus, algoritmus rozděl a panuj, Dijkstrův algoritmus, A* algoritmus

KEYWORDS

Voronoi diagram, incremental algorithm, plane sweep algorithm, divide and conquer algorithm, Dijkstra algorithm, A* algorithm

PODĚKOVÁNÍ

Děkuji vedoucímu mé bakalářské práce, doc. RNDr. Ing. Miloši Šedovi, Ph.D. za vstřícnou ochotu a poskytnuté materiály pro zpracování této práce. Děkuji. Další dík patří mým nejbližším, především rodičům a prarodičům za poskytnutou podporu, jak psychickou, tak zejména materiální a dále své přítelkyni za podporu při mých studiích. Děkuji.

OBSAH

1 ÚVOD

Aplikace Voroného diagramů v počítačové grafice a počítačové simulaci se začala využívat již v sedmdesátých letech 20. století v robotice a to zejména v úlohách plánování dráhy robotu mezi dvěma body.

Problematika plánování trasy robotu v prostředí s překážkami patří mezi netriviální úlohy. Úlohy mohou mít řadu modifikací a pro úspěšné naplánování dráhy by měl mít robot určité znalosti o prostředí, ve kterém se pohybuje. Tyto znalosti jsou buď předem známy, ať úplně či částečně nebo nemusí být známy vůbec, v tomto případě by se robot musel spolehnout na různé senzory, kterými by byl schopen překážky rozeznat a úspěšně se jim vyhnout.

Ve této bakalářské práci se budeme zaobírat plánováním dráhy bodového robotu ve scéně s nepohyblivými překážkami (ty budou též bodové) a se všemi informacemi o prostředí, ve kterém se bude pohybovat. Robot nebude ve svém pohybu nijak omezen, to znamená, že se může pohybovat ve všech směrech. Jedinou podmínkou je to, aby při pohybu ze startovní do cílové pozice nikde nekolidoval s žádnou překážkou. Protože chceme, aby se robot vyhnul co možno nejlépe překážkám mezi dvěma určenými body, musí jet co nejdále od nich. Toho docílíme využitím hran předem vygenerovaného Voroného diagramu, po kterých povede trajektorie bodového robotu.

Zmíníme se zde také o různých metodách pro určení nejkratší cesty, či o metodách přibližného usuzování a dalších způsobech a algoritmech pro určení dráhy robota. Dále nastíníme problematiku pro různé modifikace tvarů překážek.

2 HISTORIE

První představa o rozdělení plochy pomocí Voroného diagramů se objevila již roku 1644 v Descartově práci „Le Monde de Mr. Descartes ou Le Traité de la Lumière“ kde pomocí těchto diagramů poukazoval na uspořádaní hmoty ve sluneční soustavě a jejím okolí. Další, kdo se začal touto myšlenkou zabývat byl německý matematik Dirichlet (celým jménem Johann Peter Gustav Lejeune Dirichlet), který využíval ve svých studií kvadratických norem 2D a 3D Voroného diagramy již v roce 1850, proto se také můžeme setkat s názvem *Dirichletovy mozaiky*.

V roce 1908 nadefinoval a nadále se věnoval zobecněným n-rozměrným případům diagramů ruský matematik *Georgy Fedoseevich Voronoi (Voronoy)*, zobrazen na obr. 1, podle kterého jméno *Voronoi diagram*.

Obr. 1 Georgy Fedoseevich Voronoi

Tyto diagramy byly postupem doby stále více využívány v dalších a dalších oborech. Například americký meteorolog Alfred H. Thiessen je využíval k analýze prostorově rozložených dat (měření srážek, vlhkosti, atd.), podle něj se můžeme setkat s názvem *Thiessenovy polygony*, ty mají využití hlavně v meteorologii a geografii. Další známe diagramy jsou například *Wigner-Seitz jednotkové buňky*, se kterými se můžeme setkat ve fyzice materiálů. Kromě výše uvedených oborů našly ve 20. století své uplatnění i v mnoha jiných, jako je biologie, medicína, chemie, krystalografii, počítačové grafice, samozřejmě i v robotice a ve spoustě dalších oborech, které se stále utvářejí a vyvíjejí.

3 VORONÉHO DIAGRAMY

Geometrické struktury jsou výrazovým prostředkem mnoha významných aplikací jako je robotika, grafika, vizualizace informací, problémy optimálního umístění atd. Jejich studiem, vlastnostmi a algoritmy pro jejich konstrukci se zabývá počítačová geometrie. Mezi nejvýznamnější představitele patří Voroného diagramy a Delaunayho triangulace. I když tyto struktury jsou definovány i pro obecný n-dimenzionální prostor, v práci se omezíme na rovinu.

3.1 Definice a rozdělení

Voroného diagram (obr. 2) představuje základ všech ostatních zobecněných typů diagramů, na kterých lze uplatnit některé z vlastností Voronoiova diagramu.

Voroného diagram V(P) pro danou množinu bodů $P=\{p_1,p_2, \dots ,p_n\}$ v rovině je její rozklad na uzavřené či otevřené oblasti (buňky, regiony) $V(p_1)$, $V(p_2)$, … , $V(p_n)$ takový, že $p_i \in V(p_i)$ pro každé p_i, a každý bod $x \in V(p_i)$ je blíže bodu p_i než ke kterémukoliv jinému bodu p_j, což jsou body na hranicích oblastí $V(p_i)$ a $V(p_j)$, ty jsou od bodů p_i a p_j stejně vzdáleny.

Vzdálenost $d(p_i, p_j)$ mezi body $p_i=(x_i, y_i)$ a $p_j=(x_j, y_j)$ ve Voroného diagramu je definována:

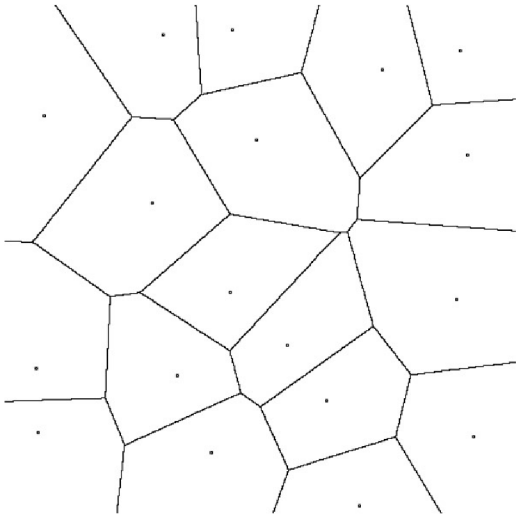
euklidovskou metrikou

$$d(p_i, p_j)=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$$

nebo *rektilineární metrikou* (též *Manhattanská metrika*)

$$d(p_i, p_j)=|x_i-x_j|+|y_i-y_j|$$

Jak vidíme dělíme Voroného diagramy dle zvolené metriky na dvě základní třídy a to na *euklidovské* a *rektilineární* diagramy. Rozdíl mezi těmito dvěma třídami si popíšeme později.



Obr. 2 Ukázka Voroného diagramu

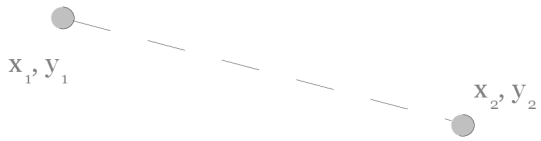
3.2 Vlastnosti Voroného diagramů [5]

Předpokládejme, že *Voroného diagramy nejsou degenerované* (tj. čtyři nebo více jeho hran se neprotíná v jednom bodě). Pak platí:

- *Každý vrchol Voroného diagramu V(P) je společným průsečíkem právě třech hran diagramu.*
- *Bod q je vrcholem diagramu V(P) právě tehdy, když jeho největší prázdná kružnice $C_P(q)$ obsahuje tři body na své hranici.*
- *Osa úsečky spojující body p_i a p_j definuje hranu diagramu V(P) právě tehdy, když existuje bod q takový, že $C_P(q)$ obsahuje jak p_i tak i p_j na své hranici, ale žádný jiný bod.*
- *Pro každé q v P je V(P) konvexní.*
- *Voronoiův diagram V(P) pro množinu bodů P je planárním grafem.*
- *Oblast $V(p_i)$ je neohraničená právě tehdy, když p_i je bodem na hranici konvexního obalu množiny P.*
- *Počet vrcholů Voronoiova diagramu n-prvkové množiny bodů v rovině je nejvýše roven 2n-5 a počet jeho hran je nejvýše roven 3n-6.*

3.3 Voroného diagram s euklidovskou metrikou

Hrany Voroného diagramů s *euklidovskou* metrikou svírají jakýkoli úhel se svislou nebo s vodorovnou osou, to znamená, že pro sestavení diagramu je využito jakéhokoli směru na rozdíl od metriky *rektilineární*. Tím je dáno, že spojnice mezi dvěma body je přímka procházející těmito body (obr. 3).



