České vysoké učení technické v Praze
Fakulta elektrotechnická

Referát z předmětu
Výpočetní geometrie (36VGE)

JÁDRO POLYGONÁLNÍ OBLASTI

Jméno studenta:	Šafránek David
Studijní skupina:	5/38
Seminář:	středa 9:15
Semestr, školní rok:	zimní 2005/2006
Datum:	17.12.2005

Jádro polygonální oblasti

(kernel of a plane polygon)

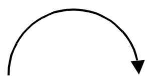
Anotace

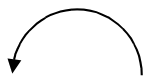
Následující referát vysvětluje pojem jádro polygonální oblasti (jádro polygonu) a popisuje základní algoritmus na jeho nalezení. Ten spočívá v průniku polorovin tvořených z hran původního polygonu. Dále detailně popisuje algoritmus, který konstruuje jádro v čase O(n), což je velmi výhodné při hromadném zpracování.

Definice použitých termínů

hrana	edge	e
vrchol	vertex	v
polorovina	half-plane	H
mnohoúhelník	polygon	P
jádro	kernel	K

Rotace:

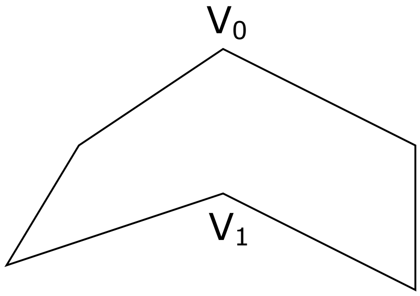
ve směru hodinových ručiček clockwise 

protisměru hodinových ručiček counterclockwise 

Polygon s neprázdným jádrem budeme označovat hvězdicovitý polygon (star-shaped , star polygon).

Terminologie vrcholů

Vrchol v_0 je konvexní (vypouklý, convex), je-li jeho vnitřní úhel < 180 stupňů. Naproti tomu vrchol v_1 je konkávní (introspektivní, zaměřený zpět, reflex), je-li jeho vnitřní úhel > 180 stupňů.
Vrcholy jsou vyznačeny na následujícím obrázku.



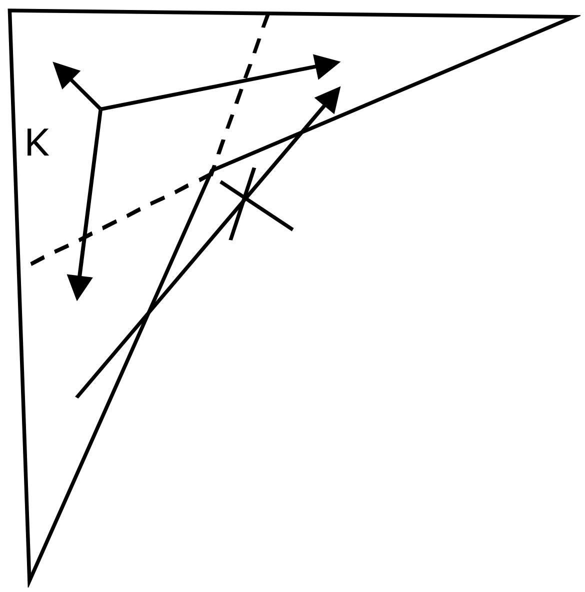
Obr. 1: Příklad polygonu s konvexním (v_0) a konkávním (v_1) vrcholem

Problém

Máme polygon s N vrcholy v rovině, zkonstruujeme jeho jádro (angl. Kernel)

Přesná specifikace problému

Jádro polygonu je množina bodů odkud jsou všechny body polygonu přímo viditelné, aniž by výhled křížila kterákoliv hrana, viz následující obrázek.



Obr. 2: Vyznačení jádra polygonu K a jeho konstrukce průnikem polorovin

Pro náš algoritmus je ale vhodnější následující definice:
Jádro je průnik N polorovin. Každá hrana polygonu P určuje polorovinu, ve které jádro musí ležet. Poloroviny jsou dvě, my uvažujeme levou, postupujeme-li proti směru hodinových ručiček.

Použití

Algoritmus se dá použít například na zjištění, zda je polygon hvězdicovitý či na vhodnou polohu umístění všesměrového vysílače, který musí ležet v jádru, aby pokryl celou oblast.

Základní algoritmus:

Použijeme-li průnik všech polorovin, uvažujeme v každém kroku částečně zkonstruované jádro. Časová složitost je v tomto případě O(N.log N).

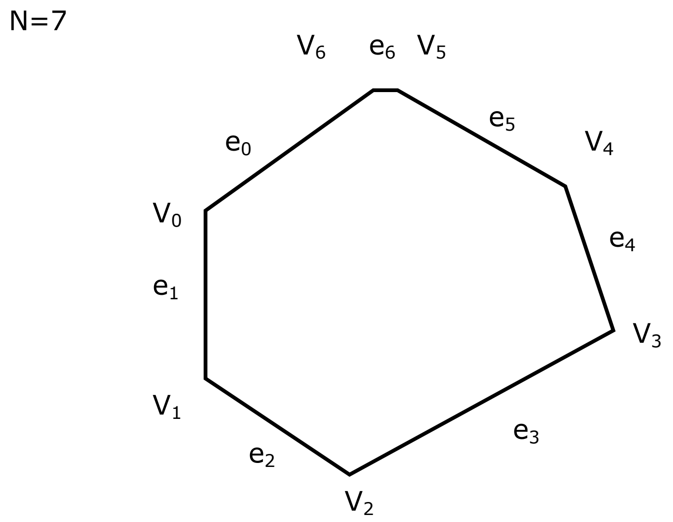
Popis postupu algoritmu

Algoritmus v pořadí prochází vrcholy P a konstruuje konvexní polygony K_i. Každý z těchto polygonů může i nemusí být ohraničený.
Polygon K_{i+1} je tvořen průnikem H_n n=0..i+1, a i=0..N-2 (N je počet vrcholů/hran počátečního polygonu).
Takto zkonstruované K_i je společný průnik polorovin ležících vlevo od hran P.
Tento výsledek má zřejmé důsledky:

1) K_{N-1}=K(P) tj. aplikujeme průnik na poloroviny získané ze všech hran zadaného polygonu
2) K_1 je nadmnožinou K_2 tj. $K_{i+1} = K_i$ "and" H_i

Vstup

Uspořádané vrcholy (hrany) - e_i následované v_i



Obr. 3: Označení vrcholů a hran pro polygon se 7 uzly

Výstup

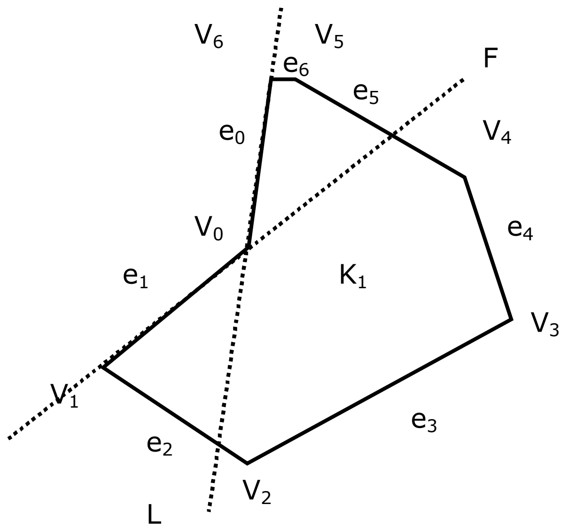
Když bude K(P) neprázdný, pak výstup bude také ve formě posloupnosti vrcholů a hran.

Triviální případ

N < 3: není polygon

N = 3: jádro je totožné s polygonem

Úvodní krok



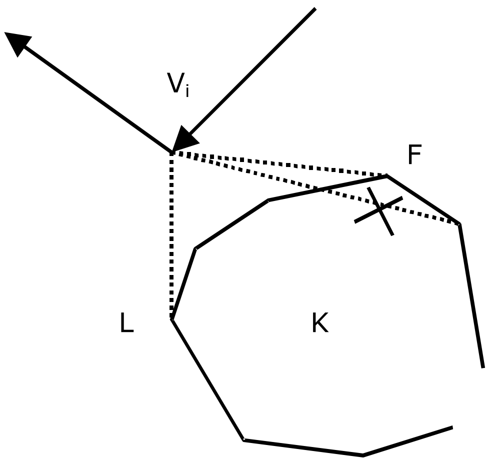
Obr. 4: Provedení úvodního kroku, zkonstruování K_1 z průniku polorovin definovaných hranou e_0 a e_1

Nalezení konkávního vrcholu a vygenerování výchozího neuzavřeného jádra K_1. Není-li vrchol nalezen, pak polygon P je konvexní, jedná se o triviální případ - jádro je totožné s polygonem K(P)=P. Tudíž připustíme, že v_0 je konkávní vrchol.
Určení F_1 jako bod v nekonečnu na přímce e_1 ve směru v_0 a L_1 obdobně na přímce e_0 ve směru v_0.

Shrnutí:
K_1=výseč e_1 v_0 e_0
$F_1 = e_1\ v_0$
$L_1 = e_0\ v_0$
Pozn. F a L se používají jako pomocné vrcholy pro rychlejší přechod od K_i do K_{i+1}.