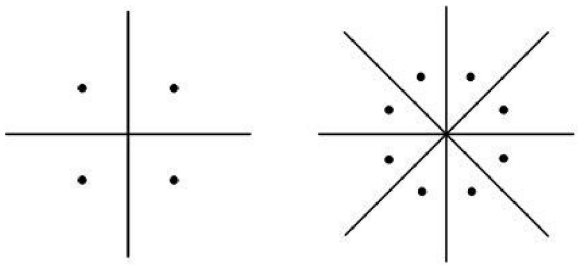
Obr. 3 Spojnice dvou bodů v Euklidovské metrice

Vzdálenost mezi danými body $p_1=(x_1, y_1)$ a $p_2=(x_2, y_2)$ je pak v euklidovské metrice dána tímto již výše uvedeným vztahem:

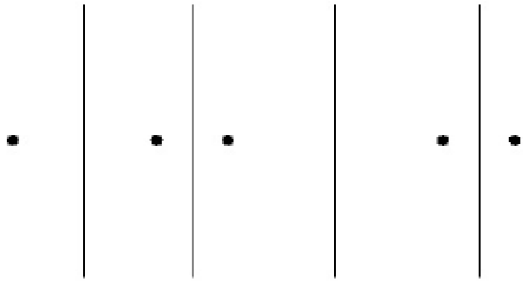
$$d(p_1, p_2)=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

Hranice mezi oblastí $V(p_i)$ a $V(p_j)$ je tvořena množinou bodů, jejichž euklidovská vzdálenost k bodu p_i je stejná jako k bodu p_j. Dále platí, že hranice je kolmá na úsečku spojující tyto sousední body (p_i, p_j) a protíná ji přesně v polovině její délky. Bod, jehož euklidovská vzdálenost je stejná minimálně ke třem bodům, se nazývá *vrcholem Voroného diagramu*. Pokud se alespoň v jednom vrcholu protínají více než tři hrany, jedná se o tzv. *degenerovaný Voroného diagram* (obr. 4). [3]

Dalším speciálním případem je diagram pro množinu bodů ležících na přímce, takovýto diagram má rovnoběžné hrany, z čehož vyplývá, že oblasti Voronoiova diagramu jsou otevřené, neohraničené a neexistuje zde žádný vrchol, jak můžeme vidět na obr. 5.



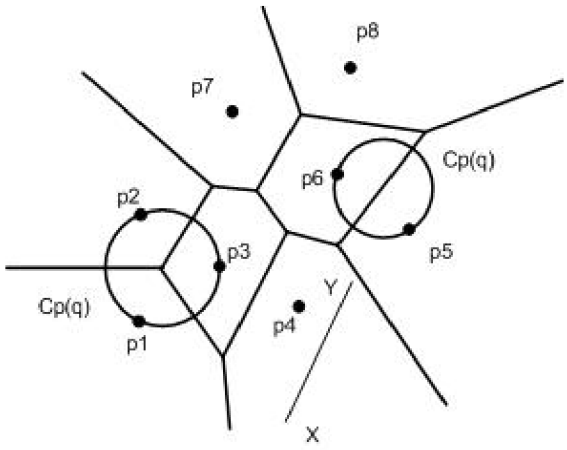
Obr. 4 Degenerované Voroného diagramy



Obr. 5 Voroného diagram s body ležícími na přímce

3.3.1 Vlastnosti Voroného diagramu s euklidovskou metrikou [3]

- *Voronoiův diagram $V(p_i)$ je neohraničený, právě když je bod p_i na hranici konvexního obalu množiny P.*
- *Pro jakýkoli bod $p_i \in P$ je $V(p_i)$ konvexní. To znamená, že všechny body spojnice jakýchkoli dvou bodů Voronoiova polygonu $V(p_i)$ leží ve $V(p_i)$.*
- *Hrany Voronoiova diagramu reprezentují rovinný graf.*
- *Obvykle platí, že každý vrchol n_i nedegenerovaného Voronoiova diagramu V(P) je průsečíkem právě tří stran. U degenerovaného diagramu se alespoň v jednom vrcholu protínají více než 3 jeho hrany.*
- *Bod q je vrcholem nedegenerovaného Voronoiova diagramu V(P), právě když jeho největší prázdná kružnice $C_p(q)$ sdružuje na svém okraji právě 3 body množiny P.*
- *Přímka ležící mezi body p_i a p_j tvoří hranici mezi dvěma Voronoiovými polygony, právě když existuje takový bod q, pro který platí, že největší prázdná kružnice $C_p(q)$ se středem v bodě q obsahuje na svém okraji právě body p_i a p_j kde $p_i \neq p_j$* (obr. 6).



Obr. 6 Ukázka vlastnosti Voroného diagramu s euklidovskou metrikou

- Počet vrcholů Voronoiova diagramu V(P) obsahující n bodů, je nejvíce 2n – 5 a počet hran je nejvíce 3n – 6.

3.4 Voroného diagram s rektilineární metrikou

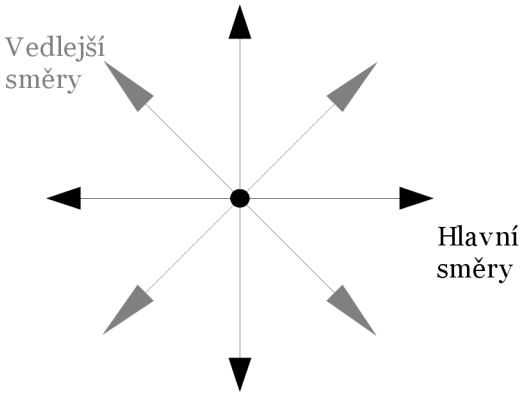
Vzdálenost dvou bodů je nyní určena pomocí *rektilineární metriky*, která se na rozdíl od *euklidovské* určuje jako součet absolutních hodnot rozdílu x-ových a y-ových souřadnic (obr. 7).

Obr. 7 Spojnice dvou bodů v rektilineární metrice

Vzdálenost mezi danými body $p_1=(x_1, y_1)$ a $p_2=(x_2, y_2)$ je pak v rektilineární metrice dána tímto vztahem:

$$d(p_1, p_2)=|x_1-x_2|+|y_1-y_2|$$

Dle obrázku 7 lze usoudit, že Voroného diagram daný rektilineární metrikou bude obsahovat pouze horizontální, vertikální a diagonální hrany. Z toho nám vyplývá, že zde bude omezen pohyb v osmi směrech, tzn. směr vodorovný a směr svislý, které nazýváme hlavními směry a směr diagonální, tj. směry vedlejší, ty jsou k hlavním směrům pootočeny o 45° nebo o 135°. Hlavní směry jsou si navzájem kolmé, taktéž i vedlejší, jak lze vidět na obrázku 8.



Obr. 8 Využitelné směry v rektilineární metrice

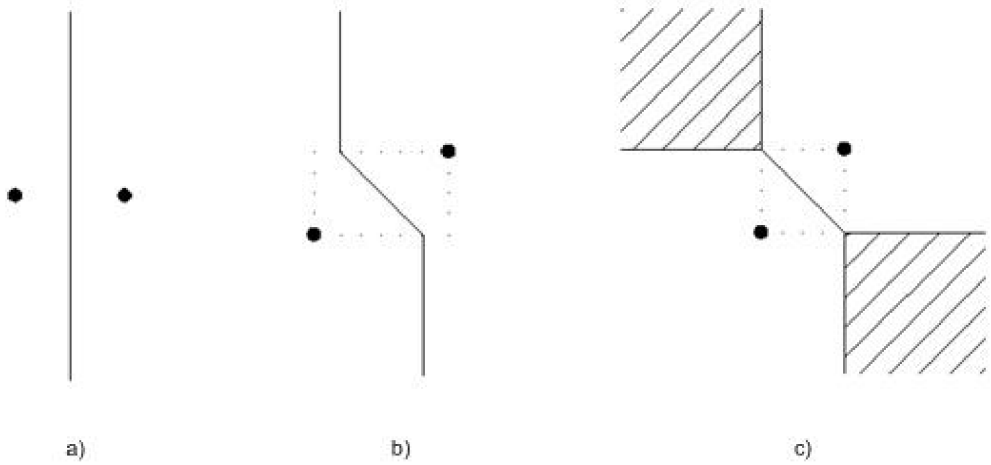
3.4.1 Možné vedení hrany mezi body [3]

Vzájemné vedení hrany mezi body může být hned několik, ale v principu je dělíme na tyto základní tři (obr. 9) :

- Body ležící na přímce. Tyto body jsou rovnoběžné buďto s osou x

nebo s osou y (Obr. 9 a).

- Body ležící na diagonále obdélníka (Obr. 9b).
- Body ležící na diagonále čtverce (Obr. 9c).



Obr. 9 Základní rozdělení vzájemné polohy mezi dvěma body

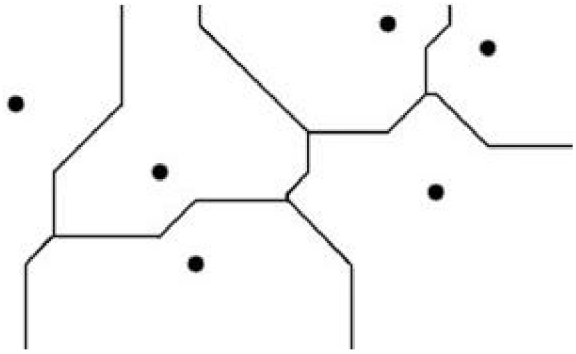
Body zobrazující obrázek 9a leží na jedné přímce, zde má pomyslný obdélník (případně čtverec) jednu stranu nulovou, proto je výsledná hrana kolmá přímka.

Další znázornění (Obr. 9b) je případ, kdy body tvoří obdélník a hrana diagramu je zde složena ze tří částí, dva jsou ve směru vertikálním (hlavním) a třetí ve směru diagonálním (vedlejším). Spojením vznikne výsledná hrana.

Poslední případ (Obr. 9c) je obdobný jako předchozí, avšak s tím rozdílem, že body tvoří čtverec. Body jsou rozděleny hranou ve směru diagonály a ostatní dvě hrany mohou ležet kdekoli ve šrafované části (tedy jak horizontálně, tak i vertikálně).

3.4.2 Vlastnosti Voroného diagramu s rektilineární metrikou [3]

- *Voronoiův diagram V(P) z P je rovinným grafem*
- *Oblast Voronoiova diagramu $V(p_i)$ nemusí být konvexní.*
- *Každý vrchol Voronoiova diagramu s rektilineární metrikou* (obr. 10) *je obyčejně průsečíkem dvou nebo tří hran.*
- *Pokud existuje bod q takový, že je vrcholem, který vznikne protnutím právě tří hran, pak jeho kružnice $C_p(q)$ obsahuje tři body (nejenom na hranici).*



Obr. 10 Voroného diagram s rektilineární metrikou

4 ALGORITMY A KONSTRUKCE

V této kapitole se budeme zabývat konstrukcí Voroného diagramu, kde využijeme nabytých znalostí a vlastností z předchozích kapitol. Našim cílem bude popsat algoritmus pro sestavení těchto diagramů. Nejprve si však upřesníme nějaké poznatky o algoritmech.

Algoritmus je daný postup, který podle zadaných (definovaných), nám předem, či v průběhu činnosti známých informací, vygeneruje výsledná data. Algoritmus má vždy alespoň jeden výsledek (výstup), v našem případě Voroného diagram, který reaguje na vložená vstupní data a řádně je zpracuje. Každý algoritmus musí být jednoznačně a přesně definován, nesmí řešit jen konkrétní problém pro konkrétní data, ale všeobecné úlohy pro různé vstupní údaje. Nepsaným požadavkem je efektivnost algoritmu.

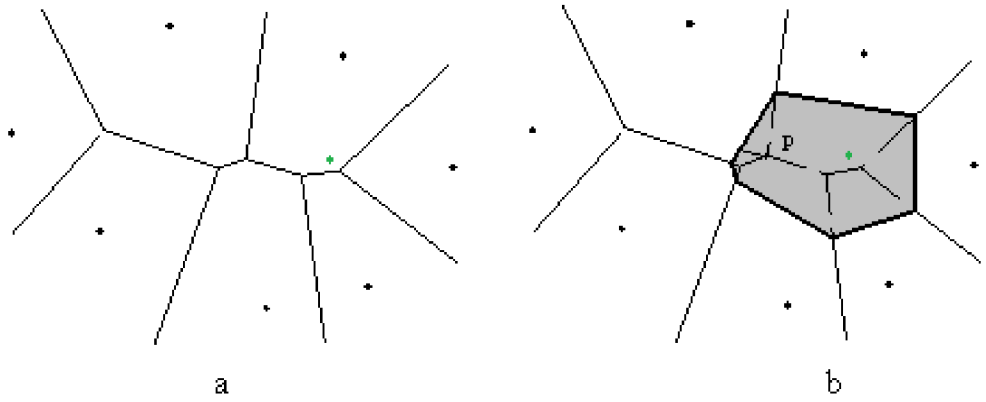
4.1 Konstrukční algoritmy

V této práci se nebudeme zabývat všemi dostupnými algoritmy pro konstrukci Voroného diagramů, ale ukážeme si pouze ty nejpoužívanější k danému tématu této bakalářské práce.

- *Inkrementální algoritmus (incremental algorithm)*
- *Algoritmus rozděl a panuj (divide and conquer)*
- *Zametací algoritmus (plane sweep algorithm)*

4.1.1 Inkrementální algoritmus

Tento algoritmus vytváří diagram postupným vkládáním bodů, proto je nazýván inkrementální (přírůstkový). Jak je již naznačeno výchozím stavem je diagram složený ze dvou výchozích bodů oddělených do dvou polorovin kolmicí, která prochází středem jejich spojnice. Jestliže k takto vytvořenému diagramu přidáme další bod p, pak algoritmus určí nejprve aktuální oblast (označme bod q definující tuto oblast) kde p leží. Poté se vytvoří hranice oblasti k bodu p a dále se tvoří hrana po hraně, tedy kolmice vedená středem úsečky pq utvoří hranu h v diagramu, koncové body hrany h jsou dány průsečíky úsečky pq a hranami oblasti s bodem p. Během tohoto procesu jsou části nejbližší k bodu p přerušeny a vymazány, jak je naznačeno na obrázku 11. Více o tomto algoritmu v [3] a [5].

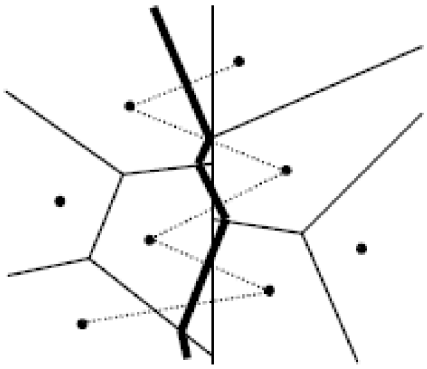


Obr. 11 Vytvoření nové oblasti inkrementální metodou

Čas potřebný pro výpočet vytvoření diagramu za pomocí této metody je v nejhorším případě $O(n^2)$, lze ho ovšem zrychlit na očekávané $O(n)$.