Hlavní krok

Máme-li K_i, pak zhotovíme K_{i+1} tak, že zpracujeme další vrchol v pořadí (v_i) a spočteme K_{i+1}. Dále upravíme F_i a L_i tak, že spojnice s aktuálním bodem v_i polygonu neprotíná žádné jeho hany. F_i a L_i jsou tečné vrcholy k přímce obsahující v_i. V případě více těchto bodů volíme vzdálenější od v_i.



Obr. 5: Podmínka pozice bodů F a L v hranici jádra K

Předpoklad: vrcholy K_i jsou uspořádány po sobě proti směru hod. ručiček, pak rozlišíme několik případů, kde se F_i nachází

1) Vrchol je konkávní

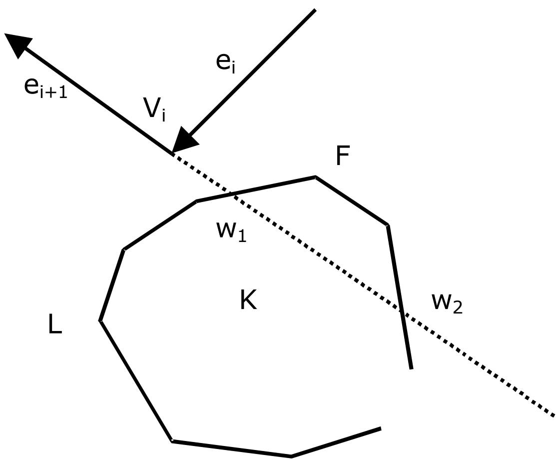
1.1) F_i leží na nebo vpravo od e_{i+1} v_{i+1}

Procházíme K proti směru hodinových ručiček od F dokud nenalezneme průsečík (w_1) s e_{i+1} v_{i+1}, dojdeme-li až k L, pal je K(P)=Ø.

Procházíme K po směru hodinových ručiček od F dokud nenalezneme průsečík (w_2) s e_{i+1} v_{i+1}, ořízneme K podle hrany w_1 w_2.

F: Je na konci hrany w_1 w_2

L: Procházíme K proti směru hodinových ručiček od L dokud nenajdeme vrchol w, takový, že následující vrchol leží vpravo od v_{i+1}(v_{i+1}w), pak L_{i+1}=w jinak zůstává rovno předchozímu L_i.



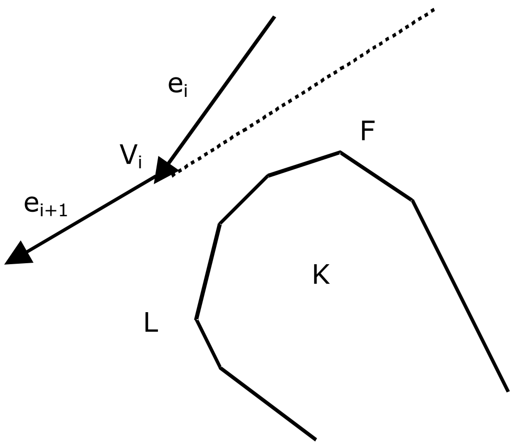
Obr. 1.1: Konkávní vrchol, F leží vpravo

1.2) F leží vlevo od e_{i+1} v_{i+1}

K zůstává stejné.

F: Procházíme K protisměru hodinových ručiček od F dokud nenajdeme vrchol w, takový, že následující vrchol leží vpravo od v_{i+1}(v_{i+1}w), pak F_{i+1}=w jinak zůstává.

L: stejně jako v případě 1.1



Obr. 1.2: Konkávní vrchol, F leží vlevo

2) Vrchol je konvexní

2.1) L leží na nebo vpravo od e_{i+1} v_{i+1}

Procházíme K po směru hodinových ručiček od L dokud nenalezneme průsečík (w_1) s e_{i+1} v_{i+1}, dojdeme-li až k L je K(P)=Ø.

Procházíme K proti směru hodinových ručiček od L dokud nenalezneme průsečík (w_2) s v_i $e_{i+1.}$ Provedeme oříznutí hranou w_1 w_2.

Nenalezli jsme (K je neohraničené): zjistíme zda nové K je či není ohraničené.

Počet průsečíků e_{i+1} s K

0)

F: Jako v případě 1.2

L: Je na konci hrany w_1 w_2

1)

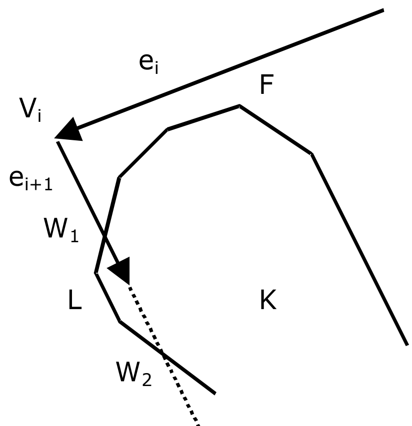
F: $F_{i+1}=w_1$

L: Je na konci hrany w_1 w_2

2)