4.1.2 Algoritmus rozděl a panuj

Algoritmus rozděl a panuj (divide and conquer) je další ze známých metod pro rychlé vytvoření diagramu. Tato metoda vertikálně rozděluje danou množinu n bodů na dvě podmnožiny o přibližně stejné velikosti. Voroného diagramy pro tyto dvě podmnožiny se vypočítají rekurzivně a poté se propojí do výsledného diagramu, jak naznačuje obr. 12.



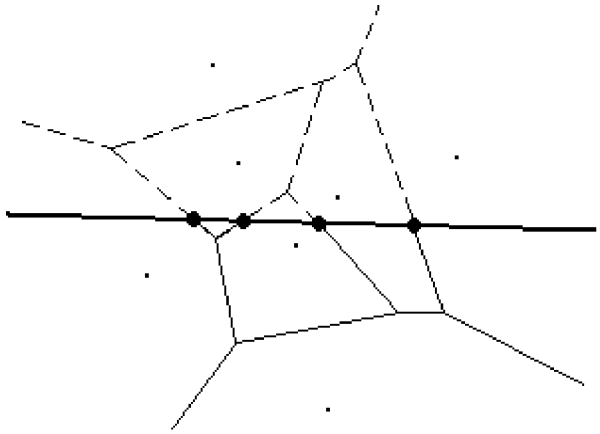
Obr. 12 Propojení dvou podmnožin ve výsledný diagram

K propojení dvou takto vzniklých diagramů ve výsledný se používá tzv. způsob „*uvažování mravence*“, kde tento mravenec začne procházet území na jedné straně (např. dole nebo nahoře) a postupně přitom prochází přes území na druhou stranu a vyhýbá se přitom objektům stojící v cestě. Aby minimalizoval

nebezpečí kolidovat s překážkou jde cestou, která je kolmá na spojnici dvou bodů (překážek) a vede jejím středem (obr. 12), tím se drží nejdále od obou bodů. Jakmile narazí na hranu diagramu, vydá se směrem k další hraně obdobně jako v předchozím případě. Aby byl krok dělení proveden vyváženým způsobem, bude potřeba pro celý algoritmus čas *O(n log n)*, kde spojení dvou podmnožin bude v čase *O(n)*. Tato kapitola převážně čerpána z [5].

4.1.3 Zametací algoritmus

Zametací algoritmus nebo též nazývaný podle Steve Fortune (1985) *Fortuneho algoritmus* je další z mnoha technik pro sestrojení Voroného diagramu. Princip tohoto algoritmu je založen na postupném posouvání *zametací přímky* přes všechny dané body množiny, přičemž algoritmus simuluje posouvání linie po rovině zdola nahoru a část pod touto linií je v jakémkoli časovém bodě kompletní viz obr. 13. Celý algoritmu se pak provede v čase *O(n log n)*. Více v [3] a [4].



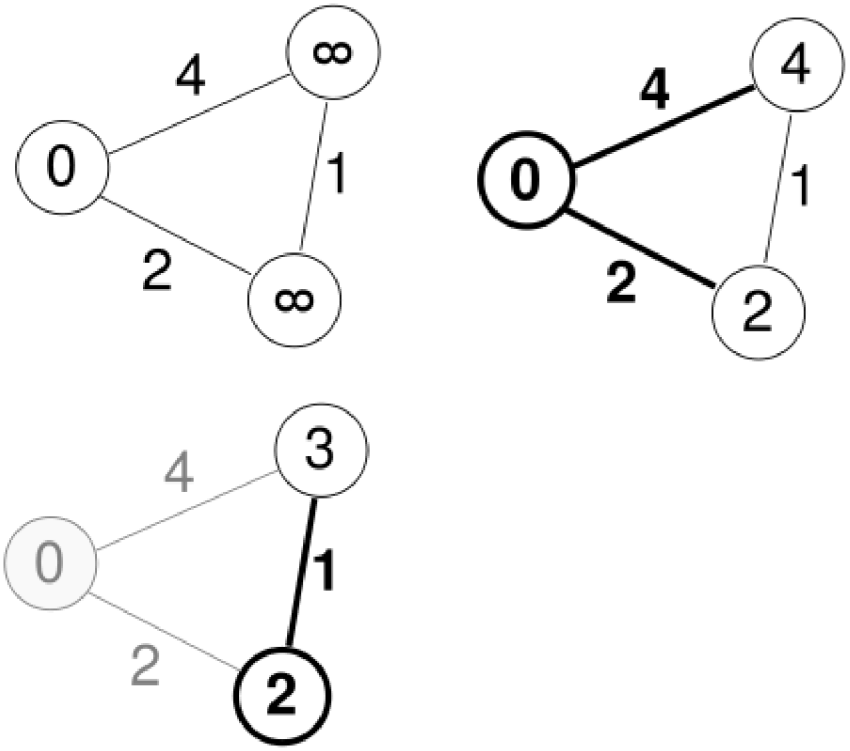
Obr. 13 Princip zametací přímky

5 NALEZENÍ NEJKRATŠÍ CESTY

Spojnice výchozího (startovního) a cílového bodu je považována za nejkratší možnou cestu mezi těmito body. Pokud však mezi těmito body leží nějaká překážka či ve většině případů hned několik, nelze již tak snadno určit nejkratší vzdálenost mezi těmito body pouhou spojnicí, aniž by během cesty nedošlo ke kolizi s překážkami ležících na cestě. Situace se však ještě může výrazně zkomplikovat, pokud tvary překážek nebudou pouhé body (jako v našem případě), ale nějaké pravoúhlé, polygonální či dokonce prostorové překážky. Dalším nezanedbatelným problémem je, pokud by se překážky pohybovaly. Při pohybu dvou a více polygonů bude zcela jistě docházet k vzájemným srážkám. Vzniklé kolize musíme umět rychle detekovat a také na ně reagovat. Detekce kolizí je jedna z úloh počítačové geometrie. Tou se však v této práci zabývat dále nebudeme. V této kapitole se budeme zajímat několika nejznámějšími algoritmy pro nalezení nejkratší cesty.

5.1 Dijkstrův algoritmus

Tento algoritmus se řadí mezi nejznámější pro vyhledání nejkratší cesty, tedy v našem případě mezi dvěma určenými body. Při hledání cesty rozlišujeme mezi grafy ohodnocenými a neohodnocenými (neohodnocený graf lze však na ohodnocený snadno převést). V případě neohodnocených hran přestavuje nejkratší cestu přechod přes nejmenší počet hran grafu a u ohodnocených grafů musí být splněna podmínka, která zajišťuje, že ohodnocení všech hran grafu musí být *kladné číslo* (v našem případě splněno vždy). Předpokládejme, že délka, označme například d_c, ze startovního bodu s k cílovému bodu c je nejkratší cestou. Tím je zřejmé, že $d_s = 0$. Ostatní vzdálenosti ze startovního bodu pak položíme rovné *nekonečnu*. Algoritmus dále pracuje na postupném zpřesňování odhadu délky nejkratší cesty ze startovního bodu k ostatním vrcholům, jak je znázorněno na obr. 14.



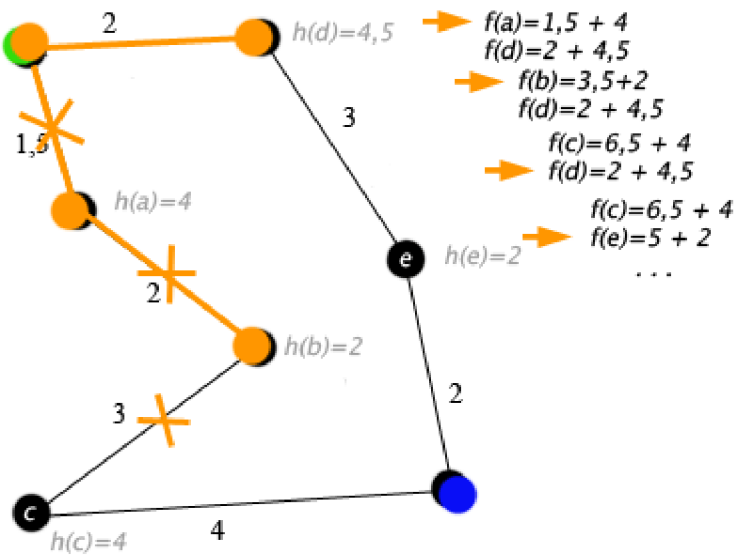
Obr. 14 Běh Dijkstrova algoritmu

Běh algoritmu je takový, že si pro každý vrchol v pamatuje délku nejkratší cesty, kterou se k němu dá dostat. Na začátku mají tedy všechny vrcholy v hodnotu $d_v=\infty$ (nekonečno symbolizuje, že neznáme cestu k vrcholu), samozřejmě kromě počátečního vrcholu s, který má $d_s=0$. Dále si algoritmus udržuje množiny Z a N, kde Z obsahuje už navštívené vrcholy a N dosud nenavštívené. Algoritmus pracuje v cyklu tak dlouho, dokud N není prázdná. V každém průchodu cyklu se přidá jeden vrchol v_{min} z N do Z, a to takový, který má nejmenší hodnotu d_v ze všech vrcholů v z N. Až algoritmus skončí, potom pro každý vrchol v je délka jeho nejkratší cesty od počátečního vrcholu s uložena v d_c. Tato kapitola čerpá z [2], [3] a [5].

5.2 Algoritmus A-Star

Algoritmus A* (A star) je verzí algoritmu uspořádaného prohledávání s tím, že hodnotící funkci f v bodě i získáme ze vztahu: $f(i) = g(i) + h(i)$, kde $g(i)$ je cena optimální cesty z počátečního do i-tého stavu a $h(i)$ je cena cesty z i-tého do cílového stavu. Cenou rozumíme libovolné nezáporné ohodnocení cesty. V mnoha úlohách ale funkce $g(i)$ a $h(i)$ neznáme a nahrazujeme je jejich odhady $g^*(i)$ a $h^*(i)$. Odhad funkce $h^*(i)$ vyjadřuje naše šance najít optimální řešení a je nositelem této heuristické informace. Proto tuto funkci nazýváme *heuristickou funkcí*. S

konstrukcí funkce *f(i)* vyvstává otázka tzv. přípustnosti algoritmu prohledávání. Říkáme, že algoritmus prohledávání stavového prostoru je *přípustný*, jestliže vždy nalezneme optimální cestu, pokud však tato cesta existuje. Lze ukázat, že pokud funkce *h*(i)* je nezáporná a menší nebo rovna skutečné ceně přechodu ze stavu *i* do cílového stavu, je algoritmus *A* přípustný. Potom funkci s touto vlastností nazýváme *přípustnou heuristickou funkcí* a algoritmus *A* používající tuto funkci nazýváme *algoritmem A** [8] (A star, na obrázku 15) .



Obr. 15 Ukázka hledání cesty pomocí A algoritmu*

Modifikace algoritmu A a A*

- Algoritmus A* zaručuje nalezení nejkratší cesty, pokud je heuristický odhad *h*(i)* menší, než skutečná cena do cíle. Heuristická funkce je přípustná, je-li $h^*(i) \geq 0$ a zároveň $h^*(i) \leq h(i)$, to znamená, že *h*(i)* je spodním odhadem funkce *h*(*i*) pro všechny uzly *i*. [3]

- Pokud položíme $h^*(i) = 0$, potom je prohledávání stavového prostoru řízeno pouze funkcí *g(i)*. Potom hovoříme o *algoritmu se stejnoměrnou cenou* (uniform-cost search). Je-li nalezen určitý cílový stav, algoritmus si zapamatuje jeho cenu cesty a ze seznamu OPEN vyřazuje uzly s vyšší cenou, než je zapamatována cesta řešení. Tento algoritmus dohledává stavový prostor, aby nalezl skutečně cestu s nejnižší cenou. [8]

- Položíme-li $h^*(i) = 0$ a $g(i) = 0$, algoritmus se stejnou cenou se degeneruje na slepé prohledávání do šířky (tedy v případě ceny rovno jedné). [8]

5.3 Ostatní metody

Je ještě mnoho metod pro určování dráhy robotu a k vyhledání optimální cesty, avšak jejich použití neznamená, že nalezená cesta bude skutečně tou nejkratší.

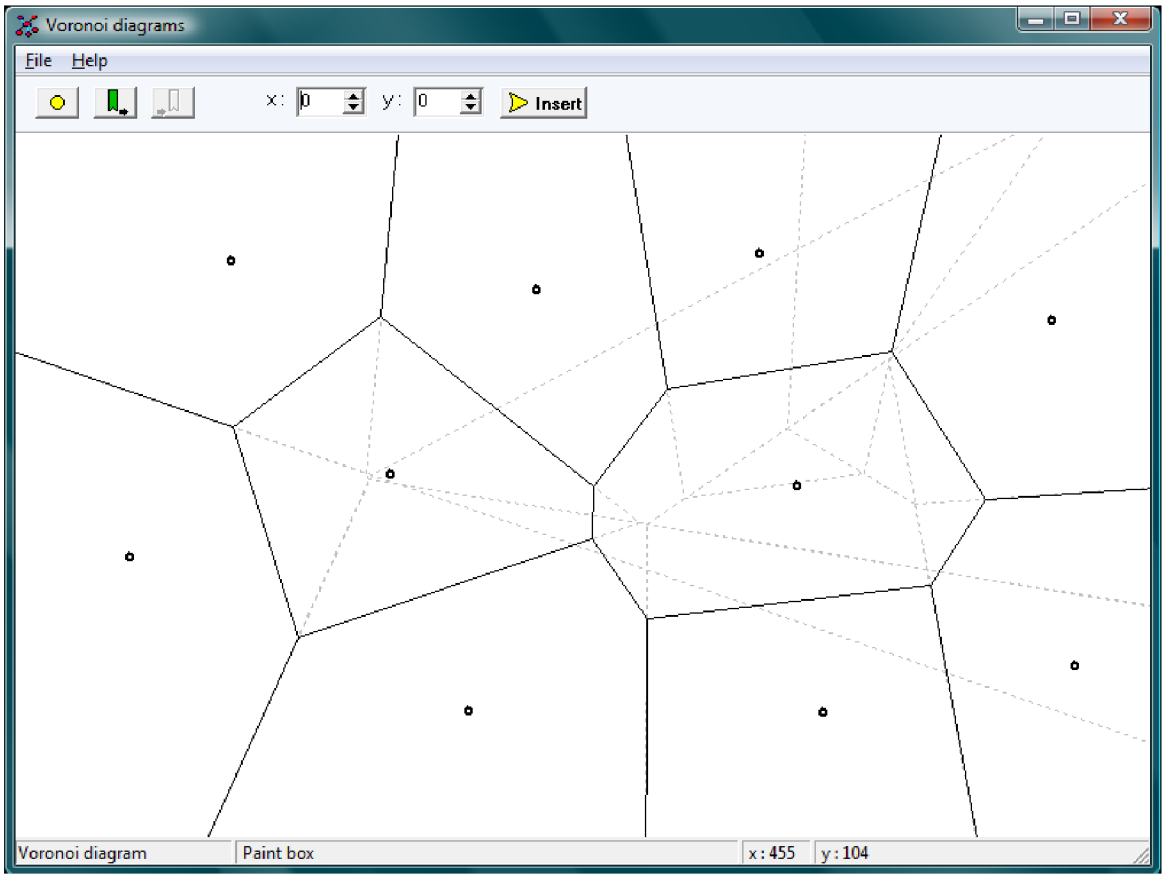
Stále více využívané jsou stochasticky heuristické metody, a to díky rozvoji výpočetní techniky, kde se tyto metody využívají stále častěji. Jako *heuristika* se chápe postup získání řešení problému, které však není přesné a nemusí být nalezeno v krátkém čase. Slouží však nejčastěji jako metoda rychle poskytující dostatečné a dosti přesné řešení, které však nelze obecně dokázat. Nejčastější použití heuristického algoritmu nalezneme v případech, kde není možné použít jiného lepšího algoritmu, poskytujícího přesné řešení s obecným důkazem.