F: Jako v případě 1.2

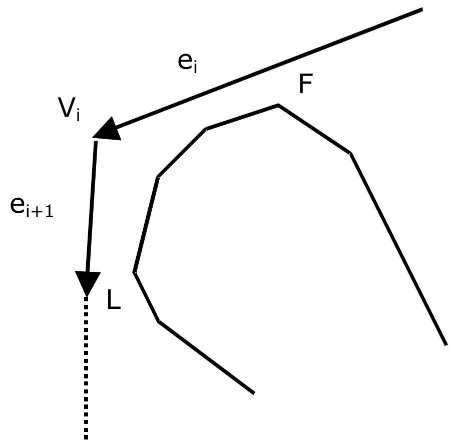
L: Je na konci hrany w_1 w_2



Obr. 2.1: Konvexní vrchol, F leží vpravo

2.2) L leží vlevo od e_{i+1} v_{i+1}

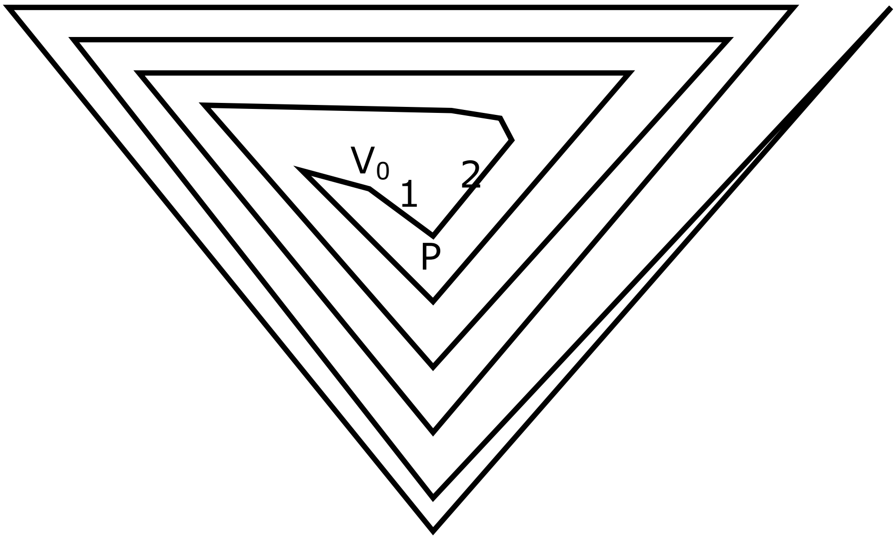
K zůstává stejné.
F: stejně jako v případě 1.2
L: Je-li K ohraničené tak jako 1.1 jinak zůstává



Obr. 2.2: Konvexní vrchol, F leží vlevo

Hodnocení operační a paměťové složitosti algoritmu

Jádro polygonální oblasti může být zkonstruováno s průměrnou časovou složitostí O(n), a s paměťovou složitostí taktéž řádu O(n).
Nejhorší časová složitost nastane pro instanci na obr. 6 - polygon bez jádra, kde má daný algoritmus časovou složitost $O(N^2)$. Začínáme ve vrcholu v_0 a postupně tvoříme částečné jádro. S přibývajícími vrcholy kroužíme kolem jádra, a časová složitost je u každého vrcholu O(N), jelikož toto opakujeme N násobně je celková složitost $O(n_2)$. Můžeme si to představit jako součet řady 1..N, což je rovno (N . N + 1) / 2.



Obr. 6: Příklad instance polygonu bez jádra, kdy časová složitost dosáhne $O(N^2)$

Řešení snížení časové složitosti na přijatelnou míru je ukončit algoritmus jakmile obteče v úhlu větším než 3π kterýkoliv bod částečného jádra K.

Další algoritmy

Competitive strategy

Algoritmus je taktéž známý pod názvem závodní strategie. Délka cesty pro nalezení bodu jádra polygonální oblasti je maximálně 5.333x delší než nejkratší cesta, předpokládá rozhled 360° - využití v robotice.

Zobecnění do více prostorů

Při přechodu do trojrozměrného prostoru je třeba přímky nahradit rovinami, poloroviny poloprostory, a najít uspořádání všech trojúhelníků (tesselace) v objektu tak, aby následující měl průnik s dosavadním jádrem.

Seznam použité literatury

[1] Applet: http://web.informatik.uni-bonn.de/I/GeomLab/VisPolygon/index.html.en
[2] Preperata, F.P., Shamos, M.I.: Computational Geometry An Introduction. Berlin, Springer Verlag,1985
[3] Závodní strategie: http://www.pi6.fernuni-hagen.de/publ/tr211.pdf