Dalšími také často využívanými metodami jsou metody přibližného usuzování, mezi které patří například *Fuzzy* metoda vícekriteriálního hodnocení.

Těmito metodami se však dále v této práci zabývat nebudeme.

6 POPIS APLIKACE

Součástí bakalářské práce je implementace popsaných algoritmů. Program, který je přiložen k této práci je naprogramován v jazyce Delphi 7. Je možné jím vytvořit Voroného diagram a následným zadáním startovní a cílové pozice s nalezením nejkratší cesty mezi těmito dvěma body.



Obr. 16 Hlavní okno aplikace

6.1 Ovládací panely

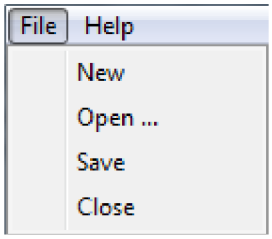
Hlavní okno programu (obr.16) obsahuje dva základní ovládací panely. Horní část menu obsahuje nabídky *File* a *Help*, které se dále větví na další podnabídky. Pod horní nabídkovou lištou je „pracovní“ panel, kde je na výběr z několika tlačítek (podrobný popis bude uveden níže). Dalším důležitým prvkem je „plátno“, na které se vykresluje Voroného diagram. Posledním prvkem je stavový řádek, který je rozdělen do tří částí. První poskytuje název aplikace, druhý vypisuje popis tlačítek, na kterém se kurzor nachází a poslední části udávají polohu myši za pomocí souřadnic.

6.1.1 Nabídka File a Help

Pomocí menu File lze provádět základní aplikační procesy, jako je vytvoření nového Voronoiova diagramu, uložení dat a následné načtení. Nabídka Help obsahuje prozatím základní informaci o programu.

Nabídka File (obr. 17)

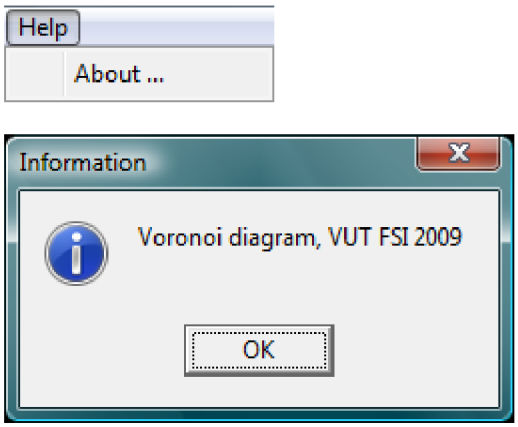
- New - vytvoření nového diagramu
- Open ... - otevření souboru s uloženými daty o diagramu
- Save - uložení dat o vytvořeném diagramu
- Close - uzavření programu



Obr. 17 Nabídka File

Nabídka Help (obr. 18)

- About ... - informace o programu (vyvolání okna s informací)

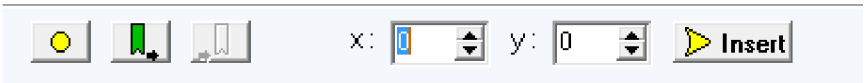


Obr. 18 Nabídka Help

6.1.2 Pracovní lišta

Tato lišta (obr. 19) obsahuje možnosti vkládání bodů na kreslící plochu.

První tlačítko (bráno zleva) nám umožňuje vložit body k vytvoření diagramu (defaultně nastaveno jako aktivní). Dalším tlačítkem vložíme startovní bod (stiskem deaktivujeme tlačítko s bodem) a po vložení startovního bodu se nám automaticky aktivuje cílové tlačítko (na obrázku 19 třetí zleva), které bylo doposud neaktivní. Další možnost vkládání bodů je pomocí tlačítka *Insert*, které umožní vložení bodu na námi určené souřadnice.



Obr. 19 Pracovní lišta

7 STRUKTURA A VÝSLEDKY PROGRAMU

Nyní si stručně popíšeme celkovou strukturu přiloženého programu. Mezi důležité objekty, které jsou použity v aplikaci patří *uzel*, *hrana*, *generátor* a *Voroného diagram*. Hlavním objektem je *Voroného diagram*, který všechny ostatní objekty sdružuje. Dalším, neméně důležitým objektem je *hrana*, ta nám utváří Voroného diagram, jedná se tedy o jeho hrany. Tato struktura je dále vázána na *uzel*, který nám hranu ukončuje (obsahuje dva uzly). Poslední neméně důležitou strukturou je *generátor*, který uchovává zpětné vazby na ostatní objekty, stejně jako předchozí dva, tedy uzel a hrana.

7.1 Tvorba diagramu

Pro tvorbu diagramu byl použit inkrementální algoritmus, který byl popsán v kapitole 4 - Algoritmy a konstrukce Voroného diagramu. Při spuštění aplikace je možno ihned zadávat body, před tím se však vytvoří automaticky tři pomocné generátory, které sestrojí rovnostranný trojúhelník. Důvod, proč se tyto pomocné body vkládají, vychází z poznatků, že Voroného diagram lze pro 3 body bez problému sestrojit. Cílem tohoto opatření je vznik diagramu ještě před vložením první překážky. Je zřejmé, že žádný bod nebude nikdy vložen tak, aby jeho poloha byla za hranicí polygonu. Jak se dále přidávají bodové překážky, utváří se Voroného diagram.

Popis procedury a vytvoření Voroného diagramu je popsán následovně:
vstup je bod p_i a výstupem je Voroného diagram $V(p_i)$

```
Procedure VoronoiDiagram.VytvorPolygon;

If SeznamVsechGeneratoru < 2 then exit;
 NarOblast := nejbližší bod k pi;

While NarOblast <> nil do
 For i:=0 to PocetHran do
  Prusecik := NajdiPrusecik(NarOblast,AktHrana);

  If Prusecik <> nil then
   If Neexistuje(Prusecik,LSP) then
    Pruseciku.Add(Prusecik);
    VsechnyPruseciky.Add(Prusecik);

  If Pruseciku.Count = 0 then
   Hranice := VytvorHranici(NovyGenerator,NarOblast);
   NarOblast:=nil;

  If VsechnyPruseciky.Count = 1 then
   If PosledniPrusecik <> nil then
```

```
            Hranice:=VytvorHranici(NarOblast,NovyGenerator,Pos
ledniPrusecik,Prusecik);
          PosledniPrusecik := Prusecik(Pruseciku[0]);
          NarOblast := NajdiNarusenouOblast(PosledniPrusecik,
NovyGenerator,NarOblast);

          If PosledniPrusecik = nil then
           Hranice := VytvorHranici (NarOblast,NovyGenerator,
Prusecici[0]);
           PosledniPrusecik := Prusecik(Pruseciku[0]);
           NarOblast := NajdiNarusenouOblast(Prusecik);

          If NarOblast = Predposledni then
           Hranice := VytvorHranici(NovyGenerator,NarOblast,
Prusecik,ZakoncBod);
           NarOblast:=nil;

          If Pruseciku.Count = 2 then
           ZakoncBod := Prusecik(Pruseciku[0]);
           Predposledni := NajdiNarusenouOblast(ZakoncBod);
           Prusecik := (Pruseciku[1]);
           PosledniPrusecik := Prusecik;
           NarOblast := NajdiNarusenouOblast(Prusecik);

     SeznamPruseciku.delete;
     ModifikujHrany(SeznamPruseciku);
```

Je jasné, že Voroného diagram může být vytvořen, když jsou zadány alespoň dva body, ty pak mají pouze jednu společnou hranu. Neexistuje zatím žádná další hrana, kde by mohl vzniknout průsečík. V tomto případě bude procedura ukončena, protože proměnné *NarOblast* se přiřadí hodnota *NIL*. Pokud přidáme další bod, vyhledá se jeho nejbližší soused a naleznou se průsečíky pro vytvoření nové hrany. Při vložení dalšího bodu budou nalezeny dva průsečíky (vždy vložen do uzavřené buňky), z nichž jeden je ukončovací a druhý bude průsečík. Tímto získáme dvě narušené oblasti, se kterými nalezneme průsečíky (budou právě dva). Následně se vytvoří hrana mezi novým bodem a generátorem narušené oblasti. Tato kapitola čerpá z [3].

7.2 Počáteční a koncový bod

Zadáním počátečního a koncového bodu vyvstává hned několik problémů. Aby bylo vůbec možné dráhu pohybu robota uskutečnit, je nutné znát některé důležité informace, neboť ze zadaných bodů nevedou žádné hrany. Musí se tedy vytvořit vlastní hrany, po kterých je robot schopen naplánovat dráhu. Je mnoho variant na propojení těchto dvou bodů, my se však zaměříme pouze na jednu z nich. Při vložení počátku a konce je využito přímé spojnice mezi těmito body.

Ovšem za předpokladu, že startovní bod leží v jiné buňce než cílový, dojde vždy k navázání na hrany vytvořeného Voroného diagramu a tím bude dráha robotu vedena v bezpečné vzdálenosti od bodových překážek. Dojde-li na případ, kdy oba body leží v jedné buňce Voroného diagramu je řešení jednoduše nalezeno spojením těchto bodů.

7.3 Nalezení nejkratší cesty

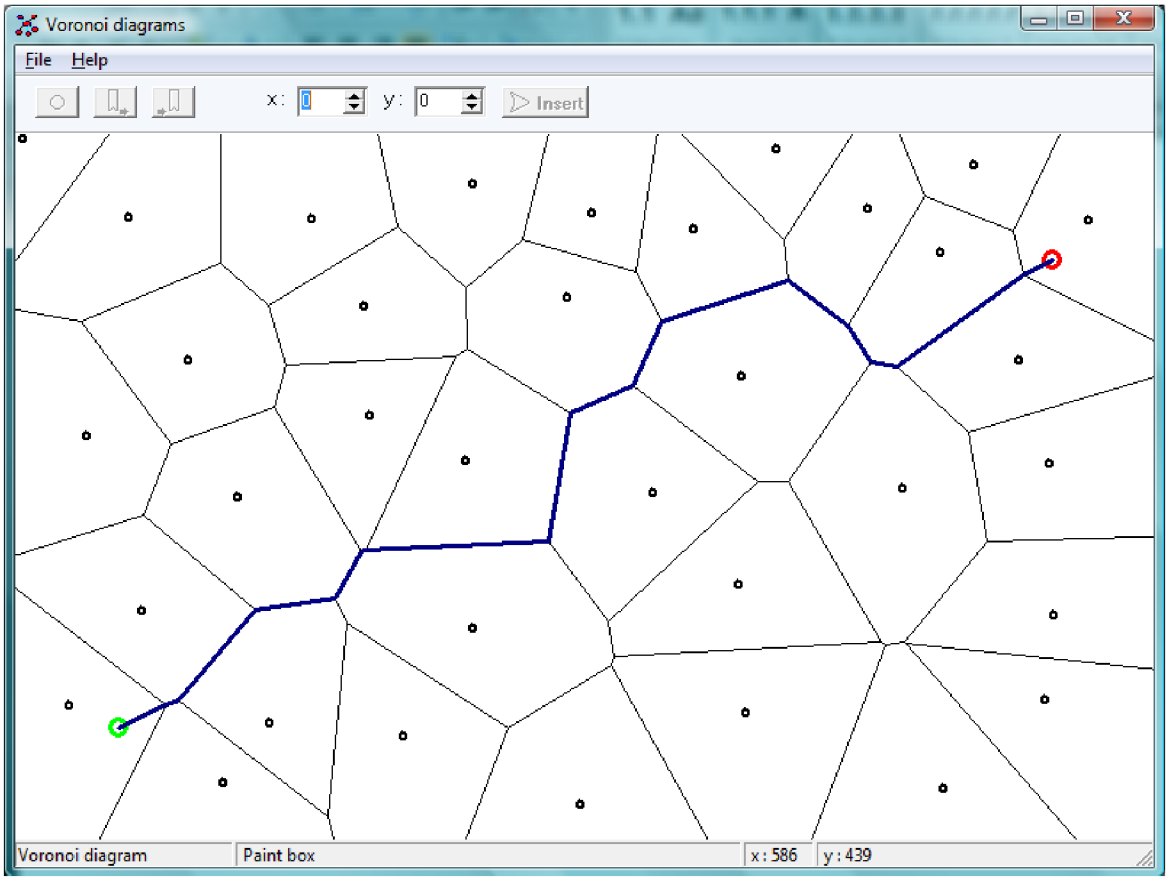
Pro plánování nejkratší cesty robota byl využit algoritmus A-star (popsán v kapitole 5). Jak bylo naznačeno algoritmus využívá dva seznamy *OPEN* a *CLOSE*. Kde seznam *OPEN* načte počáteční uzel s ohodnocením, kde první hodnota je vzdálenost počátečního bodu k aktuálnímu uzlu a druhou hodnotou je odhad vzdálenosti do koncového bodu. Následně je pak vybrán uzel s nejlepším ohodnocením. Do seznamu *OPEN* jsou pak vloženy všichni následníci uzlu, kteří ještě v seznamu *OPEN* nejsou a nakonec je samotný uzel vložen do seznamu *CLOSE*. Následně již bude pouze prováděno porovnávání ohodnocení s uzly, kteří již jsou v seznamu *OPEN* a pokud je toto ohodnocení menší, je hodnota uzlu změněna, a tím i ukazatel na předchůdce. Celý proces je opakován do té doby než je nalezena dráha nebo do vyprázdnění seznamu *OPEN* (v tomto případě by dráha nebyla nalezena). Tato kapitola čerpá z literatury [3].

Popis procedury:
vstupními parametry je počáteční a koncový bod, výstupními nejkratší cesta

```
Procedure OPENaCLOSE.VytvorSeznamy;

  Uzel := PrvniUzel;
  Open := Uzel;

  While Uzel <> nil do

     If Uzel = KoncovyBod then
       VypisCestu;
       Ukonci;
     else
       Expanduj(Uzel);
       Close.Add(Uzel);
       Open.Delete(Uzel);
       Uzel := NejOhodnoceni(OPEN);

     If Uzel = nil then CestaNeexistuje;
```

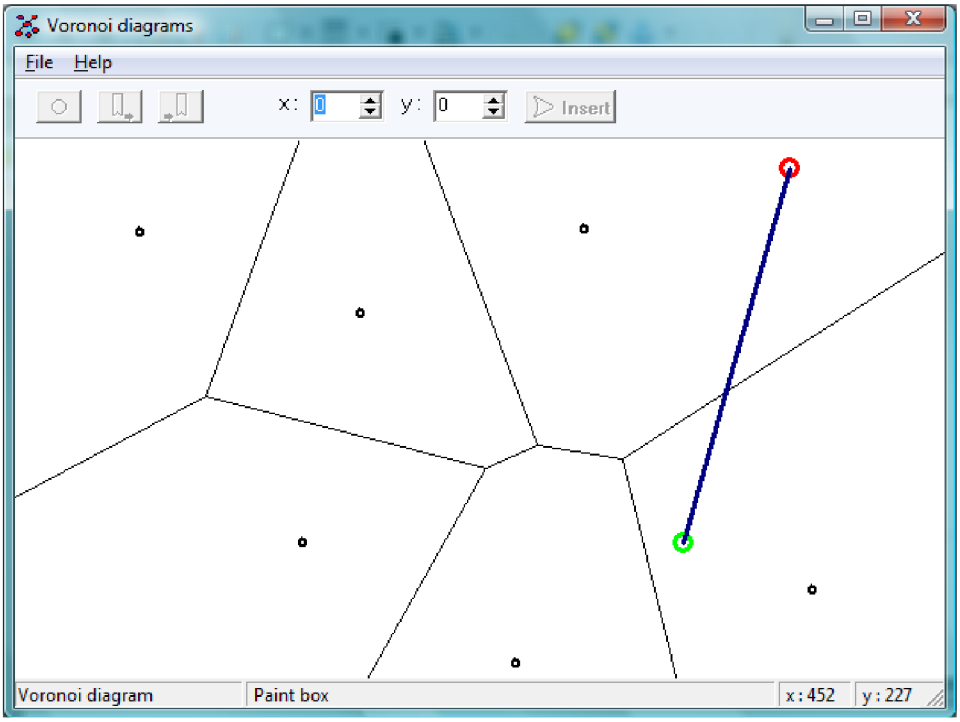
7.4 Výsledky aplikace

Na obrázku 20 je vykreslen Voroného diagram a následné vyhledání nejkratší cesty mezi dvěma vloženými body.

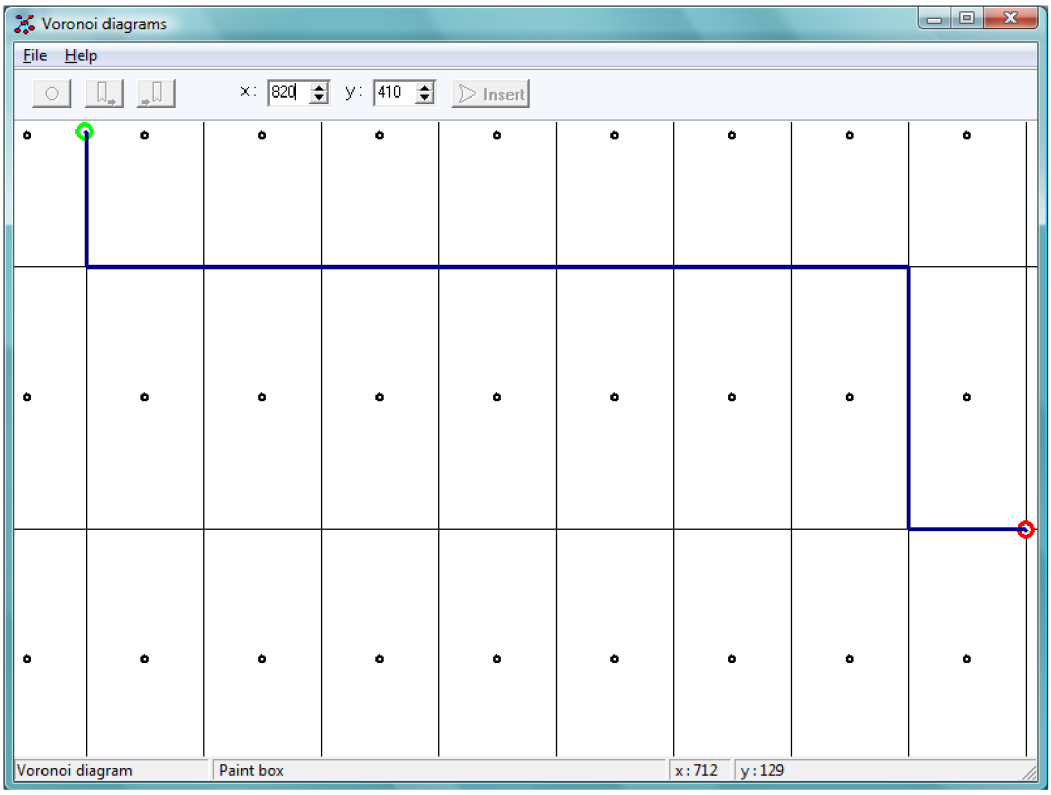


Obr. 20 Voroného diagram s bodovými překážkami

Na ostatních obrázcích (21 a 22) jsou ukázky různých specifikací diagramů (např.: degenerovaný Voroného diagram – obr. 22) a vykreslení drah robotu. Na obrázku 21 je pak znázorněna cesta, kde oba dva body nevyužijí hran diagramu, to může nastat například, když body leží v jedné buňce Voroného diagramu. Je patrno, že nejkratší cestou je zde spojnice těchto dvou bodů.



Obr. 21 Dráha robotu bez využití hran diagramu



Obr. 22 Degenerovaný Voroného diagram

8 ZÁVĚR

Konstrukce Voroného diagramu má v plánování dráhy robotu oproti jiným metodám výhodu hlavně v tom, že robot jedoucí po hranách tohoto diagramu udržuje maximální možnou vzdálenost od překážek, a tím i bezpečný průjezd mezi nimi. Nevýhodou je, že tato dráha nemusí být nejkratší a může vést mnohdy zbytečně daleko od překážky. Proto je výhodné implementovat na tento problém hned několik různých algoritmů, či je kombinovat. Tím dosáhneme větší efektivity a možnost projetí robota nejkratší možnou cestou, avšak v bezpečné vzdálenosti od překážek.

Cílem této práce bylo seznámit se s metodami počítačové geometrie a implementovat navržené algoritmy. Součástí práce je přiložený program, který využívá popsaných metod pro sestavení Voroného diagramu a k určení nejkratší dráhy robotu. Pro konstrukci Voroného diagramu byl v této práci vybrán přírůstkový (inkrementální) algoritmus, který je popsán v kapitole 4, a to hlavně pro jeho jednoduchou implementaci a koncepční jednoduchost. Pro určení nejkratší dráhy mezi startovním a cílovým bodem byl použit algoritmus A*, který je vymezen v kapitole 5. Tento algoritmus používá k ohodnocování uzlů, tvz. heuristické funkce a je optimální pro vyhledávání cest.

Pro plánování dráhy robotu, kde se vyskytují bodové překážky jsou tyto algoritmy plně dostačující. Pokud by se však v cestě vyskytovaly pravoúhlé, polygonální, či prostorové překážky je výhodnější kombinace Voroného diagramu s dalšími metodami, pro lepší efektivitu pohybu robotu. V budoucnu bych chtěl dále rozšířit tuto práci i přiloženou aplikaci o výše zmiňované problematice a nadále prohlubovat znalosti o využití Voroného diagramů.

SEZNAM POUŽITÉ LITERATURY

[1] de Berg, M., van Kreveld, M., Overmars, M. and Schwarzkopf, O: Computational Geometry: Algorithms and Applications. Springer-Verlag, Berlin, 2000 (2nd rev. ed.).

[2] PICH, V. *Aplikace Voroného diagramů při plánování dráhy robota*. Brno, 2008. Diplomová práce na Fakultě strojního inženýrství Vysokého učení technického v Brně na Ústavu mechaniky těles, mechatroniky a biomechaniky. Vedoucí diplomové práce doc. RNDr. Ing. Miloš Šeda, Ph.D.

[3] KVASNIČKA, J. *Aplikace Voronoiových diagramů v plánování dráhy robotu*. Brno, 2005. Diplomová práce na Fakultě strojního inženýrství Vysokého učení technického v Brně na Ústavu automatizace a informatiky. Vedoucí diplomové práce doc. RNDr. Ing. Miloš Šeda, Ph.D.

[4] NOVÁČKOVÁ, J. *Matematické nástroje analýzy v geomatice*. Plzeň, 2006. Diplomová práce na Fakultě aplikovaných věd Západočeské univerzity v Plzni na Katedře matematiky. Vedoucí diplomové práce doc. RNDr. František Ježek, Csc.

[5] Teorie grafů
http://www.uai.fme.vutbr.cz/~mseda/TG03_MS.pdf

[6] Voroného diagramy a jejich aplikace
http://num.kma.zcu.cz/MM-prace/Galerie%20MM%202007/Samkova-%20Voroneho%20diagramy%20a%20jejich%20aplikace.pdf

[7] Wikipedia
http://en.wikipedia.org/

[8] Prohledávání stavového prostoru
http://labe.felk.cvut.cz/~obitko/xkui/materialy/prostor.PDF